ITEA 2 Project 10014

EASI-CLOUDS - Extended Architecture and Service Infrastructure for Cloud-Aware Software

Deliverable

D5.10 – Final Report on Cloud Computing

Editors:

Juhani Toivonen, Seppo Hätönen

University of Helsinki

Security	Public
Version	1.0
Author	UH
Pages	90

History of Changes

Version	Author, Institution	Changes
0.1	Juhani Toivonen, UH	Started the document, initial structure plan in place.
	Wolfgang Thronicke, Florian Ostermair, ATOS	Contributions on Application management and SaaS enablement for legacy applications.
	Claudine Pelegrin-Bomel, Francois Verbeck, Jamie Marshall, Bull	Contributions on Cloud Federation and Brokerage, generally and in context of EASI-CLOUDS.
	Jakob Tonn, Thomas Konnerth, DAI-Labor	Contributions on SLA monitoring, negotiation and policies.
	Heikki Nousiainen, F-Secure	Contributions on data privacy and security standards.
	Hans-Joachim Goltz, Steffen Unger, Fraunhofer Fokus	Contributions on resource reservation.
	Juhana Peltonen, GearShift Group	Contributions on cloud computing market trends and situation.
	Bassem El Zant, Maurice Gagnaire, Institut Telecom (Telecom ParisTech)	Contributions on cloud computing pricing types, models and strategies.
	Mohamed Mohamed, Institut Telecom (Telecom SudParis)	Contributions on IaaS infrastructure monitoring solutions.
	Janne Lautamäki, Jari Redsven, Leonidas	Contributions on user-centric application development.
	Carsten Zoth, Orga Systems	Contributions on real-time rating, charging and Billing as a Service.
	Jörn Altmann, Netsanet Haile, Seoul National University	Contributions on smart placement, cloud federation and pricing models.
	Farshad Ahmadighohandizi, Tampere University of Technology	Contributions on cloud software development.
	Gregory Cunha, Romain Ferrari, El Hadi Cherkaoui, Thales Services	Contributions on identity management in cloud services.
	Juhani Toivonen,	Contributions on PaaS services and

	Seppo Hätönen, Tomi Ronimus, Toni Ruottu, University of Helsinki	frameworks.
	Krishnaprasad Narayanan, Lars Nagel, University of Mainz Mainz	Contributions on Infrastructure as a Service frameworks
1.0	Juhani Toivonen, Seppo Hätönen, Sasu Tarkoma University of Helsinki	Combined the contributions, removed redundancy, unified typography and citation style. Add a concluding chapter.

Abstract

The EASI-CLOUDS project has designed and developed a federated European cloud platform. In this report, we discuss the state of the art pertaining to EASI-CLOUDS related themes in cloud computing, take a view on the current business conditions, and describe selected contributions EASI-CLOUDS has made to address the key technological and business challenges of federated cloud computing.

Table of contents

History of Changes **2**

Table of contents **4**

1 Executive Summary **6**

2 Context **7**

2.1 Project 7
2.2 Work Package 7

3 State of the Art **8**

3.1 Infrastructure-as-a-Service 8
3.1.1 IaaS Cloud Platforms *9*
3.1.2 Smart Placement *19*
3.2 Platform-as-a-Service 22
3.2.1 PaaS Frameworks *22*
3.2.2 Software Engineering and Application Development Using MIDEaaS *24*
3.2.3 Cloud Computing and User-centered Design *28*
3.3 Brokerage and Federation 29
3.3.1 Cloud Brokerage *29*
3.3.2 Cloud Federation *31*
3.4 Management, Monitoring, Configuration and Post-configuration 34
3.4.1 Application Management *34*
3.4.2 Infrastructure Management *37*
3.4.3 Identity Management *40*
3.4.4 Resource Reservation *41*
3.5 Real-time Rating, Charging and Billing 42
3.5.1 Terms and Definitions *42*
3.5.2 Real-time Rating, Charging and Billing Mechanisms *44*
3.5.3 Standards *45*
3.5.4 Market Requirements *48*
3.5.5 Summary *49*
3.6 Data Privacy and Security 50
3.6.1 Data Privacy *50*
3.6.2 Data Security Standards *51*

4 Evolution in Business **52**

4.1 Summary of The General Public Cloud Computing Market 52
4.1.1 A More Detailed Look at the Cloud Value Chain *54*
4.1.2 Cloud Business Consulting Services *55*
4.1.3 Cloud IT Consulting Services *55*
4.1.4 Cloud Brokerage *56*
4.1.5 Cloud Federation *57*
4.1.6 Traditional Service Types *58*
4.1.7 Platform Enablers *59*
4.1.8 Infrastructure Providers *59*
4.2 Pricing and Revenue Sharing in Cloud Computing 60
4.2.1 State of the Art in Pricing Strategies in Cloud Computing *60*
4.2.2 Pricing Types *61*

4.2.3 Pricing Models *63*
4.2.4 Existing Strategies *63*
4.2.5 Revenue Sharing in a Federated Cloud *64*
4.2.6 Existing Strategies for Revenue Sharing in a Federation *64*
4.3 Research on Pricing Models 65

5 EASI-CLOUDS Innovation 67

5.1 Real-time Billing as a Service 67
5.1.1 Existing Billing as a Service Offers *68*
5.1.2 BaaS - Results and Benefits *70*
5.2 Cloud Federation 71
5.3 SLA Negotiation 72
5.3.1 SLA Document Models *72*
5.3.2 SLA Negotiation *73*
5.3.3 SLA Monitoring and Policy Enforcement *74*
5.4 SaaS Enablement of Legacy Applications 75
5.4.1 The Challenge *75*
5.4.2 The FreeSurfer Cloud Service *75*

6 Conclusion 77

1 Executive Summary

This report describes the state of the art on many aspects of cloud computing, gives a high level overview about cloud-related business, and describes the key concepts, techniques, and models that EASI-CLOUDS has contributed for advancing the state of cloud computing.

Chapter two briefly positions the presented work in the context of the EASI-CLOUDS project.

Chapter three presents the state of the art pertaining to different cloud technologies, such as IaaS, PaaS, and service and resource brokering. We discuss the state of the art for the different layers of the cloud computing architecture. We approach brokerage and federations from a technical point of view, and discuss management, monitoring and billing solutions available in the market today.

In chapter four we give an overview of the current cloud computing market and cloud computing as a business. The market overview includes descriptions on how the cloud market is segmented by the level of service (IaaS, PaaS…) and regionally. Pertaining to cloud computing business, we discuss different models for pricing, revenue sharing, and brokering and federation.

We describe the key innovations of EASI-CLOUDS in chapter five. We have identified real-time billing as a service, cloud federation, automated SLA negotiation, and SaaS enablement for legacy applications as the key innovations.

2 Context

2.1 Project

This document describes the state of the art and business conditions around the themes related to the project, and presents some of the innovative ways that the project has contributed to these themes.

2.2 Work Package

This document is part of Work Package 5 - Dissemination & Demonstration. The purpose of the work package is to endorse visibility and public knowledge of EASI-CLOUDS. The document works toward this goal by being an open description about some of the most innovative work around state of the art technologies in EASI-CLOUDS.

3 State of the Art

3.1 Infrastructure-as-a-Service

The Infrastructure-as-a-Service (IaaS) is a cloud computing service that allows users to setup virtual machines with a pre-defined amount of resources including different types of storage and connect them using software-based networking services. In recent years, many open source cloud infrastructure management software have been developed. Prominent examples are OpenStack, OpenNebula, Eucalyptus, VMware, Nimbus and CloudStack. The following section lists the available IaaS frameworks and describes the IaaS frameworks hosted by various partners in the project and further describes the infrastructure software stack that is part of each of the framework.

Cloud computing introduces interactions between cloud (infrastructure) providers and cloud service providers. These entities have different responsibilities depending on the service provided by the cloud. An infrastructure provider is defined by the NIST as "*a person, organization, or entity responsible for making a service available to interested parties*", while a service provider is "*a person or organization that maintains a business relationship with, and uses services from, cloud providers*"[1]. In an Infrastructure-as-a-Service (IaaS) environment, a cloud provider acquires the physical computing resources such as the servers, networks and storage. The provider then deploys an IaaS cloud framework responsible for managing the pool of physical resources, and making this infrastructure available for cloud consumers (service providers) through a set of service interfaces and computing resource abstractions (virtual machines and virtual network interfaces). On the other side, a service provider will use these physical resources based on the user's service specification (computational and bandwidth needs) and IaaS cloud deployment models.

In fact, two cloud types may exist: (a) public cloud which services are made accessible for any cloud consumer over the Internet, and (b) private cloud, which services are available for one cloud consumer (generally an organization). Therefore there exists multiple deployment models, where each one will define how exclusive the computing resources of these clouds are made to a cloud customer, thus enable different business models. These IaaS clouds deployment models are defined as:

a) Public deployment: models the usage of an IaaS public cloud by any cloud consumer over the Internet.
b) Private deployment: models the usage of an IaaS private cloud by one organization. The private cloud can be provided by the same organization consuming it, therefore called on-site private cloud, or hosted by a different organization and known as outsourced private cloud.
c) Hybrid deployment: models the usage of both, private (on-site or/and outsourced) and public cloud by a private cloud consumer.
d) Broker deployment: models the usage of a public cloud offering as a service, the management of different transactions between multiple public clouds and any cloud consumer over the Internet.
e) Federation deployment: models the usage of a public cloud, having a contract with one or multiple other public clouds specifying a cooperation agreement between corresponding parties. This agreement can specify for example the portion of physical resources each cloud can use from the other.

However in order to deploy an IaaS public or private cloud, an IaaS cloud platform is needed on top of physical resources.

3.1.1 IaaS Cloud Platforms

IaaS cloud platforms are software solutions installed (completely or partially) on servers in order to manage the underlying physical resources and offer the cloud consumer a set of services. These services are accessible via APIs, where each API requires certain authorizations from the cloud consumer in order to be used. Differences between IaaS cloud platforms lies in the virtualization system (hypervisors they support), the set of services provided by APIs, the user's managing techniques, and the network configurations.

Virtualization shifts the thinking from physical to logical infrastructure, where physical resources of a set of hardware components (e.g. physical servers) are considered as logical resources rather than separated physical resources. Therefore virtualization creates an abstraction layer between actual computing, storage and network hardware, and the software running on them. Thus allowing different operating systems contained in isolated virtual machines running on the same physical substrate. This abstraction layer is called virtualization layer. It is created and managed by a software or firmware component known as "hypervisor". Table 3.1 presents some of the IaaS cloud-platforms available on the market.

Although the IaaS cloud computing is a recent research domain and business model, several solutions were developed in past years. Some of these solutions are open source for development purposes, some are complete commercial solutions for companies wishing to provide cloud infrastructure services, while others are combination of both. OpenStack is an open source IaaS cloud platform, with a large community, which is growing every year. OpenStack is widely used in the research community, giving new users and researchers a knowledge base of forums and solved problems available online. So OpenStack delivers services satisfying market and research demands, with a large online support for users. Consequently, OpenStack code increased ten times in two and a half years [2], and several versions were released while maintaining compatibility with legacy releases.

IaaS Cloud Platform Name	Supported Hypervisor(s)	License
Abiquo	ESX, ESXi, Hyper-V, Citrix XenServer / Xen, Virtual Box and KVM	Community (free) and enterprise editions
CA 3Tera AppLogic	Xen	Commercial
CloudStack	Xen, Hyper-V, VMware vSphere and KVM	Open source
Convirture ConVirt	Xen, Hyper-V, VMware and KVM	Open source and commercial versions
Elastic Stack	KVM	Commercial
Enomaly Elastic Computing Platform (ECP)	Xen, VMware and KVM	Commercial
Eucalyptus	VMware vSphere and KVM	Open source with commercial support
HP Cloud System	VMware vSphere and KVM	Enterprise oriented commercial solution
IBM Cloudburst	PowerVM, z/VM, ESX, Xen and KVM	Enterprise oriented commercial solution
In continuum Cloud Controller	VMware, Hyper-V and Citrix XenServer / Xen	Commercial solution
Novell Cloud Manager	Xen, Hyper-V and VMware vSphere	Enterprise oriented commercial solution
OnApp	Xen, VMware and KVM	Commercial
OpenNebula	Xen, Hyper-V, VMware and KVM	Open source
OpenQRM	LXC, OpenVZ, Citrix XenServer / Xen, VMware and KVM	Community and enterprise editions
OpenStack	LXC, QEMU, UML, Xen, Hyper-V, VMware vSphere and KVM	Open source
Parallels Automation for Cloud Infrastructure (CI)	Parallel hypervisor	Commercial
VMware vCloud	VMware	Commercial
Xen Cloud Platform (XCP)	Citrix XenServer / Xen	Open source

Table 3.1: IaaS platforms available on the market

In the project, IaaS frameworks have been setup by universities and software organizations, which are used for the deployment of legacy applications and newly implemented components. The framework is setup at the following institutions: University of Mainz, University of Evry and Nexedi. Section 3.1.1.10 summarizes the EASI-CLOUDS infrastructure software stack available at Uni. Mainz and the remaining sections describe the essential components of an infrastructure cloud, available virtualization software packages and how these components are utilized in EASI-CLOUDS.

3.1.1.1 Hypervisor

A Hypervisor is virtualization software that is responsible for the lifecycle management of virtual machines. It is also called a Virtual Machine Manager or Virtual Machine Monitor (VMM). Table 3.2 describes frequently used hypervisors and different cloud software packages, which support them.

Software	VMware ESXi	Xen, Xen server	KVM	LXC, QEMU, UML, Power VM	Hyper-V	Oracle VM
OpenStack[3]	Yes	Yes	Yes	Yes	Yes	Yes
OpenNebula[4]	Yes	Yes	Yes	No	Yes [5]	No
Eucalyptus[6]	Yes	Yes	Yes	No	No	No
VMware[7]	Yes					
Nimbus[8]	No	Yes	Yes	No	No	No
CloudStack[9]	Yes	Yes	Yes	No	Yes	Yes[10]

Table 3.2: Hypervisor-support by IaaS platform [11]

Commonly used cloud software packages such as OpenStack and OpenNebula allow live migration of virtual machines, which is supported by some hypervisors. In EASI-CLOUDS, the infrastructure software stack is hosted at the University of Mainz, which uses OpenStack as the cloud management software and Libvirt and KVM as the default hypervisors. Table 3.3 lists different hypervisors and some of their essential features.

3.1.1.2 Image Store

The virtual machines are instantiated from the images that are available in the image store whose interface offers basic database operations for its management. Glance [12], OpenStack image service, supports the aforementioned functionalities and facilitates every user to upload and set the visibility of these images. Besides, it can also store disk and VM images in different back ends such as file, Swift[13], Cinder[13], S3[14], Ceph[15] and iSCSI[16].

OneImage[1], the CLI tool in OpenNebula, helps the administrators and users to manage and set up VM images. Similar to OpenStack, OpenNebula uses different image datastores[2] e.g. file-system, Ceph, VMFS (Virtual Machine File System) and LVM. CloudStack offers default templates[3], which is a virtual disk image that includes one of a variety of OS that the user can choose while creating a new instance. The templates support different hypervisors such as XenServer, KVM and VMware vSphere. vSphere provides a logical container VMFS, for

[1] http://docs.opennebula.org/4.4/user/virtual_resource_management/img_guide.html
[2] http://docs.opennebula.org/4.4/administration/storage/sm.html#sm
[3] http://cloudstack-administration.readthedocs.org/en/latest/templates.html

storing virtual machine images and files. Depending on the type of storage, Network File System (NFS) can also be used for backing the VM images.

Hypervisor	License	Technique	Live-Migration
Hyper-V	Proprietary (Microsoft)	Hardware virtualization	Yes
Kernel-based Virtual Machines (KVM)	GNU GPL	Kernel level virtualization	Yes
Linux containers (LXC)	GNU GPL v2.1	Operating system level / Container virtualization	No
User Mode Linux (UML)	GNU GPL	Kernel level virtualization	No
VMware ESXi	Proprietary (VMware)	Hardware virtualization	Yes
Xen	GNU GPL v2	Paravirtualization	Yes

Table 3.3: Features of different hypervisors

In the project, Glance is used that provides the basic VM images to the users for building the cloud components and applications. A check pointing mechanism periodically takes snapshots of running VMs so that they can be restored in case of a failure.

3.1.1.3 Storage

Storage is required by most cloud applications. The reference applications in EASI-CLOUDS for medical image processing, video gaming, engineering and photo stitching need large amounts of storage. FreeSurfer, the software in the medical domain, for instance, requires gigabytes of storage for storing the MRI scans and brain images. By default, virtual machines consist of two disk partitions namely, root and ephemeral disk. These disks are volatile by nature i.e. they are removed once the virtual machines are terminated. Table 3.4 lists the storage services offered by different cloud software.

Amazon supports block storage using its Elastic Block Storage (EBS)[17] interface and object storage using Simple Storage Service (S3) [14]. S3 is considered to be the de-facto standard that is followed by other cloud software packages. OpenStack, OpenNebula and Eucalyptus support both block and object storage. Cinder, the block storage of OpenStack, provides a persistent block storage service whose interface manages the creation, attaching and detaching of the external volumes to servers. The scalable object storage Swift is used for storing static data such as images, emails, backups, photos and archives.

In EASI-CLOUDS, the block storage is used for the FreeSurfer use-case. VM of a specific flavour is setup where an additional volume is created and attached to the VM that stores the MRI scans and brain images. The size of this shared block device is several hundred gigabytes.

Software	Object storage	Support for S3	Block storage
OpenStack	Swift	Yes	Cinder
OpenNebula	Image storage	Yes	Yes
Eucalyptus	Walrus	Yes	Storage[4] controller
VMware			
Nimbus	Cumulus	Yes	
CloudStack	Integration with Swift	Yes	

Table 3.4: Storage services offered by different IaaS platforms

3.1.1.4 Networking

The networking components of the cloud software packages provide features that include the communication between virtual machines, the configuration of private and public IP addresses, mechanisms for accessing the VMs outside the cloud and setting up the ports and firewalls.

The OpenStack project Neutron[18], is a standalone component which handles tenants' requests and defines the communication between its services. The module provides drivers that support the following network types: local, flat, VLAN, GRE and VxLAN. Besides their support for multiple drivers, they provide APIs that help the tenants to setup networking policies and offer support for adding and integrating new plug-ins that introduce advanced networking capabilities. Some of the commonly used plugins are Open vSwitch[19], Linux bridge, Mellanox neutron and CISCO UCS. Neutron also supports monitoring of network protocols using Netflow, sFlow and SPAN / RSPAN. Other cloud management software such as OpenNebula[5] and VMware[6] also use various drivers such as Open vSwitch to create the virtual networks. In the project, the traditional flat networking mode is used (where there is a single network per user) which assists the VMs to communicate with each other using their internal and external IP addresses. The virtual servers are automatically assigned with a new vNIC or private IP address from the single network during the time of VM instantiation.

3.1.1.5 User Management

The essential element in cloud is the user management that needs the establishment of the identity of a user (i.e. authentication) and the management of rights (authorization). It must be ensured that users are granted access only for their accounts and virtual resources (virtual machines, image and volumes) and other users are denied from accessing them. Some of the essential features offered by user management service are

- User interface (e.g. web portal, CLI, configuration files) for configuring the users and tenants.
- Defining access permissions (rights management) for users / tenants.
- Support for multiple identity backends, such as LDAP, KeyValueStore, PAM, SQLAlchemy.
- Using password, X. 509 certificates and SSH-RSA key for secure authentication.
- Secure communication using SSL / TLS, X.509 certificates and custom tokens.

[4] https://www.eucalyptus.com/docs/eucalyptus/4.0/index.html#user-guide/understanding_storage.html

[5] http://docs.opennebula.org/4.4/administration/networking/openvswitch.html

[6] http://blog.ipspace.net/2012/02/nicira-open-vswitch-inside-vsphereesx.html

In the Mainz testbed, the identity management is based on OpenStack component Keystone[3]. The requests generated from the users / tenants are forwarded to the Identity service which performs credential validation and returns a token which is used for successive requests. Additional information about the extended identity management developed in the project can be referred in section 3.4.3.

3.1.1.6 Scheduling

In general, scheduling in clouds can be performed at three layers: selecting the cloud (e.g. Broker) for deploying the user's application, infrastructure level scheduling that selects the host for placing the VM / VMs and application level scheduling. This section concentrates only on the scheduling policies used at the host level and Table 3.5 describes different methods used by various cloud management software.

OpenStack supports addition of new schedulers and they can be added / removed in the form of plug-ins [20]. It provides load balancing (i.e. checks the available resources such as CPU, cores, RAM, disk space), random selection and allows the use of its simple scheduler. In OpenNebula, the server is selected either based on the number of VMs placed on it or based on the amount of load present on it. It also provides an energy saving scheduler where the server that has existing load is filled with virtual machines before new servers are assigned to virtual machines.

Software	**Random**	**Lowest load**	**Energy saving techniques**	**Round robin**	**Configurable**	**Support for adding new schedulers**
OpenStack (Icehouse)	Availability zone, host aggregates and random selection	Filter scheduler				Yes
OpenNebula	Fixed policy	Load aware and stripping policy (#VMs)	Packing policy (#VMs)		Match making algorithm with rank scheduling	Yes
Eucalyptus		Greedy	Power save	Round robin		
VMware		Distributed resources scheduler (DRS)	DRS		DRS	
Nimbus						Yes. Extended with Oracle Grid Engine (OGE) and Portable Batch System (PBS)
CloudStack	First fit or Round robin	Disperse	Fill first			Yes

Table 3.5: Scheduling methods in IaaS platforms

Similar to OpenStack, OpenNebula can also be configured to use its own scheduler and provide mechanism to add external schedulers. The Distributed Resource Scheduler (DRS) is a feature in the VMware vSphere that allows improvement of the service levels by scaling the resources in virtual machines, balances the workloads to achieve an optimal performance and automatically migrates VMs without service disruption. It is configurable and fills the server with virtual machines one after the other. Eucalyptus chooses the server with the highest or lowest load and offers round robin mechanism for scheduling the VMs. Both VMWare and Eucalyptus do not offer any support for adding new schedulers in their system. Nimbus does not have its own scheduler, but is configurable with respect to scheduling and supports VM schedulers such as Oracle Grid Engine and Portable Batch system (PBS). In CloudStack, the VMs are allocated to host based on First fit, Round robin and Fill first algorithms. Besides, they provide support for adding new schedulers. The infrastructure stack in the project uses OpenStack's default scheduling policy that selects the server with lowest load for placing the VMs.

3.1.1.7 Billing and Accounting

Billing and accounting is an essential component as it helps the cloud service providers to charge the users for their services and resources consumed. In order to precisely charge the users for their consumption, a monitoring component is required that provides the utilization information of the virtual resources and applications. Pricing and billing models are defined which determines the cost of the resource usage. Besides billing, some common applications of accounting are process auditing, cost allocation and trend analysis.

OpenStack introduced a metering service Ceilometer in its Havana release [21]. This service provides information on the aggregated usage and performance data of the services deployed in an OpenStack cloud. In its earlier versions, Nova-Billing was implemented, a software billing system that stores state information about virtual resources (instance, volumes and storage) and provides information to users in the form of reports through its REST interface. This system also provides a web interface support, Horizon-billing, that can be enabled in the OpenStack dashboard. Apart from these two tools, there are few metering systems that were introduced, Efficient metering and Dough and Dash billing.

Eucalyptus[7] enterprise edition provides an accounting system "Reporting Overview" that provides information on the resources used in the cloud. The reports are generated for a specific time range and they are classified in to following types: virtual server's information, object and block storage usage, number of snapshots created and their size, number of public IP addresses in use and the overall consumption of the cloud resources.

OpenNebula[8] has an accounting toolset that addresses the accounting of the virtual resources. It provides the resource usage information obtained from the hypervisor and provides a platform for integration with chargeback and billing platforms. vCenter Chargeback Manager[9] is a reporting tool of VMware that interacts with the vCenter Database and calculates the cost for virtual environments by using the defined chargeback formula. The tool also consists of a Web based application for displaying the generated cost and usage reports.

[7] https://www.ecualyptus.com/docs/eucalyptus/3.4/admin-guide-3.4.2.pdf

[8] http://archives.opennebula.org/documentation:rel4.4:accounting

[9] https://www.vmware.com/pdf/cbm_users_guide_2_6.pdf

CloudStack[10] uses a billing system named "Fogpanel[11]" for displaying the billing / usage resource calculation, their reports and per user / project quota information. The system provides a portal using which the user can view necessary statistics and reports, invoice management, payment reminders, Hourly/monthly billing modes, etc. The previous version of CloudStack supported "Server usage[12]", a plugin that recorded metrics directly from the hypervisor and reported the resource consumption. The CloudStack also supports integration of third party billing solutions such as überSmith[22] and Amysta[23].

In EASI-CLOUDS, there are different pricing models defined for various use cases and they are classified into the three categories: usage based, flat rate and dynamic pricing. The usage-based model is computed using the metering information on the virtual hardware resources, uptime of virtual machines, image size saved in the image store, amount of storage used and the number of public IP addresses in use, etc. Flat rate models are those where the user is levied fixed charges for the amount of resources and services consumed for a period of time. Dynamic pricing is calculated based on the environmental parameters and SLA based models where price changes in accordance with SLA penalties. Detailed information about charging, billing and pricing models can be referred in section 3.5.

3.1.1.8 Interfaces

Most of the open source cloud software packages provide interfaces for the end user to create their virtual resources that are within the limits set by the administrators. In general, the interfaces are classified in to three types: Graphical User Interface (GUI), Command Line clients (CLI) and RESTful web services. Apart from the above, these software packages also implement quasi-standard interfaces for managing and monitoring the virtual environment. Common examples are Elastic Compute Cloud (EC2)[24], Simple Storage Service (S3) [14], Open Cloud Computing Interface (OCCI)[25] and Open Virtualization Format (OVF)[26].

Software	S3	EC2	OGF OCCI	vCloud
OpenStack (Icehouse)	Yes	Yes	Yes	No
OpenNebula	Yes	Yes	Yes	Yes[13]
Eucalyptus	Yes	Yes	Yes	No
VMware				Yes[14]
Nimbus	Yes	Yes	No	No
CloudStack	Yes	Yes	Yes	No

Table 3.6: Supported interfaces in IaaS platforms

EC2 and S3 are web services developed by Amazon, where the former allows the user to obtain, configure and control resources in its computing environment and the latter provides scalable interfaces for managing data using object storage architecture. OCCI aims at developing specifications and APIs for different cloud offerings using the REST (Representational State Transfer) approach for interacting with various resources offered as services. Initially, its primary focus was on the Infrastructure-as-a-Service (IaaS) but now the interface can be extended to support Platform and Software-as-a-Service (SaaS). OVF is a

[10] http://events.linuxfoundation.org/sites/events/files/slides/CCCNA14%20presentation.pdf
[11] http://www.fogpanel.com/cloudstack-billing-panel-features/
[12] http://24x7x0.wordpress.com/2012/01/24/usage-metering-and-charging-with-cloudstack/
[13] http://opennebula.org/opennebula-implements-vcloud-express-api/
[14] http://www.vmware.com/files/pdf/vcloud/VMware-vCAC-Whats-New-Technical-Whitepaper.pdf

standard that describes the format for the packaging of software, which will be run in the virtual machines. Table 3.6 below describes the standardized interfaces and their support in various cloud software packages.

It is evident from the Table 3.6 that OpenNebula implements most of the common Cloud interfaces. The vCloud service of OpenNebula enables it to launch and manage VMs through the vCloud APIs, which are implemented based on its cloud interfaces. The other software OpenStack, CloudStack and Eucalyptus can use VMware ESXi as the hypervisor.

Besides the REST approach, the above considered virtualization software offers a web interface that helps to realize on-demand-self-service in the cloud. It helps the administrators and users with a self-service portal to manage and control the physical and virtual hardware resources. Horizon[27], OpenStack dashboard that is developed using the Django framework of python, has a modular and flexible design which makes it easy to add new plug-ins and additional management tools.

Sunstone[15], GUI of OpenNebula, implemented in ruby, offers different views for various roles such as admin, user and cloud. These views are completely customizable. For instance, the user can enable or disable specific tabs or the controls present in each of the tabs. The UI details of other cloud packages namely CloudStack[16], VMware vSphere[17], Nimbus[18] and Eucalyptus[19] can be found in the reference section. There is also an add-on in Firefox, Hybridfox[20] that provides a unified interface for managing the virtual environments. The latest version of Hybridfox supports Eucalyptus, OpenNebula, OpenStack, Cloud Stack, HP Cloud and Cloud Bridge.

Every user in EASI-CLOUDS has a web account in Horizon that allows them to perform several operations such as instantiation of VMs using different resource configurations, upload images, associate IP addresses, create firewall rules, define ports and allocate block / object storage.

3.1.1.9 Monitoring

The monitoring component is responsible for measuring the Key Performance Indicators (KPIs) of the systems and services. In cloud systems, it provides the data primarily for the following areas: a) system monitoring b) Accounting, billing and auditing c) Service Level Agreements (SLAs). In system monitoring, it helps to diagnose hardware and software problems, to enhance the resource utilization and to ensure the system's performance and security. It also plays a key role for measuring services and for precisely charging the users based on their resources and services consumed. Sensors provide data for the monitoring component, and this includes information such as resource consumption of hardware and software and the Quality of Service (QoS). It is then forwarded to higher layers using web services or displayed to administrators using a GUI. The section below concentrates on monitoring only at the infrastructure level.

[15] http://docs.opennebula.org/4.4/administration/sunstone_gui/sunstone.html
[16] https://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.0.2/html/Installation_Guide/log-in.html
[17] http://pubs.vmware.com/vsphere-55/index.jsp#com.vmware.vsphere.install.doc/GUID-3FC8F86B-7F4A-450C-9D1F-0275E403F71C.html
[18] http://www.nimbusproject.org/doc/phantom/latest/webapp.html
[19] https://www.eucalyptus.com/docs/eucalyptus/4.0/index.html#console-guide/index.html
[20] https://code.google.com/p/hybridfox

In general, the cloud software has its own monitoring component. Alternatively, external monitoring tools, such as Nagios [28] and collectd[29] can be used using which monitoring information about the hardware and the services of the cloud software can be obtained from its respective plug-ins. Apart from the above data, cloud software also provides information on the total number of users, projects / tenants / accounts and the association between users / projects and virtual machines. Cloud management systems such as OpenStack use VMM (Libvirt) to provide information about virtual resources. This data can be obtained either via the cloud software itself or using external monitoring tools such as Nagios, Ganglia[30] or Collectd. However, the libvirt plugin for Nagios, nagios-virt[21] offers only the power state of the VM and cannot be used for the monitoring of virtual environments.

Ceilometer[21], the monitoring and metering component of OpenStack, provides data about virtual machines, number of images uploaded by users and their size, amount of block and object storage consumed and the amount of packets sent and received in the network interface. It was introduced only in Havana and there was no stable monitoring support in its previous versions. OpenStack is also compatible with monitoring tools such as Zenoss and Nagios. ZenPack, a Zenoss extension[31], allows monitoring of flavours, images and servers that are running in the OpenStack clouds. Similarly, Nagios also provides a plugin[22] using which OpenStack services can be defined, configured and monitored. Monitoring driver[23] is the component in OpenNebula that is responsible for collecting information about physical and virtual hardware resources. It executes a set of probes in the hosts and the information is transferred to higher-level components using events or interfaces.

Monitoring in VMware vSphere[24] is handled by several tools, which gather and display system information and resource usage. These tools can be accessed by either GUI or command line. Additionally, they support configuration of alarms, setting up alerts and notifications and the necessary actions to be performed when the threshold specific to a particular resource is breached. CloudWatch[25] is a service in Eucalyptus that collects monitoring information from the cloud resources, pre-processes them and converts them in to readable metrics. Furthermore, it provides options to configure alarms based on the generated events / data and allows publishing of new metrics in the CloudWatch system. By default, they monitor the following resources: instances, volumes (block storage), and load balancers.

CloudStack uses Usage Server[26], which creates a summary of usage records by taking data from the event logs. The interfaces on usage records accept user, project, start and end date as input and return information such as the VM run time, their resource utilization, number of public IP addresses belonging to the user and number of snapshots uploaded. Zenoss also provides an extension, ZenPack[27], for monitoring the software and hardware resources running under CloudStack. Phantom, the latest release of Nimbus, contains a package tcollector[28], which provides sensor-monitoring information about the deployed virtual

[21] http://people.redhat.com/rjones~/nagios-virt/

[22] https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/3/html/Installation_and_Configuration_Guide/Configure_OpenStack_Services.html

[23] http://archives.opennebula.org/documentation:rel4.4:devel-im#creating_a_new_im_driver

[24] http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-monitoring-performance-guide.pdf

[25] https://www.eucalyptus.com/docs/eucalyptus/4.0/index.html#user-guide/using_monitoring.html

[26] http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/latest/usage.html#usage-types

[27] http://wiki.zenoss.org/ZenPack:CloudStack

[28] http://opentsdb.net/docs/build/html/user_guide/utilities/tcollector.html

machines. It uses OpenTSDB (Time Series Database) for storing the gathered data that are received from the collectors[29]. Similar to other cloud software, tcollector also collects information about the virtual hardware resources such as processors, disks, networks, processes and the NFS storage.

The monitoring system in EASI-CLOUDS uses OpenStack along with Nagios for monitoring the physical and virtual servers and the applications deployed in the VMs. The data is gathered from these components and offered via RESTful interfaces to its subscribers. The section 3.4.2 describes several monitoring tools and provides reasons for choosing Nagios as the monitoring software.

3.1.1.10 Infrastructure and Software Stack at University of Mainz

Based on the findings from the previous sections, the following are the components part of the EASI-CLOUDS infrastructure software stack deployed at University of Mainz.

Cloud Software: OpenStack
Compute nodes: 25 Fujitsu servers; each of 32 cores (inclusive of hyper threading), 64 GB RAM and 250 GB disk
Hypervisor: KVM and libvirt
Image store: Glance
Size of the image store: 2 TB
Storage: Block storage (Cinder)
Size of the volume store: 8 TB
Networking: nova-network (flat networking mode)
Speed of the Ethernet switch: 1 GB / sec
Number of floating IP addresses: 120
Interfaces: a) GUI - Horizon dashboard, Hybridfox
b) CLI - OpenStack Nova client
c) HTTP support - OpenStack Nova REST Interface
Monitoring software: Nagios
Scheduling: OpenStack's default scheduling policy

3.1.2 Smart Placement

The problem of embedding virtual networks with different constraints in a substrate network is the main resource allocation challenge in network virtualization and is usually referred to as the Virtual Network Embedding (VNE) problem [32]. Smart placement in IaaS clouds is one of the VNE applications. In fact in IaaS clouds, the cloud customer request is modelled by a virtual network, where each virtual node is a set of specifications (operating system, computing power, storage capacity, etc.) defining the virtual machine to be deployed in the cloud, and a virtual link defines the physical links' requirements (bandwidth, delay, etc.) between two endpoints (virtual nodes). Therefore, the problem of placing the virtual requested graph in an IaaS cloud environment can be divided in two stages:

1) Partitioning the request into sub-requests and forwarding each sub-request to a corresponding cloud (inter-cloud placement)
2) Orchestrating different virtual elements belonging to a sub-request in the physical substrate (intra-cloud placement)

[29] http://www.nimbusproject.org/doc/phantom/latest/sensors.html

Since a cloud is a collection of heterogeneous physical servers managed by an IaaS cloud management software (OpenStack, OpenNebula, etc.), the efficiency of the system, in terms of energy, cost, survivability, quality of service (QoS), etc., depends on how virtual graphs requested by customers are distributed (or mapped) on different servers. Thus, IaaS cloud smart placement algorithms can be static or dynamic, coordinated and uncoordinated (mapping decomposition problem).

The majority of participants of a survey, which Rayport and Andrew conducted [33], indicated that their companies already used public clouds, or discussed, planned, trialled, or implemented the use of a cloud infrastructure. Their objective for using cloud technology has been the potential reduction of the company's data center costs. Even though there is a lot of mentioning about the benefits and even inevitability of migrating to a cloud [34] the exact costs are still unknown. This lack of cost details (and its accounting within a cost model) makes any decision on a migration to clouds uncertain [35]–[39].

Category	Reference	Contribution
S/C	Chowdhury et al. 2009	Coordination in VNE using multi-path for link mapping
	Butt et al. 2012	VNE awareness of substrates' bottleneck resources
	Papagianni et al. 2013	MIP (Mixed Integer Programming) with integer constraints relaxation. Rounding optimal solutions to obtain a final sub-optimal solution.
	Leivadeas et al. 2013	Graph partitioning inter-cloud VNE using a heuristic integrating a min k-cut algorithm followed by sub graph isomorphism
	Lischka and Karl 2009	Provides one stage VNE based on sub-graph isomorphism detection
	Tran et al. 2012	Exact solution using recursive approach. Minimizes the economical cost on client.
	Yin et al. 2012	Provides several optimizations on (Jens Lischka et al. 2009)
D/U	Cai et al. 2010	Reconfiguration based on physical substrates' evolution
D/C	Schaffrath et al. 2010	ILP (Integer Linear Programming) based VNE. Dynamically reconfigures existing mappings

Table 3.7: Classification of different IaaS clouds smart placement algorithms.

Many other statements about the need for cost models for clouds have also been made [40][36][37][41][38]. Using these cost models, one may investigate economic factors like Return-on-Investment (ROI), Net-Present-Value (NPV), Benefit-to-Cost-Ratio (BCR), and Discounted-Payback-Period (DPP). These measures are essential to decide when and under which conditions it is better to use clouds [36][37]. They enable the execution of a detailed cost-benefit analysis for determining whether running their services on the cloud is more cost effective than purchasing in-house resources.

However, to perform this cost analysis, detailed knowledge about applications, hardware, and load levels is required [42]. The difficulty of obtaining this knowledge makes it non-trivial to estimate short-term or long-term costs. As Ali et al. pointed out, it is difficult to know "the actual resources consumed by a system", "the deployment option used by a system, which can affect its costs as resources", and the "probable changes in the cloud service provider's pricing scheme" [41].

Even though knowing the estimated overall cost of services of an enterprise is essential, a further step towards finding the optimal allocation of services in the federated hybrid cloud is required. This is necessary as different service placements may incur different costs. For example, the different cost might come from different cloud provider prices, geographical differences in electricity prices, or from different fees for Internet connectivity. Besides, the rate of data transfer between services might be different as different types of applications interconnect. Therefore, for example, keeping services with high traffic within one cloud provider might decrease the traffic cost. To predict the traffic rate Koch et al. proposed a "workload-aware method of provisioning", which is effective for the case of educational institution [43]. If the workload characteristics and parameters of the domain are known, then an accurate prediction of future traffic among services is possible. In the general case, such an optimization model is not possible.

Hwang et al. proposed a cost optimization model for service provisioning, considering two types of pricing plans, namely on-demand server pricing and reserve-server pricing [44]. The objective of this work has been to support cloud providers in lowering their service provisioning cost. Consequently, the authors have not addressed hybrid clouds nor federated clouds.

Another approach is the one of Kessaci et al. [45]. They optimize the scheduling of tasks over a geographically distributed data centers. Their algorithm considers three objectives for the optimization, namely energy consumption, CO2 emission, and profit.

Bjorkqvist et al. analyse the total cost and performance of running services on hybrid clouds [46], using an earlier version of the cost model of Kashef and Altmann [47]. Their focus is on a cost-performance framework for allocating services to clouds, considering the performance-cost trade-off between nodes of private clouds and nodes of public clouds.

Tran and Agoulmine proposed an algorithm for service placement that considers the network topology, resource availability, and customer demand [48]. Therefore, Tran and Agoulmine's solution can deal with changes in the network environment. In addition to this, the efficiency of the new locations for the services versus the necessary modifications that have to be made to obtain them are compared.

Bittencourt and Madeira propose an algorithm for hybrid cloud customers to decide which of the services should run on the public cloud and which on the private cloud [49]. Their algorithm considers budget aspects and service requirements.

Table 3.7 presents the classification of some IaaS clouds smart placement algorithms presented in literature.

3.2 Platform-as-a-Service

3.2.1 PaaS Frameworks

Platform as a Service (PaaS) is the level of cloud computing, where the cloud service provider exposes an interface that the developers can use to send their own programs that will then be exposed to the Internet. On the other side of such an interface is a collection of software called a PaaS framework.

Currently application developers have a number of platforms to choose from. These include CloudFoundry, Engine Yard, Google App Engine, Heroku, OpenShift and Windows Azure. The management of the platform is out-sourced to a third party, in the case of public cloud, or another in-house department, in the case of private cloud.

Moving applications between different cloud offerings and between the public and the private cloud often requires modifying the program code. Such porting efforts may be costly, but may still be a viable option when compared to doing a full rewrite of the application. From the application developer's perspective it is desirable to write the application once and then run the same application in various compatible public and private cloud offerings.

To support interoperability, it is required that several platforms implement the same APIs. With multiple implementations, it may eventually be possible to build a standard that guarantees interoperability between the public and the private cloud, and between different cloud providers. In practice, it is required for the APIs to be well defined, which leaves out platforms that let the user select arbitrary software components to be part of the cloud offering.

The programming model for Google App Engine simplifies development of scalable application by restricting the set of features that are available to the application; by removing access to features that would hinder scalability. Restricted features include accessing the local file system of the application host, directing programmers to use a (distributed) database instead, as well as execution time restrictions that prevent a single computation from affecting performance of other computations. We have surveyed the relevant frameworks that implement the Google App Engine API and discuss them in the next section.

3.2.1.1 Google App Engine

Google App Engine (GAE)[50] is the original App Engine implementation by Google. It supports programs written in Java, Python, Go and PHP. The Java support can enable a number of3.2.1.23.2.1.3 other languages, for example Ruby with JRuby [51] and Clojure [52].

For different kinds of storage needs, GAE provides Cloud SQL, a traditional SQL database based on MySQL, a schemaless database called NoSQL and an object store called Cloud Storage. While interactive tasks with the applications are intended to remain short, GAE provides Task Queue for performing longer tasks in the background.

The GAE SDK contains command line tools for deploying applications and graphical deployment tools are available through an Eclipse plugin. Deployment is also possible using the revision control tool Git.

A rich management console is available, that contains information about the apps' usage, quotas, and billing. It also provides the developers with configuration tools, application version management tools, logging capabilities and debugging tools.

Google charges its App Engine customers based on how much load and traffic they produce, and how much storage they consume. It is therefore possible to start cheap and scale up rapidly if the service gets popular. Some simple applications may even be able to run within the free usage quotas given that the amount of users remains small [53]. Unlike AppScale and Cape Dwarf, Google App Engine is offered solely as a hosted service.

3.2.1.4 AppScale

AppScale [54] is an open source App Engine framework that aims for full compatibility with Google App Engine, but is available for private deployments. It consists of a collection of open source programs that work together to form the AppScale platform. The project started at the RACElabs, Department of Computer Science in University of California, Santa Barbara. AppScale Systems, Inc. was formed in November 2012. AppScale is supported by Google and is one of Google's Cloud Technology Partners [55]. The development is active and new versions come out at regular intervals.

Most of the AppScale code is written in Python, with some parts written in Ruby. The open source components include such programs as Apache Zookeeper, which is used for synchronization and distributed coordination, RabbitMQ, which is used for Task Queue, and ejabberd and Strophe.js to implement XMPP and Channel. Supported databases include Cassandra and Hybertable but AppScale does allow manual addition of other databases. Deployment and management of applications and users can be done through AppScale Dashboard, or through a command line interface.

For private clusters AppScale provides tools for the administrators to specify how they want to deploy the platform. During the deployment, administrators can specify different roles for the nodes such as App Engine, database, login, master or zookeeper [56]. Specifying a role can help routing traffic in the cluster to increase fault tolerance and performance.

3.2.1.5 CapeDwarf

The CapeDwarf [57] project was created by Aleš Justin, Marko Lukša, and Matej Lazar at Red Hat [58]. The goal of the CapeDwarf project is to provide a way to deploy existing Java Google App Engine applications on JBoss' WildFly Application Server (JavaBeans Open Source Software Application Server) without any modifications. This makes it possible to deploy App Engine applications on cloud platforms as long as they can run the WildFly server. For example, the RedHat OpenShift platform supports WildFly and can thus run App Engine applications with CapeDwarf [59].

CapeDwarf is implemented as an extension to WildFly, using the WildFly APIs. These WildFly APIs include Infinispan, HornetQ, JGroups and several others. This way, CapeDwarf can be deployed to a WildFly server as a module that handles GAE applications.

The aim of CapeDwarf is to be fully compatible with the GAE APIs. Currently, CapeDwarf supports many of the APIs used by GAE, although here are still some API implementation in CapeDwarf, which are just placeholders, and further implementation is needed. The

incomplete APIs include appIdentity, capabilities, channel, mail, OAuth and XMPP APIs [60]. The current beta version of the CapeDwarf supports GAE API version 1.9.11; applications compatible with this API version should function properly.

The CapeDwarf applications can be deployed to the WildFly server in several ways. The WildFly administration console can be used to deploy the application or the application package can be placed into a directory, where the WildFly server looks for applications for automatic deployment. If the RedHat OpenShift cloud is used, CapeDwarf applications are deployed by creating an OpenShift gear with a WildFly cartridge and then using git to insert the CapeDwarf modules and the application to the running OpenShift gear.

3.2.2 Software Engineering and Application Development Using MIDEaaS

MIDEaaS (Mobile IDE as a Service) tries to give cloud-based (web-based) IDE as a Service to programmers to use instead of desktop-based ones like Eclipse or Visual Studio.

3.2.2.1 MIDEaaS Architecture

MIDEaaS consists of 3 main parts: MIDEaaS-App, MIDEaaS-IDE and MIDEaaS- Editor. In Figure 3.1, high-level architecture of MIDEaaS is depicted. You can think of each rectangular as a separate Jar file, which uses (imports in Java language) its connected Jar file. The core component is MIDEaaS Editor. In this document, I feel describing MIDEaaS Editor will give enough information about MIDEaaS.

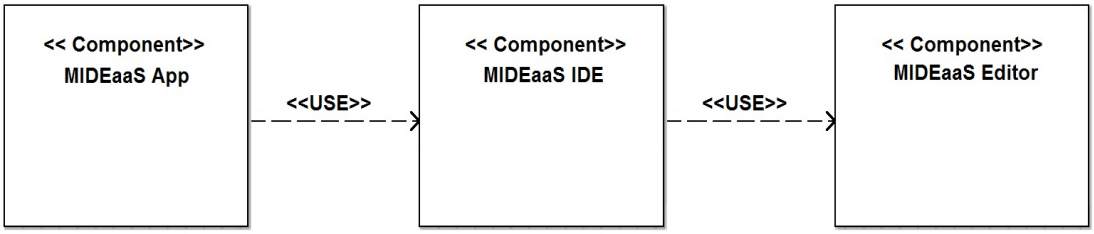

Figure 3.1: MIDEaaS Main Components

3.2.2.2 MIDEaaS Editor

The editor which is selected to be used in MIDEaaS is CoRED[61], Collaborative Real-time editor. The basic tools for designing CoRED are Vaadin Framework, Java Developer Kit (JDK) and Ace editor [61].

Vaadin framework is responsible for HTTP(S) communication between client and server side. Vaadin is also used for making CoRED a Vaadin component. Therefore, CoRED can be used as a part of any Vaadin application or individually as it is. On server side JDK analyses source code and on the client side Ace editor plays the role of fronted to interact with clients [61].

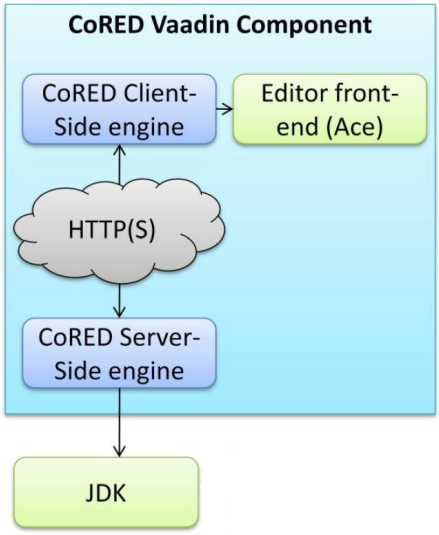

Figure 3.2: CoRED architecture

3.2.2.3 CoRED Architecture

Let's first have a look at Figure 3.3. As said before, CoRED is a Vaadin component. In order to take care of client- server communication in CoRED, Vaadin needs the server side component, which is CollaborativeCodeEditor, and client side widget, which is VCollaborativeCodeEditor. As the nature of Vaadin requires, most of computation (like error checking and code suggestion) should be performed on the server side and client side is just for showing UI to user (either by rendering UI to JavaScript using GWT or directly writing JavaScript or combination of them), interacting with user and sending those interactions to server side [62]. For example when user writes a line of code semantically wrong, source code is compiled on server side and the result will be shown on client side editor to inform user about their mistakes.

In CoRED, the client side widget is actually a wrapper for a third-party code editor. This is the power of Vaadin that enables us to use the combination of Java and JavaScript to write client side widgets of Vaadin components. The developers of CoRED decided to use Ace editor for its strong supports of indentation, syntax highlighting and customizable marks [61]. Moreover, CoRED architecture is designed to make it flexible for further development. For example error checking and code suggestion functionalities of CoRED are both extendable and replaceable [61]. The CoRED architecture is depicted in Figure 3.2.

3.2.2.4 Code Suggestion

While writing code with CoRED, a user can trigger code suggestion functionality by either using a special combination of keys or typing dot after objects or interfaces.

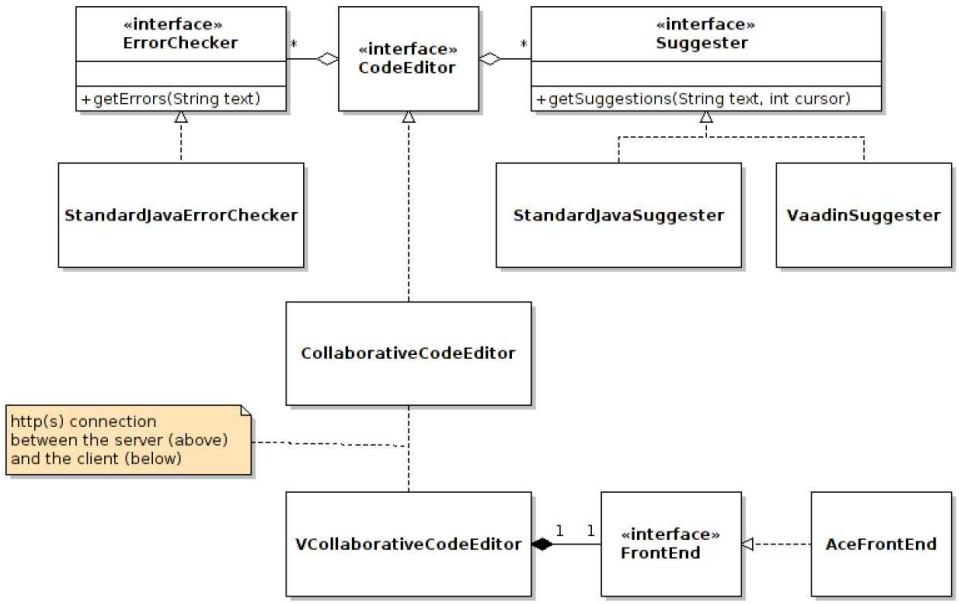

Figure 3.3: CoRED high level class architecture

CoRED can suggest code completion of Java language and Vaadin frameworks. Two implementations for Suggester Interface are given in Figure 3.3. There are two main scopes where suggestions could be from:

- Code written by developer: In this case, JDK (by the help of its ParserPath- Scanner) scans through all the program source code and builds a tree of classes, methods, variables and all other Java types.
- Code coming from imported packages: The solution is loading classes and using reflection[63] to extract public methods and variables.

3.2.2.5 Error Checking

Before jumping into any design for error checking, two important questions should be answered: how and when error checking must be done [61].

- Error checking is performed by the JDK. On server side, Java compiler of JDK compiles the source code and returns error diagnostics. These error diagnostics have all the necessary information like error messages, with line and column numbers. This information will be sent to the client to be shown to user in the editor. (see Figure **3.4**)
- Because compiling is a resource consuming process, it should be done in a trade-off manner. Practically, the compilation is started when the user stops modifying source code and waits for a while before starting again.

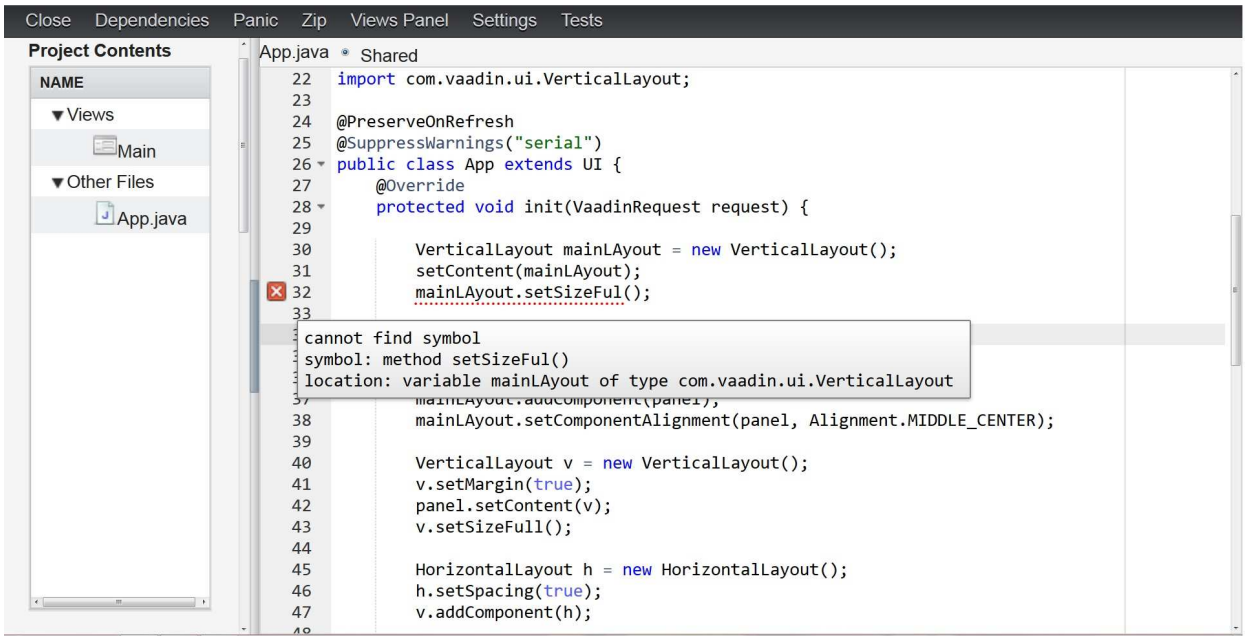

Figure 3.4: Error line and marker in MIDEaaS

3.2.2.6 Related Works

There are a wide range of web based IDEs released in the past few years. They can be analysed from different points.

Client-side and Server-side Development

Some IDEs have their focus on client side development, including js-Fiddle that supports JavaScript, HTML and CSS. Some IDEs also enable server side development. One of the most common ways of providing server side development is Node.js in which JavaScript is written on server side. IDEs like Cloud9 and AkShell supports Node.js.

In our case, MIDEaaS is going to do most of computation stuff (hard job) in cloud and client is just for communicating with developer. Therefore, MIDEaaS resides in the second category, server side development.

Range of Languages and Solutions

IDEs like Cloud9 and Codiad supports a wide range of languages like Node.js, PHP and so on. Moreover, some IDEs like eXo Cloud IDE supports different solutions like Java Spring Framework and Rest services [64].

The philosophy of MIDEaaS is different from above mentioned IDEs. MIDEaaS's mission is to develop and deploy Vaadin based web applications. Choosing only one language and limiting the era of software development enables us to provide more complete and specific tools than a general use IDE. As a proof, MIDEaaS provides graphical editor for generating UI in which developer could drag and drop previously made component and integrated them into source code.

We compare MIDEaaS to two popular web-based IDEs (Cloud9 and Nitrous.io) in the Table 3.8. There are lots of features that can be compared, but we have chosen the most important ones.

IDE	Collabo-ration	Real Time	Version Control	Code Completion	Deploy	Real-time Error Checking	Code Folding	Command Line
MIDEaaS	yes	Yes	Yes (MGit Plugin)	Yes	Yes	Yes	Yes	No
Cloud9	Yes	Yes	Yes (Command Line)	Yes	Yes	No	Yes	Yes
Nitrous.io	Yes	Yes	Yes (Command Line)	Yes	Yes	No	Yes	Yes
	Languages							
MIDEaaS	Vaadin, Java							
Cloud9	Node.js, HTML5, WordPress, PHP, Python/Django, Ruby/Ruby on Rails, C/C++, StrongLoop, Others							
Nitrous.io	Node.js, PHP, Python/Django, Ruby/Ruby on Rails, Go							

Table 3.8: Comparison of on-line IDEs

3.2.3 Cloud Computing and User-centered Design

User-centered design is a method used in designing new interactive products and services. It a process that helps designers to define end user's needs and limitations that will serve as a guideline all the way through the design process. Users are being analysed by building up personas, scenarios and use case stories that help designers to understand who are the key users of the service, how they would like to use the service and what would be the most effective and user friendly way to use it. Besides end users it is also important to recognize relevant stakeholders and their roles in the context of the service design. Recognizing all relevant stakeholders and end-users of the product helps also to prevent challenges and risks.

In user-centered design users are being involved the all the way through the design process. Interviews, focus groups, observation, and user testing are typical methods for gathering the user information. Design that is made to fit the needs of an end user typically reduces time and costs from the product development. Typically also the learning curve of the products that are being designed involving the real users is shorter.

It is quite common to understand usability as an essential part of user-centered design process. Usability means ease of use and learnability of the service. By enhancing usability designers make sure that the service is more efficient to use, easier to learn and more satisfying to use.

In cloud computing typical user and stakeholder roles can for example be end users, software developers and system administrators. Software as a Service (SaaS) offerings are provided straight to end-users. Their user interfaces are typically thin clients like web browser. Software developers are being offered Platform as a Service (PaaS) offers with different kinds of storage and distributed computing capabilities. System administrators are being offered Infrastructures as a Service (IaaS) offers that are usually iterations of existing hosting services.

When designing a new cloud computing service by user-centered design process all stakeholders and end users would be analysed to find out what kind of groups they would form based on their needs, limitations and expectations. It is likely that different user profiles are using the same product to do different kind of tasks and by clarifying their needs it is possible to target the best service for each of them.

In cloud computing there are for example some services that provided different cloud service levels. This means that they are categorized and targeted to different user profiles. Interfaces that allows user to make customizations based on their needs are also good examples of highly motivating web services. Self-service portal model is another example of typically web based services that are user friendly, efficient and cost reducing for the service provider.

Key elements on designing a user-friendly product are to boost user's motivation in to design services that are motivating, usable and efficient. Following guidelines of user-centered designs can ensure this.

3.3 Brokerage and Federation

3.3.1 Cloud Brokerage

Cloud federation and inter cloud aim at the same objective: the interoperability of cloud services. However, they have very different styles:

1) The cloud federation is unifying, and gathers under the same governance of voluntary service providers to join
2) Inter-cloud is "globalist" and brings together the world on same principles, protocols, etc. process.

In both cases, technical broker technologies are necessary but either in a federated architecture or in a globalized architecture.

Cloud Brokering is a service paradigm that provides interoperability and portability of application across multiple Cloud providers. A broker provides a single interface through which you can manage multiple clouds (each exhibiting its own interface, pricing model and value-added services) and share resources across clouds. None of the involved entities (cloud service provider / cloud service consumer / cloud service broker) has a complete control over actions of the others. Brokers intermediates, rather than control, in coordinating inputs and outputs of multiples services. The "Broker" term highlights the indirect nature of the business model it provides. This introduces a new way of doing things and new needs such as risk of failure (detection and reaction), risk of service quality degradation (control, monitoring and reaction), liability between providers and consumers (Transparency & Auditability).

3.3.1.1 Different Views on Cloud Service Brokerage

The NIST Cloud Computing Security Reference Architecture [65], derived from the NIST Cloud Computing Reference Architecture [66], enhances the description of the roles and types of services that a cloud Broker may offer to cloud Consumers. A cloud Broker renders some combination of services that can be divided into five Architectural Component categories: Secure Service Aggregation, Secure Service Arbitrage, Secure Service Intermediation, Secure Cloud Service Management, and Secure Cloud Ecosystem Orchestration.

According to Gartner [67], CSB is a role of intermediary, in which a company or other entity adds value to one or more (generally public or hybrid, but possibly private) cloud services on behalf of one or more consumers of those services. Cloud-enabled technology services are a prominent aspect of the cloud services supplied by a CSB. The CSB offering will also often

include some combination of capabilities that fall under three primary roles: Aggregation brokerage, Integration brokerage, and Customization brokerage. Gartner's Intermediation encompasses these 3 primary roles.

According to Forrester an intermediary has to offer a certain complex "combined" value proposition in order to qualify as broker. Forrester also distinguishes three types of cloud brokers, according to the level of the cloud stack at which they operate [68]:

- Simple cloud broker (dynamic sourcing within one cloud segment, such as public cloud IaaS)
- Full infrastructure broker (dynamic sourcing across public, virtual private, and private clouds)
- SaaS broker (unified provisioning, billing, and contract management with multiple SaaS offerings, potentially including integration of services).

3.3.1.2 Research on Cloud Brokering

In "A comparison Framework and Review of service Brokerage Solutions for Cloud Architectures" [69], Open Source Service brokerage solutions are compared, according concerns like:

- System category and type,
- Core Capabilities, core features and advanced features,
- Architecture & Interoperability,
- Service Languages, Programming Model and Service Engineering,
- Quality: Scalability / Elasticity and SLA's

Authors place emphasis on

- The emergence of cloud broker solutions on top of cloud management.
- The need for further separation of marketplaces and cloud broker solutions
- Service description mechanisms to commoditize the cloud:
    - To abstract, manipulate and compose cloud service offerings
    - To serve as starting point in federated clouds

Broker@Cloud Project: Deliverable D2.1 - "State of the art and research baseline"[70] proposes a Taxonomy of Cloud Services Brokerage Capabilities based on two orthogonal dimension of clouds brokerage space: Cloud Service Type (SaaS/PaaS/IaaS) and Cloud Brokerage Capabilities (Discovery / Integration / Aggregation / Customization / Quality Assurance / Optimization) and classifies 30 current providers and enablers of Cloud Service Brokerage Capabilities.

This analysis shows that the majority of CSB service providers or enablers appear to focus on Discovery, Integration, Aggregation and Customization with a particular emphasis on SaaS services. For both kinds of offerings, PaaS is the least supported type of cloud services. IaaS appears to be the most commoditized category of cloud services today. Coverage of quality assurance and optimization capabilities is sparser.

The state of the art of enabling technologies for continuous quality assurance and optimization capabilities in cloud service brokers covers the following research areas:

- Methods and Tools for Cloud Service Description.
- Methods and Mechanisms for Cloud Service Governance and Quality Control
- Methods and Mechanisms for Cloud service Failure Prevention and Recovery
- Methods and Mechanisms for Optimization of Cloud Services.

F. Diaz-Sanchez, in *Cloud brokering: new value-added services and pricing models* (June 2014), proposes new value-added services and pricing models in Cloud brokering at the infrastructure level. The problem of a single figure of merit of VM Cloud performance and the problem of VM placement in cloud brokering are addressed, and a new pricing model for cloud computing known as pay-as-you-book is proposed.

The author outlines that the description of the computation of a single figure of merit of VM Cloud performance is a multi-criteria problem (Communication, Computation, Memory, Storage, Availability, Reliability, Scalability and Variability). The weight of these criteria in the computation of a figure of merit of Cloud performance depends on the application profile foreseen to run on top of the Cloud infrastructure. The Analytic Hierarchy Process (AHP) technique has been used to analyse and to solve the Multiple Criteria Decision Making (MCDM) problem of finding a single figure of merit of Cloud performance. In this case, AHP enables an objective determination of the relative merit of the VM performance criteria for a given set of Cloud providers.

Similarly, the problem of placement in Cloud brokering is described as a multi-criteria problem This problem refers to the efficient distribution of Cloud infrastructure across multiple and non-interoperable Cloud providers. Pre-emptive goal programming has been used to tackle this problem by defining a set of multiple LPs (Linear Programming) with different priorities assigned by the end-user.

A pricing model between pay-as-you-go and subscription-based known as pay-as-you-book is proposed. Contrary to subscription-based pricing models, pay-as-you-book allows reservations of Cloud resources for future use without long-term commitment. Three resource allocation policies to manage the extra-time required by running reservations under pay-as-you-book have been described and evaluated. Among the evaluated policies, the economic agent maximizes Cloud provider's revenue while keeping an acceptable ratio of resource utilization.

3.3.2 Cloud Federation

Although there are many definitions of cloud computing, the NIST definition seems to have captured the commonly agreed cloud computing aspects that are mentioned in most of the academic papers [33]. The NIST definition states that cloud computing is "A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [33].

Furthermore, we assume that a cloud provider can own several clouds [71]. Each of these clouds is assumed to be at different geographical locations (e.g., different regions of Amazon AWS are considered to be different clouds) and could use the same cloud standard. Besides,

from an economic perspective, there is no difference between a cloud provider owning the data center and a cloud provider that rents hardware from a data center provider [71]. The only difference from the perspective of a hardware-renting cloud provider is that this provider outsources the maintenance of the data center hardware. The cloud provider's profit comes from the difference in selling cloud services on demand and the fixed renting cost paid to the data center provider. The data center provider benefits from a 100% utilization of their hardware services and does not need expertise on cloud computing.

To clarify the differences between public cloud, private cloud, hybrid cloud, and federated clouds, we categorize these clouds with respect to the ownership of the clouds, the number of clouds involved, and the number of cloud standards used in Table 3.9.

<table>
<tr><td colspan="3" rowspan="2"></td><td colspan="3">Cloud Ownership</td></tr>
<tr><td>Clouds are Owned by the Same Provider</td><td>Clouds are Owned by Different Providers</td><td>One Cloud is Owned by the Cloud Customer and the Remaining Clouds are Owned by Different Providers</td></tr>
<tr><td rowspan="3">Number of Clouds</td><td>One Cloud</td><td>One Cloud Standard is Used</td><td colspan="2">Category 1:
Public Clouds</td><td>Category 2:
Private Clouds</td></tr>
<tr><td rowspan="2">Two or More Clouds</td><td>Different Cloud Standards are Used</td><td colspan="2">Category 6:
Differentiated Clouds
(Clouds are not interoperable)</td><td>Category 3:
Hybrid Clouds
(Private cloud can access public clouds using their cloud standards)</td></tr>
<tr><td>One Cloud Standard is Used</td><td>Category 7:
Distributed Clouds</td><td>Category 4:
Federated Clouds
(Standard used is based on agreement)</td><td>Category 5:
Federated Hybrid Clouds</td></tr>
</table>

Table 3.9: Cloud categories

The first category (public clouds) describes clouds, which cloud customers do not own but use to fulfil their computing service needs. The cloud provider uses a cloud infrastructure standard, which can be proprietary. The second category (private clouds) defines the company's private data center as a private cloud. The cloud customer, who owns the data center, uses the data center with cloud computing technology to meet all of its computational needs. The third category (hybrid clouds) describes interconnected clouds, in which the cloud customer owns one of the clouds and the second cloud is a public cloud. This category defines a combination of a private cloud and a public cloud. The fourth category (federated clouds) represents public clouds that use the same cloud infrastructure standard. Therefore, VMs can easily be migrated between the federated clouds owned by different cloud providers. A cloud federation requires, at least, an agreement between cloud providers to commit to a specific cloud infrastructure standard. The fifth category (federated hybrid clouds) represents clouds that are the focus of this paper. A cloud customer, who uses federated hybrid clouds, runs some of its services on its private cloud and some others on a federation of public clouds (federated clouds). The sixth category (differentiated clouds) represents the overall cloud market, in which cloud providers offer their services using different standards [72]. Consequently, the different public clouds are not interoperable. A cloud customer, who wants to use several public clouds, would need to use the standards of each cloud. The seventh category comprises clouds, which use the same cloud standard and are owned by the same provider. Those clouds are located at different geographical locations (e.g., Amazon AWS).

We focus here on two cloud properties, namely, hybrid clouds and federated clouds. The definition of hybrid clouds used follows the one of Metzler and Taylor [73] as well as Van den Bossche et al. [74], in which organizations use public clouds to cover their demand for computational resources in excess of the capacity of their private cloud.

With respect to federated clouds, technology is needed to combine disparate public clouds, including those owned by different organizations. Only through federation (including its interoperability requirement) can a single cloud provider take advantage of the aggregated capabilities to provide a seemingly infinite service computing utility. We refer to this category of clouds as federated clouds [34]. In detail, federated clouds comprise clouds of (competing) cloud providers, who have reached a cross-site agreement for cooperating regarding the deployment of service components (e.g., through a marketplace of standardized goods [72], [75]. The concept is similar to electrical power providers, who use capacity from each other to cope with demand variations among their own customers [76].

Figure 3.5 illustrates the above definition of cloud federation by showing an example of a cloud customer (company) that is in need of computational resources. This cloud customer runs its private cloud (i.e., its own data center with cloud technology) to host its security-critical services. Beside the private cloud, the company uses two different clouds (i.e., public cloud 1 and public cloud 2) for services that are needed at times of peak demand. The arrows represent the communication between the services. The two public clouds offer the same cloud interfaces to each other and the cloud customer, following the cloud federation agreement between the two public clouds.

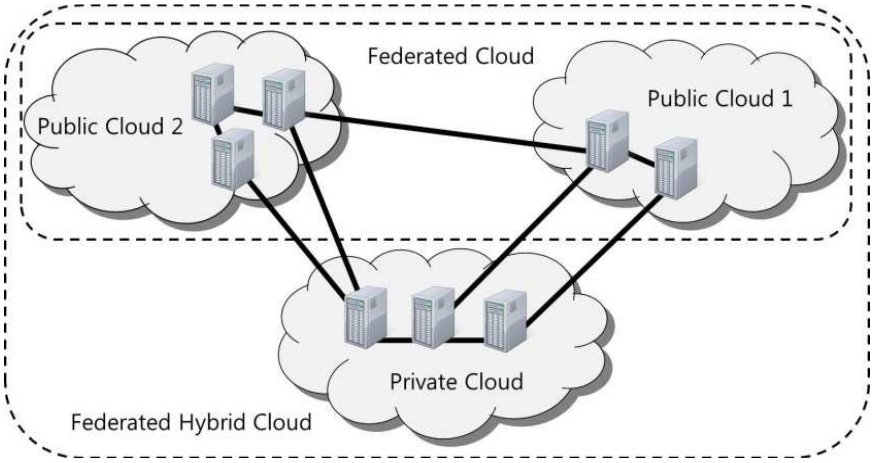

Figure 3.5: Schematic view of a federated hybrid clouds

The key to a Cloud federation is a common interface through which Cloud providers access each other [77]. With the common interface, a third party agent involved in a Cloud platform can utilize the computing resources of another Cloud provider without changing any business relationship and technical set up. By doing so a federation gives benefits not only to Cloud providers but also to third party agents. From Cloud providers' point of view, they do not just share their computing resources through a common interface, but their clients and service providers in the ecosystem do share resources without investing further taxes on their systems or changing their business models. The extension of computing resources and users enhance their economies of scale by moving the overload of a Cloud provider to a third party with ease.

From the third party agents' point of view, they gain the guarantee of the quality of service without signing additional contracts with multiple cloud providers. The third party agents do not even recognize their request to its Cloud tossed to another Cloud provider because sharing and migrations are achieved automatically.

Like other businesses in the ecosystem on a monopolistic Cloud, clients utilize the platform provided by a Cloud federation without any payment, and buy the services developed on the basis of the platform at reasonable prices. Service providers earn benefits from the clients gathered on the platform of a cloud federation while outsourcing the computing for their services. However, the business in the side of Cloud providers changes. A Cloud provider gains a benefit from the federation, i.e. economies of scale, by sharing its computing resources and Cloud users with another Cloud providers. But, notably, this requires costs for collaborating with each other to form a federation, i.e. investing on the development of a common interface, making contracts with partners on technical connection and benefits shares etc. Therefore, establishing a Cloud federation requires a strategic decision of Cloud providers according to the benefits and costs.

3.4 Management, Monitoring, Configuration and Post-configuration

3.4.1 Application Management

Application management in the cloud differs from its traditional counterpart significantly. First of all in a cloud there are different types of "applications" – or better – services: First of all there are services, which are designed for the cloud. They require the cloud infrastructure, the scalability and the other cloud services directly and are tightly integrated in the cloud. Another type of cloud applications can be seen as standard web-applications, which are already designed for being used over the Internet. In the cloud they are hosted by virtual machines in the same way as in a standard server environment. On the end of this spectrum there are standard desktop applications that have no notion of the Internet or a cloud at all. They require the greatest effort to make them "cloud"-manageable.

Application management covers the whole lifecycle of a cloud service. In the following paragraph a detailed look into the major phases is given.

3.4.1.1 Service Creation and Development

A cloud service can roughly be divided into three different types denoting the depth of integration into the cloud ecosystem:

1. Full cloud-aware service: The service is developed with respect of the cloud features and services. It uses the cloud APIs to access specific resources and trigger events. The service directly encodes the logic to drive SLA-monitoring and billing functionalities. Eventually it has to cooperate with the identity management and security services of the cloud, too. Because cloud services are delivered over the Internet, such services have a cloud-compatible user-interface or support web-compatible protocols for communication.

2. Cloud-based services: This kind of services uses some of the cloud-services, like a scalable database or an eco-system like Hadoop. This means that all the cloud related features is inherited by the used or integrated services, but does not affect the program code as such.
3. Third party cloud services: A third party cloud service can be an application that is installed in a virtualized environment, but has no aware, that it is running in a cloud. In EASI-CLOUDS the FreeSurfer[30] software for processing and analysing human brain MRI images is such an example. The real task in service creation is here to integrate this software and enable it as cloud service. This means in the integration layer the original interfaces have to be turned into cloud interface. For example the shell-scripts and command-line options have to be turned into a REST-interface. An interface for monitoring the application has to be built and suitable control functions have to be established. With this approach a cloud-awareness wrapper serves as integrator for the tool into the cloud.

The result of this stage is the service implementation, the service manifest describing how the service is to be instantiated and its dependencies, pricing models and information allowing the cloud and service to offer the service with a set of options for the service-level agreement with the consumer of the service.

3.4.1.2 Service Provisioning

Service provisioning is the phase describing the task of integrating a new cloud service in the offering of a cloud provider. This is the preparatory step of creating the service template and providing all resources for the service instantiation. In the EASI-CLOUDS environment the service provisioning contains the recipe how to create a running instance of the service on request in the cloud and the definition of all resources needed for the service runtime. Additionally all information pertaining communication with the cloud services like SLA-management, monitoring, and billing has to be specified because at the instantiation phase all these communication links inside the clouds will have to be created. In a typical IaaS-based cloud all these steps have to be done manually. Also the services have to be "fine-tuned" to communicate through the given SLA-management APIs and the SLA-management has to be made aware how to handle a new installed cloud service. The same applies to billing and of course the service catalogue where the end-user can find and request the new service. A promising approach is to use a service specification. Linked USDL[31] as *Unified Service Description Language* is a good candidate and can handle also complex information like pricing models for services and is currently used in the EASI-CLOUDS project. In order to do be able to do the instantiation of the new cloud service a suitable description has to be provided which allows creating the needed virtual resources in the cloud. The ACCORDS[32] platform is such a component, which works on manifests describing resources and scripts necessary for the configuration. The platform maps these specifications into the native cloud instantiation calls (for instance for the OpenStack cloud).

[30] http://surfer.nmr.mgh.harvard.edu/
[31] http://www.linked-usdl.org/
[32] http://www.compatibleone.org

3.4.1.3 Service Instantiation

The instantiation sets of the resources for the service and creates a runnable instance with respect of the defined recipe. There are different approaches possible in the cloud: With a full template –e.g. preconfigured virtual machines -- it is copied, parameterized and can then be started. Another option is the use of software configuration tools like puppet[33], which installs the service after the virtual environment has been created with all dependencies from a repository. This has advantages if updates or changes need be effectively managed in a modular environment.

3.4.1.4 Service Monitoring

An important part of the service operation in the cloud is the monitoring of the service performance. This data is necessary to control if the SLA with the service user can be met or will be violated. The monitoring can be viewed from two perspectives:

- The resource monitoring records the use of resources, like CPU, storage, or network. As the resources are essential for the service function, they need to be monitored closely and a lot of environments exist and are already built in into the cloud, as they are also crucial for the cloud operation as such.
- The application/service monitoring allows tracking the performance of a service. This data is correlated to the semantics of a service. For instance if a service does transactional processing the number and the duration of transactions is such a monitored parameter. Also this monitoring can be used for billing purposes if the pricing model is defined on the number of transactions and not on the computing resources used. This monitoring information is the foundation for advanced SLA monitoring of services in a cloud. Depending on the SLA to be upheld and SLA manager may automatically increase resources for the service if the performance falls below a critical threshold, or conversely reduce allocated resources if they can be utilized for other services without compromising the actual one. This approach also can enable automatic up- and downscaling of a service by creating or removing new service instances.

3.4.1.5 Service Termination

In the service lifecycle the termination denotes the point when the service instance is removed from the cloud. In case of a user this specific service for no longer available with the previous instantiation parameters. However, the service itself is still available in the cloud for creating new instances. Service termination can be implemented as the destruction of the virtual resource, which is desirable in some scenarios when data security is of paramount importance, so every time the service is needed again it starts from scratch with new data. If a service is reusable there may be implementations where the termination from the point of a user means making it unavailable for the user and severing all links between user, monitoring, and billing of this service, thus making it available for other cloud-users, of course this entails a very careful design of the service with regard to data protection.

[33] http://puppetlabs.com/

3.4.1.6 Service Decommission

The service decommission is the final stage in the lifecycle of a cloud application/service: All service artefacts as defined in the service-provisioning step are removed from the cloud. With the exception of still running instances in the cloud no new ones can be created.

3.4.2 Infrastructure Management

The monitoring in cloud environment needs to be accurate and performed at a fine-grained level. As described in section 3.1.1.9, it is essential for measuring the KPIs of systems, applications and services. In this section, we concentrate on infrastructure level monitoring. There is a wide range of tools available that supplies monitoring information and we evaluated a few of them, which are listed below: libvirt, Nagios, Collectd, OpenStack monitoring solutions and SIGAR. We further give reasons for selecting Nagios as the monitoring software in the project.

3.4.2.1 Libvirt

Libvirt[78] is a toolkit written in C that provides a uniform interface for different virtualization platforms such as Xen, KVM, QEMU and VMware ESX [78]. Its API is used in the management of cloud resources such as virtual machines, networks and storage. It also provides monitoring information such as the number of CPUs, uptime of an instance, available memory, available and used disk space and network traffic. External monitoring systems for instance, Nagios, Collectd, Munin and Zenoss have plug-ins available for libvirt. Its advantage is that it is used by most of the cloud software stacks and hence does not require additional installation. The primary disadvantage is that libvirt does not provide information about the application RAM usage.

3.4.2.2 Nagios

Nagios[28] is an open source monitoring software, which is widely used by the administrators for tracking the infrastructure in the data-center [28]. It offers thousands of plug-ins that are installed on the physical and virtual servers, amongst them are the plug-in for libvirt and OpenStack. The notable features and drawbacks of this plug-ins are already described in section 3.1.1.9. The data provided by Nagios is retrieved from its check-plug-ins, which are deployed on the virtual and physical servers. The plug-ins monitors services and report the utilization of VMs at run time. They perform system calls on the virtual servers and provide information on dynamic parameters like the current CPU, memory, I/O operations, network statistics, processes and application status.

In EASI-CLOUDS, the implementation of the monitoring component uses two existing libraries: openstack-java-sdk[34] and nagios-jaxws[35]. The latter is a daemon written in Java that is triggered periodically when there is a change in the output written by Nagios on the specified path. The sources of the library were adapted such that it stores the information about individual virtual servers in a suitable data structure. When the clients request for monitoring information, the data from Nagios and OpenStack are aggregated, and offered via a REST API. The disadvantage of Nagios is that it does not store data for longer periods, but Context Store, a component in the project, which performs pre-processing, and persistence of

[34] https://github.com/woorea/openstack-java-sdk
[35] https://github.com/ethiclab/nagios-jaxws

enriched monitoring information, supports this. Additional information about Context Store can be seen in section 3.4.1.

Even though Nagios causes an additional overhead of installing its plug-ins in the VMs, it was chosen to be used in the Mainz testbed for the following reasons: the ease of building, integrating and deploying new plug-ins for monitoring custom applications and processes, availability of external libraries for reading its output and support for distributed monitoring solutions for achieving scalability.

3.4.2.3 Collectd

Collectd[29], similar to Nagios, is a tool that runs as a daemon in background and collects system information with the help of plug-ins. The essential feature is that it supports persisting information in various formats such as CSV and RRD. Unlike Nagios, collectd does not need to be installed on the virtual servers and monitoring information about VMs can be retrieved from the hosts by enabling the libvirt plugin[36] of collectd. The limitation of the plugin is that it does not provide information about the memory usage of VMs.

3.4.2.4 OpenStack Monitoring Solutions

Since the launch of OpenStack, there have been several monitoring solutions, namely efficient metering[37], utilization data[38], system usage data[39], cloud inventory manager[40] and health and monitoring[41]. Ceilometer was introduced in Havana, which is responsible for providing monitoring and metering information about cloud resources. More details about the information offered by Ceilometer are described in sections 3.1.1.7 and 3.1.1.9. It obtains the data from libvirt, which does not provide information about applications, their status or consumed memory.

3.4.2.5 SIGAR

SIGAR[42] (System Information Gatherer and Reporter) is a component of Hyperic's management platform, offers a cross platform, cross language programming interface for accessing the system and hardware level events. Similar to Nagios, it needs to be installed on the virtual servers. The major limitation is that it does not provide any information about the application and the services, which are hosted in the VMs.

3.4.2.6 Ganglia

Ganglia [30] is an open-source monitoring system for high-performance computing systems. It is based on a hierarchical design targeted at federations of clusters. It uses a multicast-based listen/publish protocol within a cluster. Within each cluster, Ganglia uses heartbeat messages on a well known multicast address as the basis of a membership protocol. Membership is maintained by using the reception of a heartbeat as a sign that a node is available. Each node monitors its local resources and sends multicast packets containing monitoring data on a well-known multicast address. All nodes listen for monitoring packets on the agreed multicast address to collect and maintain monitoring data for all other nodes. Each cluster can be

[36] https://collectd.org/wiki/index.php/Plugin:libvirt
[37] https://wiki.openstack.org/wiki/EfficientMetering
[38] https://wiki.openstack.org/wiki/Utilizationdata
[39] https://wiki.openstack.org/wiki/SystemUsageData
[40] https://wiki.openstack.org/wiki/CloudInventoryManager
[41] https://launchpad.net/healthnmon
[42] http://www.hyperic.com/products/sigar

represented with one node, since all the nodes contain a complete copy of the cluster monitoring data.

Aggregation of monitoring data is done by polling child nodes at periodic intervals. Monitoring data is exported using a TCP connection to the node being polled followed by a read operation of its monitoring data. Ganglia Monitoring is implemented by a monitoring daemon, which is organized as a collection of threads, each assigned a specific task: 1) Collect and publish thread: collects local node information and publishes it on a well known multicast channel. It sends periodic heartbeats, 2) Listening threads: listen on the multicast channel for monitoring data from other nodes and updates monitoring data storage, and 3) XML export threads: accept and process client requests for monitoring data. Ganglia Monitoring system assumes the presence of a native multicast capability, an assumption that does not hold for the Internet in general.

3.4.2.7 mOSAIC

The mOSAIC framework [79] offers a Monitoring/Warning system that monitors applications' components and cloud resources. From authors' point of view, this system should realize the following tasks: monitor cloud resources, monitor applications' components and discover warning conditions. The proposed framework contains four basic elements: 1) Monitoring event buses that collect monitoring events from the resources, 2) Connectors related to the event buses to enable the interception of monitoring events by the suitable components, 3) Connectors receiving the events from applications to the event buses, and 4) Monitoring/Warning component. In this system, only one archiver collects monitoring information from different collectors and stores the messages in a storage system, and one component called the observer accesses the storage filled by the archiver and generates events in order to distribute selected information to all the interested components.

3.4.2.8 OVIS

J. Brandt et al. proposed OVIS [80], as a tool for monitoring Cloud resources enhancing high-performance computing in Cloud computing environments. This tool can extract the application and resources state, and based on that state it can assign new resources or shut down unused ones during the application's runtime or for next usages. The data is collected from the resources using data collectors able to collect information and save it in a distributed database. Then, a statistical analysis is necessary to take decisions to keep or to manually reconfigure the resources' assignment for the application. Authors affirm that scalability still remains an area of concern since the monitoring system can be flooded with a big amount of information and that would form a bottleneck.

3.4.2.9 Proposed Solutions

Augusto Ciuffoletti [81] proposed a monitoring infrastructure based on OCCI resources. The infrastructure consists basically on a Sensor resource and a Collector Link. The Collector Link is responsible of the collect of monitoring data from a given resource. The Sensor receives monitoring data and republishes them. The Sensor and the Collector are abstract types specified using mixins: (1) a metric mixin is used to specify how to bring monitoring data from a resource instance to a Sensor resource; (2) an aggregator mixin specifies the eventual aggregation functions to be applied on monitoring data and; (3) a publisher mixin specifies how to publish the monitoring data.

In their work [82], G. Katsaros et al. proposed an architectural approach spanning over virtualization and physical levels to collect monitoring data. They combined many existing open source solutions (e.g. Lattice[83], Ganglia[30], Nagios [84]) to get one holistic application that cover different layers. They use collectors to extract data from different layers (virtual and physical) and externalize it to the upper layer using an external data collector. Moreover, a monitoring manager serves as the orchestrator of the whole monitoring process by controlling and providing the needed interfaces to add or to consume monitoring information. In this approach, only one aggregator is responsible of aggregating and storing all the collected data.

3.4.3 Identity Management

Identity Management (IdM) provides attributes and authentication. From the internet-originated solutions and initiatives the most relevant are: OAuth[85], OASIS SAML v2.0[86], OpenID[87] and all the WS-* specifications[88].

For the purpose of this project, the current IdM solutions are insufficient in a number of points. Motivated by the IoT, IdM should also cover new user attributes such as the things they have, as well as to manage the identity of things themselves (attributes, current users, location, use history, etc.). Furthermore the authentication feature of IdM should also cover the authentication of things for services, other objects or users as relying parties, and the authentication of users, services and other things for things as relying parties. It should also support user Single Sign-On (SSO) across multiple things. Motivated by Cloud computing, the IdM solutions should be able to be run in the cloud; when doing so, special care must be taken (and most probably also adaptation is needed) so that the sensitive data is not exposed to the threats related to the nature of Clouds (e.g. deployment in a public Cloud).

Authorization and usage control policies are best approached by means of the terminology of OASIS XACML (eXtensible Access Control Markup Language), which is a comprehensive access control policy language in XML, together with a conceptual framework and a processing model. It is now considered to be a widely acceptable industry standard. It can be used for handling both general access control cases and specially privacy policies.

Privacy protection enables digital identities to be available to other entities without exposing these identities to privacy threats such as traceability (the digital traces left during transactions), linkability (profile accumulation based on the digital traces), unsolicited marketing (spamming), and loss of control over personal data and identity theft. The most relevant solutions and initiatives in this space are: P3P[89], XACML[90], SWRL[91], as well as the broad range of privacy-preserving technologies, where the most relevant in the space of user authentication are IBM's Identity Mixer[92] and Microsoft Uprove[93]. The noteworthy and most influential EU research projects in terms of privacy and identity management are: PRIME[94], which developed a working prototype of a privacy-enhancing identity management system, PrimeLife[95], which was PRIME's follow-up project that aimed at ensuring that the community at large adopts privacy technologies, as well as ABC4Trust[96], which brings trustworthy yet privacy-preserving Attribute-based Credentials (ABC) into real live pilots.

3.4.4 Resource Reservation

Cloud service provider can offer two basic resource-provisioning plans: reservation and on-demand plans. In general, cost of utilizing computing resources provisioned by reservation plan is cheaper than that provisioned by on-demand plan. If a cloud consumer wants reduce their cost for resource use then it is necessary to minimize the total provisioning cost by reducing the on-demand cost and the cost of under provisioning and over provisioning. The under provisioning can occur if the reserved resources are not enough for the demand and over provisioning can occur if the reserved resources are more than the actual demand. In [97], [98], and [99] an Optimal Resource Provisioning Algorithm (OCRP) are discussed. This algorithm minimizes the provisioning cost of the resources. Stochastic integer programming and deterministic equivalent formulation are used for this OCRP algorithm. Heuristics of such an algorithm are discussed in [100].
For the cloud service providers it is difficult to allocate the cloud resources dynamically and efficiently. Job-oriented resource scheduling becomes a very complicated task in a cloud-computing environment where many alternative computers with varying capacities are available. Efficient task scheduling mechanism can improve the resource utilization. In [101] were analysed various scheduling algorithms and tabulated various parameters. Furthermore it was noticed that disk space management is critical issue in virtual environment.

In general, a cloud service provider has a restricted infrastructure (physical resources). If a cloud service provider wants to guarantee a reservation of resources within a certain time interval or the completion time of a big job, then the provider needs a management of the resource reservations in advance.

Resource reservations in advance were discussed in [102] and it was used a game theoretic approach for the proof that a truthful reservation is the best. In [103] was proposed a model for optimization of SLA-based resource schedule in cloud computing based on stochastic integer programming technique. The considered problem is a combinatorial optimization problem, which ensures the optimal mapping between each abstract service and available resources. A scheduling strategy that performs reservation for prioritized jobs and dynamic scheduling of the Cloud is presented in [104].

For the management of resource reservation we develop a Resource Manager. The Resource Manager is a high-level component for checking the availability, managing resource reservations, and “abstract” scheduling (determines/checks the time of the availability, but no allocation of concrete resources). The main task of the Resource Manager consists in guaranteeing that the necessary resources are available for every by contract agreed service when they are needed, so that the SLA-conditions of the service realization concerning the resources can be kept. The Resource Manager is tightly coupled with the SLA negotiation and supports the SLA Manager of a provider for the decision whether the resources needed for a requested service are available in the desired time interval. In contrast to other approaches (e.g. [103], [104]) we consider dynamic resource reservation with assigned time slots of a more abstract level instead of resource scheduling strategies.

For the implementation of the Resource Manager Constraint Programming is used. Constraint Programming is based on the idea that many complex problems can be expressed declaratively in terms of variables and constraints. The variables range over a (finite) set of values and typically denote alternative decisions to be taken. The constraints are expressed as relations over subsets over variables and restrict feasible value combinations for the variables. A solution is an assignment of variables to values which satisfies all constraints.

Constraint Programming with constraints over finite domains has been established as a practical tool for solving discrete combinatorial problems, especially in the field of resource management, scheduling, and advanced planning. An Example for the application of Constraint Programming in the field of advanced planning and scheduling is given in [68]. In this year 2014, a workshop on "Cloud Computing and Optimization" will be firstly organized during the Conference on "Principles and Practice of Constraint Programming" (see [105] for an example of proceedings). This workshop will bring together interested researchers from Optimization/Constraint Programming and Cloud Computing communities, and shows the actuality of this topic.

3.5 Real-time Rating, Charging and Billing

This chapter deals with the accomplishment of a state of the art analysis regarding mechanisms, formats and protocols that can be used for real-time Inner- and Inter-Cloud rating, charging and billing in the scope of the EASI CLOUDS project.

For the state of the art analysis in the EASI CLOUDS scope specific topics are given particular consideration that will be presented in the following chapters.

3.5.1 Terms and Definitions

3.5.1.1 Rating

Rating means the determination of costs of a particular data (service or respectively resource) into a monetary-equivalent value. This rating will be done in real-time over a corresponding real-time connection using a request-response message protocol. Real-time Rating enables various features such as Cost Prediction, Cost Control or respectively Bill Shock Prevention.

3.5.1.2 Charging

The real-time Charging allows to perform charging operations in real-time via different charging types (see 3.5.1.6). Also several charging models (see 3.5.1.5) are supported.

The Charging system holds all monetary and non-monetary information about resources and services relevant for billing. Customers may have several balances for different purposes filled with monetary or non-monetary units. Thus, a 'price' may be expressed in monetary units, but may also use non-monetary units like reward points, bonuses etc.

A special Charging features is the ability for real-time Revenue Sharing. This provides the possibility to automatically split the revenue from all charges between all parties involved.

3.5.1.3 Billing

The process of regular bill generation includes aggregation of event data, calculation of charges, and reporting the bill details in a structured way (often also the term invoicing is used equivalent). One special feature of real-time Billing is the ability to create/generate on-demand bills at any time, independent from the common bill cycle.

3.5.1.4 Payment Types

When talking about payment there are three basic types that have to be differentiated – Prepaid, Post-paid and - the combination of both – Convergent:

- Prepaid
  - Payment in advance
  - Usage only if credit is left
  - Configuration of thresholds and notifications
  - Cost control
  - Bill shock prevention
- Post-paid
  - Usage without restrictions
  - Payment at the end of a specified interval (week, month, year,)
- Convergent
  - Combines the benefits of Prepaid and Post-paid
  - Convergent pre / post online and offline charging
  - Comprehensive financial management
  - Charging and billing for all services, customer segments and payment methods
  - Real-time rating for post-paid customers
  - Unified/Single subscriber repository for advanced service differentiation
  - Embedded policy control for data service monetization

3.5.1.5 Charging Models

A variety of different charging models exist. The most common ones are listed below:

- Usage based / Pay per use
  Usage dependent parameters (e.g. amount, time, number of requests,) are metered, mediated and charged.
- Freemium
  Free of charge usage for a limited testing period, continuing with flat rate or usage based tariff.
- Flat rate
  Fixed flat charge for a defined period (per week, month, year,).
- Roaming
  Access to or from other networks.
- Dynamic pricing
  Pricing can be influenced by various environment parameters such as “current load of the system” or “current energy costs”. Ability to easily change tariffs
- Quality of Service (QoS)
  Charging is done based on service level agreements (SLAs) that have been agreed on by the service provider and the service customer.

Furthermore many of those models can be combined to more sophisticated charging models, depending on what is to be charged at which degree of detail.

3.5.1.6 Charging Types

Charging operations can be performed in real-time either directly via "Direct Charging" or using reservations in a "Session-Based" manner (i.e. by subsequent reserve, extend, and charge operation calls).

- Direct Charging (important for direct payments)
- Reservation Based Charging (important for sessions/transactions)

3.5.1.7 Charging Levels

Usage based rating, charging and billing can be applied to each of the existing levels:

- IaaS (Infrastructure as a Service)
  - Quantity
    - Number of CPUs
    - CPU time
    - RAM
    - Storage
    - I/O network usage
  - Quality
    - Availability
- PaaS (Platform as a Service)
    - Server size
    - Operating systems
    - Available development frameworks
    - Available development tools
- SaaS (Software as a Service)
    - Charging of individual applications
    - Bandwidth
    - Volume (data)
    - Duration (time)
    - Number (requests)

3.5.2 Real-time Rating, Charging and Billing Mechanisms

The availability of appropriate mechanisms for accounting and billing enable Cloud computing providers and users to see which models fit best to their needs. Therefore, there is a need of an integrated support for metering, rating, charging, and billing of services based on different charging models (see 3.5.1.5) with architectural support for automatic and adaptive monitoring and management of allocated resources and Service Level Agreements.

- Metering Mechanism:
  Real-time metering of service and user data
- Penalty-Mechanism:
  Service level agreements (SLAs) to guarantee the required QoS (Quality of Service) and grant drawback in case of failure (penalties)
- Rating-Mechanism:
  Intelligent/Context-aware real time rating (Advice of Charge depending on actual consumption of resources and quality of service)

- Cost Control Mechanism:
  Ability to preview the current costs at any time (Bill Shock Prevention)
- Revenue Sharing Mechanism:
  Possibility to automatically split the revenue from all charges between all parties involved
- Billing Mechanism:
  Ability to generate bill previews or final bills on demand
- Real-time Event Notification Mechanism:
  Charging events in combination with e.g. thresholds can trigger events that can be used for simple notifications or as input for complex Policy Management systems (e.g. bandwidth management)
- Customer and Account Management Mechanism:
  Flexible customer and account management functionality to manage customers and their corresponding accounts in the Rating, Charging and Billing System
- Tariff Change Mechanism:
  Flexible change of payment models (tariff changes) or prices

A professional Rating, Charging and Billing System for the Cloud should support all these mechanisms to provide a maximum of flexibility, reliability and control for the customers as well as for the service providers.

3.5.3 Standards

An important topic regarding the state of the art is the evaluation of existing standards and standardization groups.

3.5.3.1 Standards in Cloud Environment

In March 2012 the German Federal Ministry of Economics and Technology has published a report about "The Standardisation Environment for Cloud Computing“[106]. This comprehensive report analyses the most important German, European and International standardization organizations such as:

- ETSI
- NIST
- Open Cloud Consortium
- OSGi (Open Source Gateway Initiative)
- OSCi (Open Source Cloud Initiative)
- BITKOM

One result of this report was the fact, that “There exists no standard for Cloud Billing interfaces and corresponding protocols”. There are some “approaches” for billing in one or two of the standards, but according to the report these are far from what would be required from a general concept as such a concept should be covering major acceptance requirements such as “Transparency” with respect to:

- Pricing Models available/offered
- Traceability of Charges that are effected
- Dedicated Billing
- Agreements that have been agreed on by the involved parties

Furthermore the respective approaches for billing are restricted to specific areas or respectively cover only subsets of those areas, e.g. billing of only IaaS or parts of it.

3.5.3.2 Standards Applicable to Cloud Environment

Extending the search for billing standards to other industries may fill the gap by analysing the potential applicability of well-established standards from these industries.

The most promising candidate is the GSMA OneAPI[107] standard (former OSA Parlay / ParlayX) used in the telecommunication area (OneAPI). The OneAPI initiative defines a commonly supported set of lightweight and Web friendly APIs to allow mobile and other network operators to expose useful network information and capabilities to Web application developers. It aims at reducing the effort and time needed to create applications and content that is portable across mobile operators. Nevertheless, we think that this standard is also well applicable for the EASI-CLOUDS project, as it comprises a set of REST-based APIs for the required interfaces Rating, Charging and Billing, Account Management and User Management.

3.5.3.3 De facto Standards Applicable to Cloud Environment

Besides all the standardization organizations there is a broad landscape of de facto standards being recognized and used by a wide audience. The most interesting de facto standard with respect to pricing and billing is USDL.

USDL is a platform-neutral, generic language for describing business, operational and technical aspects of services for the “Internet of Services”. This combination of Technical as well as Business and Operational Service Information is also called “Unification of information”. USDL is a platform-neutral, generic language for describing business, operational and technical aspects of services for the “Internet of Services”. This combination of Technical as well as Business and Operational Service Information is also called “Unification of information” as depicted in Figure 3.6.

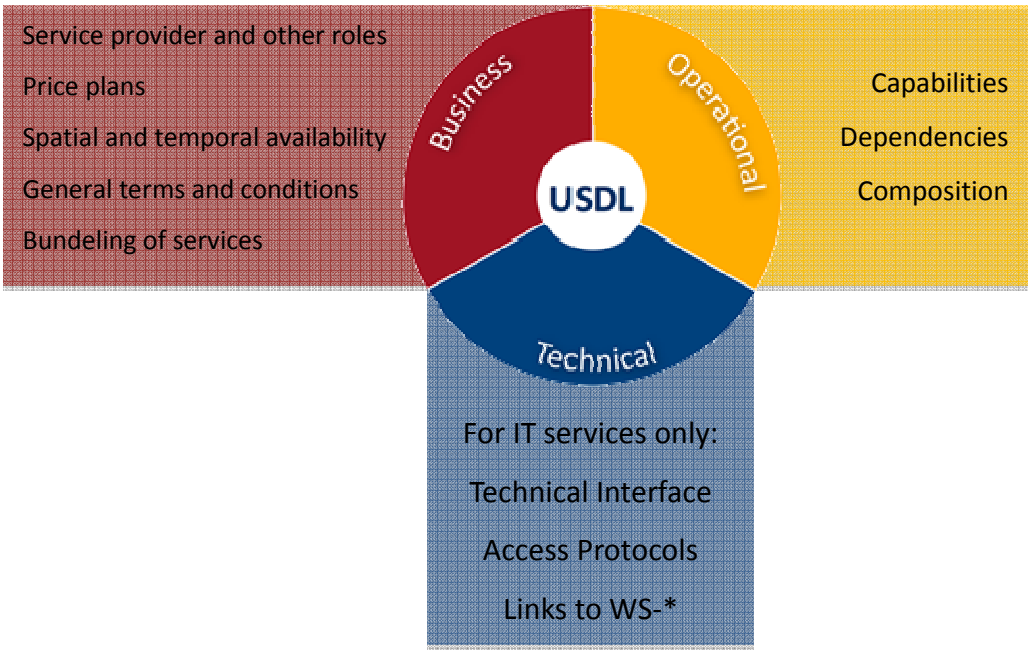

Figure 3.6: USDL Overview

The initiating members are SAP, Siemens, DFKI and Attensity (formerly Empolis). USDL is built in a collaborative and interdisciplinary way. Modelling is done in the context of several publicly funded research projects under the "Internet of Services" theme:

- German Federal Ministry of Education and Research projects
  - TEXO (project within the THESEUS1 research program)
- EU DG INFSO projects
  - FAST
  - RESERVOIR
  - MASTER
  - ServFace
  - SHAPE
  - SLA@SOI
  - SOA4ALL
- Australian Smart Services CRC

USDL defines normative UML class models and a corresponding serialization in XML Schema for capturing "master data" of services. USDL on a whole is made up of a set of modules, each addressing different aspects of the overall service description.
Modularization was introduced to improve readability of the model, which drastically grew in size compared to its predecessor. The modules may reuse concepts from other modules, so they have dependencies among each other (shown in Figure 3.7 below):

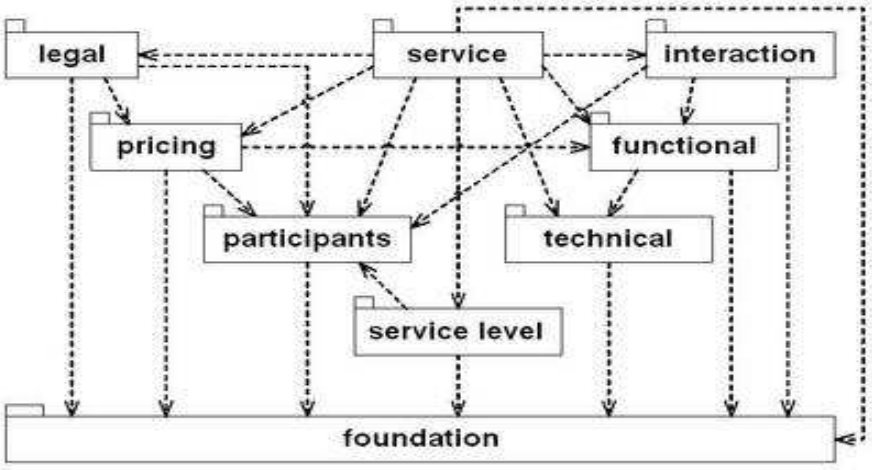

Figure 3.7: USDL modules and dependencies

USDL provides a "pricing" module with a cascading backbone structure. This pricing module offers three basic elements in a strict hierarchical structure:

- PricePlans (set of charges associated with an entity)
- PriceComponents (fees included in a PricePlan, which may contribute to the total amount charged)
- PriceLevels (capture amounts charged by a PriceComponent)

This structure allows for a very flexible scenario modelling with various features, such as e.g.:

- Assign alternative price plans to an offered service or bundle
- Each plan possibly made up of multiple components
- Each component possibly varying its charges
    - By specifying different levels
    - By adjusting them by means of premiums and discounts
- Constrain elements by segmenting conditions detailed in price fences (i.e. criteria a customer must meet or the service limitations he/she needs to accept to qualify for a certain price)

3.5.4 Market Requirements

When analysing the Cloud Billing market requirements there are basically two categories – the demands that customers of Cloud Billing solutions (e.g. Cloud service providers) have on the respective vendors and the strategies that Cloud service providers (CSPs) pursue in the Cloud Billing market[108].

3.5.4.1 Demands on Cloud Billing Vendors

Cloud Billing vendors should provide guidance for configuration of current billing systems with cloud services. They need to provide support for current cloud pricing models, as well as for other enterprise services. Sophisticated mediation tools are required to provide detailed usage information for each customer and service. Easy integration of new virtual environments and associated management platforms is required. Furthermore the Cloud Billing system needs to incorporate Multi-Tenancy capability to support customer hierarchies

and complex multi-party service value chains. Cloud service providers are a new market segment whose requirements can be solved by the Cloud Billing vendors. Therefore the Cloud Billing systems need the Ability to address standalone CSPs as potential customers.

3.5.4.2 Strategies of Cloud Service Providers

From the Cloud Service Providers' point of view, modern billing systems are already able to support the requirements of today's Cloud services. In the mid- to longer term some changes will perhaps become necessary to support more-advanced usage-based pricing metrics. Overcomplicated billing options make it difficult for customers to compare the costs of in-house resources with those of cloud service offerings, and will tend to slow or prevent sales. In the future, however, customers will need sophisticated usage-based pricing models that enable them to optimize their costs in order to be economically viable. CSPs in general may need to consider how to support resellers of their services to manage and to bill their users. Here the "Revenue Sharing" mechanism could be an appropriate tool for realization of this feature. Furthermore, CSPs should consider that their cloud services might be sold to new market segments. This may place additional requirements on billing systems - such as the need to support other tax schemes or to provide different customer information.

3.5.5 Summary

To fulfil the promise of cloud computing with all charging types and models on all levels of the Cloud, flexibility in billing - the complexity of which is comparable to convergent telecom billing - is a key ingredient for cloud providers. Orga Systems' real-time rating, charging and billing system in combination with USDL for standardized configuration of the system and the GSMA OneAPI as the standardized web interface to access all functionality of the billing system is the key to enable real-time rating, charging and revenue sharing on all levels in the cloud computing domain. This approach is the basis for tailored cost models, cost control and transparency for all involved parties.

In addition – following the market requirements (see 3.5.4) – there is a need for a Billing Service that can be booked and used in/from the cloud. The main characteristics required are:

- Configuration of price plans in a simple way (e.g. via Web-GUI)
- Customized API for easy and flexible integration into services
- Amenities of a professional billing system without the efforts and costs for its hosting, service and maintenance
- Standardized interfaces for integration with already existing billing systems

This would especially be leading to new business opportunities for small and medium-sized enterprises in the cloud business.

3.6 Data Privacy and Security

3.6.1 Data Privacy

Data privacy is perhaps one of the most controversial topics revolving around the cloudification trend. By entrusting our private and personal data to cloud service providers, we're giving away the direct control on just how, by whom and for which purposes that same data is used, stored and processed. The problem domain can roughly be divided into two areas: the personal aspects of data that is gathered from us during normal course of service utilization, and the content we process or store on cloud but do not explicitly share.

3.6.1.1 Personal Data and Regulatory Environment

The European Data Protection Directive [95/46/EC] regulates personal data processing and forms basis for data handling requirements within Europe. Its protections are implemented into national laws of each of the EU states. Under the directive, personal data is defined as "*any information relating to an identified or identifiable natural person*" when such person can be directly or indirectly identified from the said data. Processing such data is prohibited, unless the processing meets the specific requirements set forth for such processing. Sensitive data, such as medical health, religious beliefs, political opinions, sexual orientation, associations and race, require special care.

Under the Data Protection Directive and the national laws, private data collection generally requires user consent before any storage of data. Similarly, user has the rights for notification of such data collection, and right to review and rectify invalid data. Data subjects have the right to request erasure or blocking data that is not compliant with the EU national laws.

Data classified as personal data cannot be transferred outside European Economic Area unless the data processor can guarantee that the recipient complies with the European data protection rules. To streamline working with US companies, European Commission has agreed to the Safe Harbor process [2000/520/EC]. Companies opting in to the process by adhering to the principles and a specific FAQ section of the decision are allowed to receive European Personal data.

The major cloud providers at the time come from the US. It should be noted that US does not have one single data protection law covering personal data, but rather multitude of laws governing specific circumstances and/or data types. Typically, the data usage is self-regulated and set forth in terms of conditions and/or privacy policies.

European Commission is in the progress of reforming the EU data protection legislation, aiming to eliminate differences in 95/46/EC implementations and create a single set of rules that would be valid throughout Europe. The proposal will also cover the new data protection challenges that have risen from the rapid technological developments, globalization and the effects of ubiquitous social-media, cloud computing and location-aware services.

User content that falls outside the scope of personal data is governed by contracts and privacy policies between the user and the Cloud Service Provider. There currently exists no standard for the format or content of a privacy policy.

3.6.2 Data Security Standards

While Cloud Service Providers often keep technical implementation details private, they must follow good security practices and ensure safe and sound confidentiality and availability for the offered system. Building trust requires transparency on the adequate security practices. Certifications are one good way of creating trust on establishment and maintenance of proper organizational security practices.

Currently the most widely adopted global security standard is ISO 27001. It specifies requirements for implementing and maintaining an information security management system. It also includes requirements for risk assessment for security threats for IT systems.

The Cloud Security Alliance STAR certification is a third party independent assessment of the security of a cloud service provider. The certification builds on top of ISO 27001 and Cloud Security Alliance defined Cloud Controls Matrix.

Service Organization Control (SOC) reports are audit reports used for building trust and confidence in selected security principles. The audits are performed and the reports written by independent certified auditors. The principles include the traditional security principles: confidentiality, integrity, availability, and authenticity. American Institute of Certified Public Accountants defines the certification scheme.

The Payment Card Industry Data Security Standard (PCI DSS) is a security standard aimed at organization handling credit card information. Cloud Service Providers can certify their part of IaaS/PaaS stacks for compliance to instil trust in the proper security controls.

4 Evolution in Business

4.1 Summary of The General Public Cloud Computing Market

Typically the public cloud market is segmented into infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and software-as-a-service (SaaS). The general pattern in these sub-markets is that their market size grows when moving up the value chain. That is, the SaaS market is considerably larger than for example the IaaS market. Similarly, direct price competition is more intense in the IaaS market, whereas the SaaS markets offer firms plenty of opportunities to differentiate. Despite the promise of public cloud computing, it is common for particularly large organizations to possess information that they simply are unwilling to place into public cloud. Hybrid cloud seeks to combine the best elements of private and public cloud computing. Gartner forecasts that by 2017, half of large enterprises will have hybrid cloud developments[109]. The company also observes that in terms of aspiration and adoption, hybrid cloud is currently in a similar position as private cloud was three years ago.

Figure 4.1: Estimates of main cloud computing segments (excluding BPaaS). Source: Gartner[110]

Figure 4.1 depicts Gartner's view on the main public cloud computing segments and their sizes and forecasts. The largest individual component of the cloud computing market is Cloud Business Process Services (BPaaS)[43]. It is debatable whether these services are actually a part of the cloud market[111], because the concept includes a rather open-ended inclusion of legacy systems and business process outsourcing as long as relevant parts are sourced from the cloud[112]. Depending on its inclusion, Gartner's estimates of the public cloud computing market reside between $35Bn and $75Bn for 2013[44]. IDC's estimates the public cloud market to be at $45.7Bn in 2013[113]. Forrester estimates the public cloud market to be $58 in 2013[114]. Both IDC and Gartner expect the public could market to roughly double in the next three years. These estimates were however mostly made before the widely publicized Snowden revelations regarding the NSA, which further fuelled concerns related to

[43] Future reports by Gartner also consider cloud advertising as a part of the public cloud services market with $677Bn revenue.
[44] Here we also exclude 'cloud advertising'.

information privacy. Yet, while some more recent market estimates have been slightly revised down, the primary reason appears to be related to the macroeconomic situation.

Other research institutes have also given estimates on the sizes of the different cloud market segments. Hosting provider Parallellis estimates that the SaaS market for small and medium businesses was $14.5Bn[115] compared to Gartner's estimate of $16Bn for the entire SaaS market. Forrester, on the other hand, places the SaaS market at $47Bn in 2013. According to Gartner, the most significant SaaS segments in 2013 are CRM ($3.4Bn), ERP ($1.5Bn) and conferencing/team platforms ($1.8Bn), and these segments are expected to maintain their relative order also in 2016. Overall, North America is a clearly the largest market for public cloud services (Figure 4.2); It has been estimated that the West European market constitutes as little as about a quarter if its North American counterpart with Asia's combined modest market share being dominated by Japan. However, market growth in West Europe would also be significantly faster.

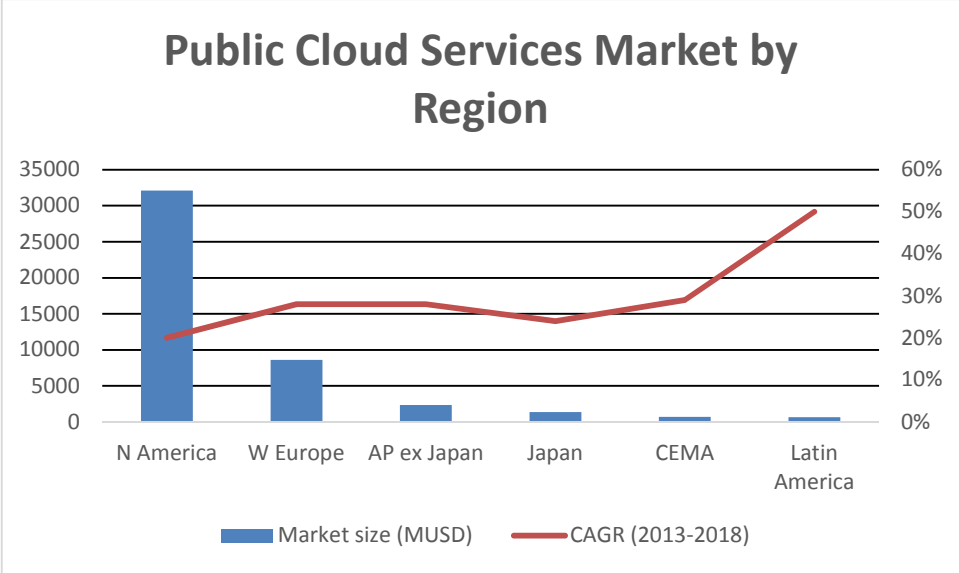

Figure 4.2: Geographical distribution of the public cloud market according to IDC[116].

If BPaaS is excluded, most of the cloud market resides in SaaS. Analysts, however, seem divided between the revenue distribution between IaaS and PaaS: Gartner sees IaaS to be significantly larger ($9Bn vs. $1,6Bn in 2013), and that its dominance over PaaS would continue. Forrester also sees that IaaS dominates over PaaS ($5,6Bn vs. $4,4Bn in 2013), but that their order would change as early as 2014. The variations in forecasts may reflect both the fundamental difficulty in predicting how a dynamic market will evolve, and differences in how key concepts are defined. Gartner expects the global SaaS market to grow at approximately 20% during the new few years, while growth in IaaS is above 40% and approximately 30% for PaaS[110].

To give these numbers some more context, Gartner forecasts that total global IT spending in 2014 will be $3,75 trillion[113], while Forrester gives an estimate of $2,2 trillion[117]. In other words, public cloud services would be in the order of one or two per cent of total IT spending. Public cloud services also remain fragmented when looking at the whole market. However in IaaS/PaaS it has been estimated that Amazon is the clear leader by having approximately one quarter market share, which is slightly more the three following

competitors: IBM, Microsoft and Google[118]. The SaaS market is more fragmented because it addresses a broad range of user needs that are not in direct competition with each other. For example, Salesforce.com, a leading SaaS company, obtained revenues of $3Bn in its 2013 accounting period[119], which would give it an 18% market share in SaaS[45].

Taken together, the public cloud market as a whole poses good potential for new technology-based entries: there is strong growth, and new positions are opening up in the cloud value chain as the market matures and standards –whether formal or de facto– gain ground. Also even some of the strongest players in IT (e.g. IBM, Google, Microsoft) in have limited market shares in IaaS/PaaS, while Amazon remains a clear leader as competition has picked up. Apparently the market has not fully consolidated into a pure volume business, and that new entrants may be able to differentiate through their offerings. On the other hand, the growing cloud services market constantly requires new enabling technologies and services that form an interesting opportunity on their own – both in the realms of public and private cloud.

4.1.1 A More Detailed Look at the Cloud Value Chain

In this section, we provide a more thorough view on the cloud computing market by analysing different parts of the cloud value chain in addition to IaaS, PaaS and SaaS. A value chain is a chain of activities that a firm operating in a specific industry performs in order to deliver a valuable product or service for the market[120]. Value chain positions are commonly used to segment markets and analyse general competitive dynamics, such as market entries and exits. Adopting value chains as a lens provides a higher perspective on implications of the EASI – CLOUDS project, even though the project resides at higher levels of the cloud value chain.

When looking at the cloud market, it is important to acknowledge that it is not "fluid", in the sense that any player in a given level of the value chain can freely transact with all entities below and above it. In other words, value chain positions are linked to each other through markets that are far from perfect[46], though the emergence of new firms in areas that were previously internalized by large players is taking place. For example, preferential access to large player's IaaS resources and data communication infrastructure are important market drivers. Second, the IaaS, PaaS and SaaS space witnesses frequent market entries from players that occupy other positions in the cloud value chain. For example, network service providers have been actively entering the public cloud market, i.e. they have vertically integrated into data center operation and IaaS/PaaS parts of the value chain.

In the following, we review some essential positions in cloud value chain. Our view is focused on an end user, who is of non-technical nature, and is primarily a consumer of SaaS-based offerings[47]. Figure 4.3 depicts a summary of the value chain, which is discussed in the following.

[45] Authors' calculations using Gartner's numbers (and making the simplifying assumption that all of Salesforce's revenues come from SaaS)

[46] A perfect market is a theoretical construct in economics that includes, among other things, the free entry and exit of buyers and sellers, and perfect information for all transaction parties.

[47] Therefore, it should be noted, that the value chain might look very different from the perspective of e.g. a SaaS provider especially with respect to technical consulting services. It is also possible to decompose the value chain even further especially at its lower levels. Here we put more emphasis on higher value chain positions that are more relevant from the perspective of the EASI-CLOUDS project.

4.1.2 Cloud Business Consulting Services

Cloud business consulting services refer to a segment of management consulting services that are related to cloud business. Customers of these services are business decision makers who seek to exploit business opportunities related to cloud computing. Cloud business consulting services can include for example analysis of market entry strategies, the actions and positions of competitors, merger and acquisition opportunities, new production technologies, and product and service portfolio management. The focus on technology (including cloud computing) is on its business implications, and the resources and actions that are required to develop and exploit the technology, rather than how the technology actually operates.

Despite its long history, analysts reach varying results when sizing the global management consultancy market mostly due to differences in definitions. Estimates of the global market size vary between $ 95 Bn and $ 344 Bn[121] with market growth estimates ranging between 4-2-5%[122]. According to estimates, the management consulting market is dominated by the EMEA region and North America, which both have roughly the same size, and jointly occupy about 80% of the total market[121].

Cloud computing touches upon many areas of management consulting, and to a varying degree. However, perhaps the most significant areas are strategy and operations. These segments are estimated to have revenues of 30 and 60 billion USD respectfully, with growth rates slightly higher than the general management consulting market[48]. Based on these figures and centrality of cloud computing related issues in these areas, the order of magnitude for the global market for cloud business consulting services is perhaps around $ 50 billion with the US market being a clear leader. However, the market for business consulting where cloud computing plays a central role is arguably smaller.

Significant players in the management consulting market include likes of McKinsey, Boston Consulting Group, Bain & Company ('the big three'), Accenture, and Strategy& (owned by PwC, formerly Booz & Company). Companies like Deloitte, PricewaterhouseCoopers, Ernest & Young and KPMG have significant management consulting operations despite being better known for their auditing and accounting services. Many firms that are better known from technology consulting also provide business consulting services, for example IBM, Microsoft, Atos, Thales Group, and Bull. The distinction between business consulting and IT consulting is naturally vague in many situations due to their close relatedness.

4.1.3 Cloud IT Consulting Services

Cloud IT consulting services aim to inform managers on how to exploit cloud technology for business purposes. IT consulting can also include technical outsourcing services, such as custom software development, system integration, deployment and management, and vendor selection. The customers of cloud IT consulting services include both technical and non-technical managers.

Based on Forrester's market decomposition[123], we estimate that the global market IT consulting is roughly $400Bn. The subsegment of this market that addresses cloud-specific issues is often called cloud professional services. IDC estimates the size of this market to be

[48] Sometimes IT is also considered to be a segment of management consulting, however here we place it under IT consulting.

$9.6 Bn in 2013 with CAGR of 24.8%[124]. The growth rate of this service category is hence about 5 times greater than what Forrester estimated for the IT market in Europe.

IDC views IBM and Accenture to be the leading cloud professional services firms. Major players include PwC, Infosys, Fujitsu, CSC, Microsoft, Dimension Data, Wipro, Cisco, and HP. In addition to PwC, Capgemini is the only European company in IDC's analysis of the top 13 vendors, which the firm categorizes as a 'contender' in terms of its capabilities and strategy. While the cloud professional services segment may not capture all essential parts of IT consulting that are related to cloud computing, the market is clearly US-dominated.

4.1.4 Cloud Brokerage

A cloud broker is an entity that manages the use, performance, and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers[125]. We provide a deeper overview into the cloud brokerage definitions and market in D1.5, and here we briefly summarize the market in the context of the broader cloud value chain.

Here we consider that cloud brokers to operate on IaaS and PaaS services, and provide services in two main categories. In service intermediation, a cloud broker enhances a given service by improving some specific capability and providing value-added services to cloud

consumers. In service aggregation/arbitrage, a cloud broker combines and integrates multiple varying services into one or more new services.

In these sections we conclude that the market size estimates for cloud brokerage (from $1.6Bn currently up to $100Bn already in 2015) vary significantly mostly due to variations in definitions. If other value-added consulting services are excluded, the cloud brokerage market can also be sized by reviewing the market of public PaaS and IaaS markets. If, for example, we optimistically assume that a broker collects a 20% commission on 50% of all IaaS and PaaS transactions, the size of the brokerage market would be in the range of $1Bn USD (see section 4.1 for figures on the IaaS and PaaS markets). This approach would also imply that after an initial growth period, the growth of the cloud brokerage market would converge on the growth rate of the public cloud market (primarily IaaS and PaaS), and that the same geographic distribution of revenue seen in the public cloud market would also reflect on the cloud brokerage market. However, if other forms of cloud-related consulting are included, the brokerage market looks significantly larger, and geographic differences (e.g. between Europe and the US) will likely be smaller.

Barriers of entry into cloud brokerage can be low when we consider the case of a human-delivered professional service. In essence, any IT service provider (e.g. telcos and IT consulting firms) can enter the cloud brokerage market almost unavoidably through customer projects that relate to cloud deployments. The situation is however different for companies specializing in brokerage that deploy automated platforms. In their case, up-front investments into technology and marketing are required, and economies of scale will ultimately dominate especially less differentiated markets[49]. Cloud brokers with high volumes can also gain bargaining power over cloud suppliers and gain higher margins. Currently numerous companies are entering the cloud brokerage market[50].

4.1.5 Cloud Federation

Cloud federation is the possibility for a cloud consumer to send a cloud request to multiple cloud providers as if they were a single cloud provider.[51] Cloud federation (‘intercloud’, ‘cloud of clouds’) enables cloud service providers to ‘pool’ together their data center resources with the aim of being able to jointly offer more comprehensive and especially more flexible cloud resources to their customers. In this section, we briefly review the concept and provider a deeper market overview in D1.5.

Based on our review, London-based OnApp is currently the only significant commercial actor that operates a cloud federation (OnApp CDN and cloud storage). The company provides a software solution that enables cloud service providers to sell their excess cloud resources or obtain additional capacity from other users of the platform. The federation currently spans 170 locations in 113 cities across 43 countries[126]. The company also operates Cloud.net, which is a marketplace for resources in the federation. Another starting player is the new Deutsche Börse Cloud Exchange[52]. Their public trading platform for IaaS resources is in a public beta testing phase.

[49] We see e.g. flight search engines as relatively similar business segment, where competition is primarily based on price and only a handful of players can exist on the market due to low margins.
[50] A useful list of cloud service brokerage companies is available at: http://talkincloud.com/cloud-services-broker/cloud-services-brokerage-company-list-and-faq
[51] This is a common definition adopted by the EASI-CLOUDS project consortium.
[52] http://www.dbcloudexchange.com/

Due to its nascence, it is difficult to estimate the size of the cloud federation market and how it will develop[53]. However, we expect that the number of "horizontal" cloud federations, i.e. federations that seek to compete directly with players like Amazon and Microsoft, will remain very limited due to strong network externalities[54]. However, it is likely that the market could support a higher number of "vertical" federations that address the special needs of certain industries. Potential entrants into cloud federation include other technology enablers (e.g. OnApp) and cloud integration service providers. In addition, small CSPs and public sector entities have the incentive to form federations. We also see that there is also an internal latent market for federations in large corporations seeking to improve the efficiency of their distributed IT resources.

4.1.6 Traditional Service Types

SaaS (Software-as-a-Service) market can be segmented most clearly into enterprise and consumer markets. Like in the case of the software market as a whole, the consumer SaaS market represents a small fraction of the total market. The most important enterprise SaaS segments include CRM (customer relationship management), ERP (enterprise resource planning), and SCM (supply chain management). Gartner has forecast that the enterprise SaaS market in Western Europe in 2014 will be $4.2Bn, which represents less than a quarter of the global market of approximately $19Bn. It is also less than half of the US market[127].

SaaS is broadly considered the largest segment in public cloud computing in terms of growth and size and also the most differentiated one. PwC maintains a list of top companies in terms of SaaS revenue[128]. The leading firms in their listing include Salesforce.com ($2,7Bn), Microsoft ($1,4Bn), Intuit ($1,2Bn) ADP ($1,2Bn), SAP ($1,1Bn), Oracle ($1,0Bn), and Cisco ($0,8Bn). In addition to SAP, DATEV is the only company to make PwC's top 20 list from Europe with estimated $0.4Bn SaaS revenue.

A special category of SaaS is *SaaS aggregators* that create a value added service by combining a set of existing external SaaS offerings[55]. The set of services being aggregated is mostly fixed, and the number of possible services is low. SaaS aggregators can for example give users better control of their data, contracts, and billing that is spread out over several SaaS providers especially in enterprise markets (e.g. CloudConnect, Sigma Systems). On the consumer side, F-Secure's Younited service provides a common data management interface for many cloud storage and social media platforms in addition to cloud storage services hosted by F-Secure.

PaaS (Platform-as-a-Service) market size estimates range from $1.6Bn (Gartner) to $4.4Bn (Forrester). Gartner estimates market growth to be approximately 25% in the next few years. Leading companies in PaaS include Amazon (e.g. elastic beanstalk), Salesforce (force.com), Microsoft (Azure), IBM (SmartCloud), Google (AppEngine), Redhat (OpenShift), Pivotal Software (e.g. CloudFoundry), CloudBees, and EngineYard.

[53] Especially, OnApp is a private company and its financial statements are not available.
[54] This is a similar case for airline alliances: having a very high number of them would defeat the benefits to member airlines. In other words, all things equal, a CSP gains more value by joining a larger federation than a smaller one.
[55] In some cases, this may be considered a part of cloud brokerage. However, due the differentiated nature of SaaS, we distinguish between these two value chain positions.

IaaS (Infrastructure-as-a-Service) market size estimates range from $5,6Bn (Forrester) to $9Bn (Gartner). Gartner places IaaS's growth at over 40% in the upcoming years. The market is characterizable by its intense price competition where double-digit price drops have been common in recent years. In IaaS/PaaS it has been estimated that Amazon is the clear leader by having approximately one quarter market share, which is slightly more the three following competitors: IBM, Microsoft and Google[118]. IaaS is clearly a volume business, but its applicability is also clearly limited e.g. in several governmental sectors and also in many enterprise contexts due to data security and control issues. In essence, the IaaS market does not yet effectively serve all market needs, and we see more potential for vertical offerings in terms of region and industry.

4.1.7 Platform Enablers

Platform enablers are complementary software services that facilitate the development and provisioning of IaaS, PaaS and SaaS services. IaaS (or PaaS) enablers include proprietary and open-source cloud computing software orchestration/virtualization platforms like OpenStack, vCloud, Hyper-V, Xen and Eucalyptus. PaaS enablers for example include e.g. proprietary infrastructure Google App Engine and Azure platform software, and AppScale; and configuration management/orchestration platforms like Chef and Puppet. Platform enablers for SaaS form a diverse highly diverse group. Examples range from payment solutions (e.g. Avangate, Orga) to various broader ecommerce frameworks etc. and common APIs used in mashups (e.g. Google maps, Facebook comments).

The EASI CLOUDS project is primarily interested in the development and integration of platform enablers related to brokerage and federation. There are also various technology enablers lower in the cloud value chain (e.g. SDN), but given the focus on the EASI clouds project, this report does not examine them in detail. Sizing the market for platform enablers in this context is challenging because the revenue they create is predominantly realized in other parts of the value chain, of market segments are too emergent for existing analyses to cover them. For example, OSS is monetized by either selling services that the OSS software enables (e.g. IaaS) or selling a diverse range of related consulting services. Direct licensing revenue is also only partially available. As individual exceptions from the virtualization market, VMware's license revenues are approximately $2.3Bn which is less than half of its total revenue[129]. Citrix, which also focuses on virtualization, reported $891M license revenues for 2013[130].

4.1.8 Infrastructure Providers

Data center operators (and related) primarily manage (and own) data centers. Operating data centers is commonly internalized by IaaS providers (e.g. Amazon, Microsoft, Google, Rackspace etc.), and telecommunications companies and IT service firms represent a major groups that are strong players in this area in addition to several specialized 'carrier-neutral' firms (e.g. Telecity, Centurylink, Interxion, Zenium, Equinix).

While data center operations and development are hotbeds of innovation both in terms of technology and business models[56] (e.g. SDN, SDDC, bare-metal clouds, total hardware

[56] Data Center Knowledge (http://www.datacenterknowledge.com/) is one of many news sites following recent developments.

solution providers etc.), we review this value chain position in less detail, because data center operators are not core to the EASI CLOUDS project. Estimating a market size for data center operators is difficult as data center assets are typically monetized completely of partially by offering higher-level services in the cloud value chain. Colocation services form an exception, and Research and Markets estimates this market to be $26Bn with expected 11% CAGR for the upcoming years[131].

A related value chain position is *data center real estate services* that includes providing data center facilities to their customers, but do not manage the hardware inside the datacenters. Verizon Terremark is an example of a company that has its roots in real estate, but has gradually evolved into operating data centers. Digital Reality Trust, Dupont Fabros, CyrusOne, and CoreSite Reality represent major data center real estate investment trusts that rent data centers to CSPs that prefer not to get involved in real estate ownership.

Network service providers (e.g. telcos, ISPs) provide various data communications services to their customers including CSPs. Gartner values this telecom services market in 2013 at $1600Bn in 2013 with expected growth of 2.1% for 2014 and 3.7% for the following year[132]. As special segment of network service providers are *virtual network service providers* that do not own the necessary communications infrastructure, but rent of lease it from network service providers. An important function related to network service providers is *network exchanges* that interconnect different networks. This gives CSPs the ability to effectively transfer data between data centers and customers, which is also critical for brokerage or federation-based offerings. Equinix is an example of a company that both operates data centers and provides a vast range of interconnection capabilities.

In addition to telcos, *dark fiber lessors* own physical installed communications cables, but do not provide other communications infrastructure needed to transfer data over the cables. Instead, they sell rights to use the cables to network operators. Typical dark fiber lessors include telcos leasing fiber to other telcos (due to competitive regulation), and cities and municipalities. Data communications infrastructure is a valuable resource with limited supply that puts boundaries on entry opportunities in the otherwise largely fluid public cloud market. Specifically, the availability of communications infrastructure is an important factor when considering the viability of cloud federations, as information needs to flow effectively between the members of the federation and the customers of the federation.

4.2 Pricing and Revenue Sharing in Cloud Computing

4.2.1 State of the Art in Pricing Strategies in Cloud Computing

There is not only one given definition of the accounting and billing process. These definitions rely on the semantics and terminology used by the authors or the creators of accounting research and systems, respectively. Thus, in [13] accounting is defined as a meta-concept that involves several functions such as pricing, billing, etc. The accounting process includes multiple functions related. As we explain in Figure 4.4, the overall process is based on relations between different functions. Each of these functions involves a specific task. Metering records generated by the Metering entity are used by the Mediation entity to build the accounting records. Accounting is twofold: first it gives an input to the pricing entity, and second it has to communicate with the roaming entity to get the information of a local session

running at another CSP. In [2], we define the following sequence of all activities in the whole process, as follows:

- **Metering**: Collects information related to the resource usage in the form of metering records. This information is used during the entire accounting process.
- **Mediation**: Generates a homogeneous data format (*accounting records*) from *metering records* provided by the *Metering* function. *Accounting records* can be used for storing and further processing.
- **Accounting**: Filters, collects and aggregates the information that represents a resource usage by a certain consumer *into session records*.
- **Roaming**: Keeps accounting of resource usage while clients' jobs are operating on different CSPs.
- **Pricing**: Gives a price to a resource usage provided by the *accounting* function. The price may be calculated using different economic models (auctions, flat-rate, usage-based).
- **Charging**: Calculates the cost of a resource usage by applying the *pricing* function.
- **Billing**: Summarizes the *charge records* provided by the *charging* function into monetary units. The billing process is dedicated for the generation of a regular bill after the aggregation of event data, calculation of charges. Billing aims to convert the theoretical model represented by Billing by an invoice where the unit is a real currency (for instance in Euros). This shows how Accounting serves as an input for Pricing and Pricing serves as an input for Billing.
- **Clearing**: Specifies how the payment is done (i.e. paper check, electronic payment, credit transfer).

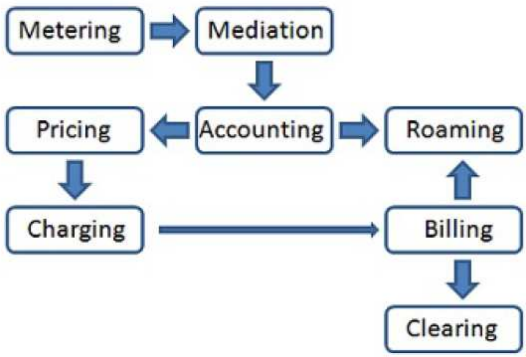

Figure 4.4: Accounting process

Pricing in Cloud computing is an important step in the accounting process for both CSPs and their customers. It is different from one CSP to another.

4.2.2 Pricing Types

Three pricing types are presented in Figure 4.5. We differentiate between two main types: Dynamic Price and Static Price. On demand (OD) and Reserved (R) instances are two examples of fixed price for resources. Spot (S) is an example for the dynamic price. For the moment, it is the case of OD, R, and S.

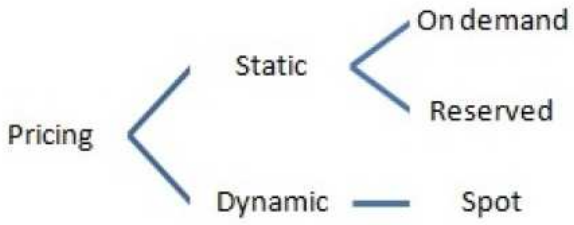

Figure 4.5: Pricing types

- **Fixed price**: It is the simplest pricing scheme, which fixes all prices for the whole time horizon. As the prices are fixed, the optimization in this pricing scheme is done only once. This is the main drawback of this scheme. This scheme gives option for pay as-you-go. In pay as you go scheme, the user pays per query and has to pay only for how much resources are used and proportionally to time usage.
    - ***On-demand***: The requests are purchased on the fly. The prices of the VMs are fixed and inspired from Amazon instances prices.
    - ***Reserved***: It refers to the fact the end-user pays a reservation fee in advance, and in return gets a discount on the usage of the VMs. This plan assures the availability of resources when they are requested to be used.

- **Dynamic pricing**: It emerges as an attractive strategy to better cope with unpredictable customer demand. Figure 4.6 illustrates the service provider's loss when using static pricing. There can be two cases. First, under-demand where the supply is more than demand and fixed price is likely to be higher than the price in the market in which case users will look for alternative resources. Second, over-demand where supply is less than demand and fixed price is likely to be less than the price in the market leading to service provider's loss.
    - ***Spot***: It refers to the fact the end-user proposes a bid to the CSP he is ready to pay for the provisioning of a given set of resources (CPU, RAM, and disk). The spot prices are supposed to change on hourly basis. A spot request is accepted only if the value of the bid is greater or equal to the spot price.

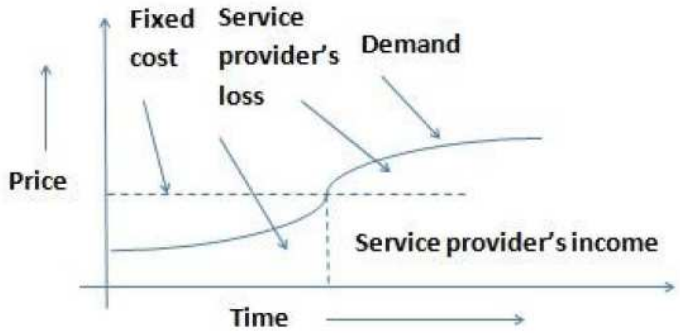

Figure 4.6: Static vs. Dynamic price

4.2.3 Pricing Models

Three different pricing models are presented in Figure 4.7.

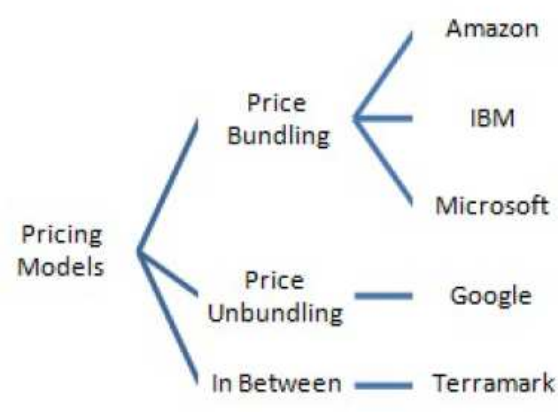

Figure 4.7: Pricing models

- **Price bundling**: the users choose a discrete set of predefined packages (e.g. "Gold instance" with 32 GB of Memory, 1 TB storage, and 4 CPUs for 300 $ per month as well as "Silver instance", etc.). Amazon, IBM and Microsoft are using this pricing type.
- **Price unbundling**: the users are charged specific price per unit (e.g. 60 $ per CPU or 20 $ per 100 GB of storage) per month. Google is using this pricing type.
- **In between:** customers choose their storage requirements but they have bundles of CPUs and memory. Terramark for example uses this pricing type.

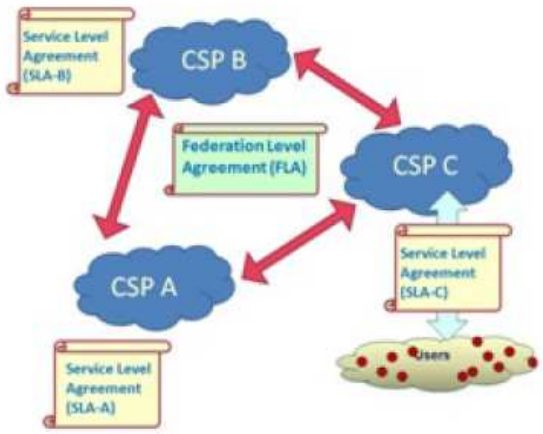

Figure 4.8: Cloud federation

4.2.4 Existing Strategies

In [17], authors describe a bid based pricing policy, which managed two main constraints: the budget of the cloud customers and the completion time of a given job. Authors in [3] use pay as you go pricing policy of Amazon, and introduce the pricing fairness both personal and social. Dynamic auction based resources pricing are presented in [15] which are theoretical or

simulation based. User Welfare is examined in [10] to evaluate a dynamic reverse auction pricing scheme on federated clouds. Genetic Model that explain how to deal with pricing parameters and maximize the utility in cloud computing is discussed in [9]. In [7], an iterative algorithm is used to update the price. The algorithm analyses the historical utilization ratio of the resource, iterates the current prices constantly, gets the availability of resources next time, and calculates the final expected price to the users.

4.2.5 Revenue Sharing in a Federated Cloud

In a Federated environment, a CSP may benefit from quasi-unlimited resources belonging to other CSPs. Meanwhile, such collaboration among distinct CSPs to provide a common service to their customers necessitates a priori a form of business agreement among them. We assume that all the members of the federation provide the same type of hardware and software resources to their clients. In other terms, Cloud federation seems less justified in the context of CSPs offering very specific services that require very different types of resources. Each CSP can use the resources of other CSPs to serve its own clients when its own resources are not enough to serve the demand. Reciprocally, CSPs will be able to sell their unused capacity for other CSPs. From this definition, CSPs could buy resources in the federation in order to serve the upcoming requests that go beyond one CSP capacity. Also, by selling resources, CSPs will increase their revenues. In this matter, the problem of revenue sharing should be examined and analysed.

In a Cloud Federation, cloud bursting is provided via the interoperability between independent CSPs in a transparent way for the end-users, as it is illustrated in Figure 4.8. Cloud Federation assumes that each CSP member of the federation agrees on a common Federation Level Agreement (FLA). In parallel, each CSP of the federation provides services to its own clients via its specific Service Level Agreement. The members of the federation agree to share resources upon defined pricing policies.

4.2.6 Existing Strategies for Revenue Sharing in a Federation

Multiple revenue sharing techniques exists in the economic field [4]. The Shapely Value [16] is one of the traditional techniques used for revenue sharing between multiple players. In the proportional share as its name says, each player gets proportional revenue depending on its own contribution to the coalition. Each CSP is a player and has its own characteristics and clients. In a Federated context, multiple CSPs cooperate in order to satisfy each other's clients when needed. The problem of revenue sharing among the different members is a main issue because it determines how much each one gets when participating to the coalition, and this determines if it is profitable or not for one CSP to work with other CSPs or independently. In [14], the sharing problem in the federation is modelled as a repeated game between the CSPs, considered as selfish players. CSPs tend to increase their own profit in such scenario by selling their unused capacity in the spot market. The scenario is based on a central entity managing the federation.

A game theoretical approach is used for resource and revenue sharing in [11]. A stochastic linear programming game taking into account the demand uncertainty of internal users is the solution for the cooperation problem in [8]. The concepts of core [12] and Shapley value [6] from cooperative game theory are applied for revenue sharing in the cooperative mobile cloud coalition. In our previous work [1], we investigate the economical advantage that a federation can provide. We analyse multiple federation scenarios changing the sessions types (On-demand, Spot, Reserved) that can be outsourced based on a simple revenue sharing algorithm

that gives a constant percentage of an outsourcing job for the home CSP and the rest for the outsourcing CSP. In [5], we studied multiple revenue sharing methods. We considered one scenario changing the prices and the capacities of the CSPs.

4.3 Research on Pricing Models

Dynamic pricing schemes have been investigated with respect to different objectives, ranging from determining price equilibria, achieving stability, controlling demand, and maximizing profit [133]–[140]. All of these pricing schemes can be applied to cloud resource pricing.

Masuda and Whang studied pricing mechanisms for networks, maximizing the net utility of a network [135]. In particular, they characterized the equilibrium and the stability conditions of three dynamic pricing schemes.

Altmann et al. investigated the demand change under different pricing schemes within an empirical study [139]. They compared the differences in demand for Internet access service with respect to different usage-based pricing schemes and the flat rate pricing scheme. In a subsequent work, a decision support tool for helping Internet service providers (ISPs) to analyse pricing schemes, called Dynamic Netvalue Analyzer, was introduced by Altmann and Rhodes [138].

In a simulation conducted by Kephart et al., intelligent agents called pricebots automatically adapt product prices to changing market conditions, using a dynamic pricing algorithm [134]. In addition to this, they investigated shopbots. Shopbots respond to prices set by pricebots. They compare product prices, rank products based on customer preferences, and then make purchase decisions.

To capture the fact that perfect market knowledge is almost impossible to obtain in the real world, vendors with limited knowledge about customers and competitors can implement the derivative-follower pricing scheme[134]. Using this pricing scheme, vendor agents incrementally increase (or decrease) prices until they experience a drop in profit. Then, vendor agents decrease (or increase) prices. Dasgupta and Das developed a simulation that focuses on the derivative-follower pricing algorithm [137]. They proposed an algorithm that enhances the derivative-follower pricing by re-estimating the price-profit relationship for vendor agents for each time period. This work has been extended by Dasgupta and Hashimoto, who applied collaborative filtering for analysing customer preferences. Their algorithm uses the result of the collaborative filtering as input to a dynamic pricing algorithm, which calculates the profit-maximizing price [136].

In another research paper by Lehmann and Buxmann, a pricing algorithm uses knowledge about customer preferences [133]. Because of that, the algorithm, which is referred to as a demand-driven pricing algorithm, achieves profit maximization. Precisely identifying customer preferences becomes a critical success factor for the demand-driven pricing algorithm [140]. However, acquiring perfect information of customer preferences requires huge amount of resources. These resources are usually only available to large firms.

In order to attract customers with low reservation prices, the penetration-pricing scheme can be implemented. This pricing scheme initially sets prices lower than the prices of competitors, and then gradually increasing price [133]. The objective is to benefit from lock-in effects, i.e., to benefit from customers whose cost for switching to a new provider is high. Penetration

pricing scheme is aimed at customers with high price sensitivity [141].

The polar opposite of penetration pricing is called skimming pricing. A skimming pricing scheme targets customers with high reservation prices or inelastic demand [141]. This pricing scheme creates high prices for products that are released newly, and then continuously lowers prices to capture customers with lower reservation prices [133]. High- tech vendors or innovative software vendors use this pricing scheme to sell software.

5 EASI-CLOUDS Innovation

5.1 Real-time Billing as a Service

Looking at how today services and applications are offered in an increasingly interconnected world, one can find that there is a trend towards a variety of services and applications developed for very specific tasks[142]. Instead of providing large, monolithic applications, complex applications are stitched together based on interoperable services, which is known as service orchestration in the Service Oriented Architecture (SOA). Several electronic service marketplaces now offer customers the possibility to build their own, tailored applications (e.g., Logistics Mall[57], goBerlin[58], SAP Service Marketplace[59]).

This trend is supported by the proliferation of cloud computing, which allows, on the basis of scalable infrastructures (Infrastructure as a Service, IaaS) and well-equipped platforms (Platform as a Service, PaaS), to provide specific applications and services as Software as a Service (SaaS).

Together with a vast number of specific services and applications also a large number of charging models emerge, similar to those are already common in the telecommunications market (see 3.5.1.5). Especially usage-based charging models are suitable. They allow charging each individual service use, even depending on the amount of data transferred or the desired and provided quality of a result, where appropriate. Because cloud services are often used ad hoc, it is more than helpful if the anticipated costs can be determined before the service is actually used and the incurred costs are directly visible after or even already during the service usage. Such functionalities can usually be provided by a real-time billing system.

Real-time billing systems are usually large, sophisticated commercial systems with a high purchase price and require a great installation and maintenance effort. Providers of specific services and applications, however, are usually small and medium-sized supplier companies, which cannot afford to purchase and operate their own real-time billing system.

To overcome this problem, an approach to offer real-time billing functionality based on the SaaS principle has been prototypically developed in the EASI-CLOUDS project.

The SaaS principle provides the following main benefits:

- Opportunity to reduce capital cost by elimination of on-premises installations and thereby savings on hardware, software licensing fees, support and operational management
- Usually low start up/first year cost
- Agility - often short time to start up of services
- Flexible usage and scalability – easy to adjust per need and demand
- Boundless availability – over internet
- No software monitoring obligations
- Allows organizations to focus on core skills
- Enables easier expansion to new areas/opportunities/verticals

[57] http://mmp.logistics-mall.com
[58] http://www.cloudwatchhub.eu/node/48
[59] https://websmp109.sap-ag.de/public/tio

This allows for flexible usage-based billing in real-time without the need of large investments in an own powerful billing system.

5.1.1 Existing Billing as a Service Offers

Looking at commercial BaaS solutions one can already find already several providers. But when looking at the different offerings it becomes clear that these offerings are often based on different definitions of BaaS: Some providers cover the entire processing and value chain, e.g., in the utility domain this regards all processes from meter readings via rating and charging to invoicing, payment tracking, collection, and fulfilment. Within EASI-CLOUDS we have considered BaaS in the cloud-computing domain, which is a software service offered as part of the SaaS market. Billing services are essential to e-commerce sites as well as for the monetization of SaaS and cloud software and infrastructure providers.

Evaluating the current BaaS offers reveals a set of important features that are commonly listed:

- Ease of use
- No need to maintain on-site hardware or software
- Scalability
- Support of low-risk business model (Pay as you go)
- Support of multiple currencies and languages
- Support for multiple tax schemes

From the EASI-CLOUDS point of view there are some additional important features to be considered in order to be successful in the cloud computing domain: One such feature is the use of open interfaces and standards, supporting an easy integration of the offered billing services into existing platforms and services. In this context, some research activities can be observed. For example, the research initiative FI-WARE is building an open cloud-based infrastructure for cost-effective creation and delivery of future Internet applications and services. It has integrated support for pricing, accounting, charging, billing, and revenue sharing models in the so-called FI-WARE Store Generic Enabler [143]. Just as also proposed in the EASI-CLOUDS approach, FI-WARE intends to expose such functionality as REST APIs based on open interface specifications. However, at the time of writing, the corresponding APIs are not yet published[144]. Another example is the Mobile Cloud Networking (MCN) research project that investigates emerging technologies, synergies, and integration potential of cloud services and mobile communication infrastructures. The project proposes a component called “Rating, Charging, and Billing as a Service” (RCBaaS) that supports other MCN services to perform rating, charging, and billing for both end users and service providers[145]. The interfaces (or: reference points) of the RCBaaS heavily rely on charging-related standards and architectural concepts of the 3GPP [146], [147].

Another important feature is the support of real-time rating and charging. For real-time charging, a dedicated Real-time Charging System determines already before and during resource or service usage whether access to the desired network resources and services is granted or not. Real-time rating even provides the ability to determine the price right before service usage without really applying the charges. This functionality can be used to check whether service usage is allowed and/or wanted for the indicated price.

In the past, real-time rating and charging was mainly important for prepaid systems to guarantee that customers do not overdraw their account while using resources or services. But nowadays cost and price transparency is also very important for non-prepaid systems. Real-time capabilities allow providers to see their current revenues at any time and to evaluate whether the applied pricing models are appropriate or not. Only if providers are able to see their costs and revenues in real-time they will be able to detect unfavourable tariffs early enough to avoid loss of revenues.

In the following sections the currently most widely known existing BaaS solutions are shortly described. But as far as could be evaluated during the EASI-CLOUDS project, none of them has real-time support *and* uses open interfaces or standards.

Redknee
Redknee's[60] cloud-based converged billing and customer care solution offers a platform supporting the entire subscriber lifecycle with end-to-end functionality including self-activation, Web self-care, dealer portal and customer care support. The solution is available as a SaaS offering and Redknee states that their solution is easy to use, scalable, supports Pay as you go and no on-site hardware or software support is needed. Redknee offers real-time charging support, but does not seem to use open interfaces or standards.

SoftCom
SoftCom[61] offers resellers a flexible and customized solution that enables them to outsource the entire billing and provisioning components of their service delivery platforms. A reseller is charged a monthly wholesale fee for all services sold under their reseller account and has the ability to earn additional discounts based on overall aggregate sales volumes. Supported features are an easy and fast service deployment process (i.e., in weeks, not months), and multiple currencies and languages. In addition they integrate with third party SaaS applications and third party billing and provisioning platforms, but in EASI-CLOUDS we are not aware if this is supported by the use of standardized interfaces or not.

Cerillion
Cerillion Skyline[62] offers a high-performance billing engine that supports a wide range of subscription billing features including support for multiple tax schemes to calculate taxes under different geographic rules or according to the customer type. In addition, Skyline offers a set of standard invoice templates, which can be re-branded according to local business requirements, as well as offering an invoice design service for customizing invoice layouts. As far as we are aware in EASI-CLOUDS, Skyline does not offer any real-time rating or charging features and does not support any open interfaces or standards.

Tieto
Tieto[63] offers a BaaS solution for utility companies. Their solution covers the entire value chain from meter readings to invoicing, payment tracking, and collection. The features for customers are a lower risk and less investment and thereby lower costs through more efficient services and scalability. In EASI-CLOUDS we are not aware if the Tieto solution supports real-time rating and charging or open interfaces and standards.

[60] http://www.redknee.com/
[61] http://softcom.com/
[62] http://www.cerillionskyline.com/
[63] http://www.tieto.com/

5.1.2 BaaS - Results and Benefits

For service providers, access to some kind of billing system is crucial in order to collect charges for their service(s) [148]. But billing, or to be more precise, rating, charging and billing of services is very often underestimated in terms of complexity as well as performance. Various pricing models are generally applicable to all kinds of services. Especially the real-time aspect becomes more and more important in order to enable these increasingly dynamic and sophisticated combined pricing models.

Many service providers do not have an own billing system. On the one hand, they would like to spare the efforts and costs for its hosting, service and maintenance. On the other hand, they do not want to abandon the amenities of a professional billing system and demand standardized interfaces.

Considering the cloud as an example for a fast growing market, especially when thinking about federated clouds[149], where resources and services are yet shared between different clouds and cloud environments, also information about the availability and characteristics of resources and services as well as their usage costs need to be exchanged between the different cloud environments. As an integration of different cloud environments into a federation as well as the data transfer between these environments needs to be easy and effective/sustainable, the authors suggest relying on open and standardized interfaces and data models. But according to a comprehensive study from the German Federal Ministry of Economics and Technology, there are currently no open standards available for cloud environments that support real-time rating and charging [106].

The BaaS approach developed in the EASI-CLOUDS project is a real-time billing service that can be booked and used by the service providers themselves. This can, e.g., be done by a SaaS via a Cloud Marketplace. The BaaS approach allows service providers to create, configure and deploy their own pricing models. The configured pricing models are then exported in the standard format of the Unified Service Description Language (USDL)[150].

Easy and flexible integration of real-time rating, charging and billing functionality into the service providers' own services is ensured by providing a standardized API, covering all the required functionality for Customer Profile Management, Account Management and Payment. This functionality is provided via Web Service interfaces based on REST Web Services and is compliant with the GSMA OneAPI[107] standard.

The following list of results and benefits points out the innovative character of the real-time Billing as a Service approach developed in EASI-CLOUDS:

- Configuration of price plans via Web-GUI in a simple way
- Customized API for easy and flexible integration into services
- Amenities of a professional billing system without the efforts and costs for its hosting, service and maintenance
- Standardized interfaces for integration with already existing billing systems
- Operation of a billing system in a federated hybrid cloud system to benefit from scalability and flexibility
- Billing system configuration with cloud services
- Billing as a Service (BaaS) as a managed service for offers in a federated cloud environment
- Standardized interfaces for easy billing system integration and service portability
- Sophisticated mediation tools

- Multi-tenancy capability
- Revenue sharing support
- Facilities for efficient business models (pay-per-use, on-demand, etc.)
- Support for sale to new market segments

5.2 Cloud Federation

EASI-CLOUDS has proved the feasibility of the Cloud Federation concept, based on decentralized and homogeneous broker architecture. EASI-CLOUDS has chosen the ACCORDS platform (Output of CompatibleOne project) as a core component of the EASI-CLOUDS project to address the broker and federation Challenges.

During the EASI-CLOUDS project, ACCORDS Cloud Broker has been extend to support a decentralized Federation of ACCORDS Platforms (Declaration of an ACCORDS platform as a provider of another ACCORDS platform) with new features such as:

- Federation Management - Creation and deployment
- Federation Members Management
- Management of resources shared in the Federation
- Service Provisioning in a Federation Environment

A Federation use case has been implemented in order to show a real Business Case:

- Four cloud provider companies wishing to federate their heterogeneous service offers in order to increase their offer (EC2, CloudSigma, OpenStack, CloudFoundry).
- Provisioning of four ACCORDS platforms, and Federation grounding (i.e. declare other ACCORDS platforms as providers)
- Provisioning of service targeting the right provider of the federation

In addition ACCORDS Cloud Broker has been extended to enhance its brokering capabilities through:

- More types of Cloud supported by ACCORDS (IaaS, PaaS, SaaS),
    - IaaS: more types of IaaS provider supported (vCloud, IaaS++)
    - PaaS: Two levels of PaaS' features are supported:
        - Provisioning of Services on PaaS Platforms (Accords connectors for OpenShift, CloudFoundry)
        - Accords as service provider for CloudFoundry.
    - SaaS: Provisioning of SaaS services through Accords.
- Service registry features, to manage Service Template, Agreement and Service Instance.
- Integration of Accords platform within a Marketplace based on Liferay, and supporting USDL Service Description, IAM and Billing Components
- Implementation of demonstrators leveraging these new features (Free Surfer Service with SLA negotiation, GPES Service with Specific Smart Placement algorithms, Photostiching SaaS Service provisioning, PaaS Continuous Integration).

Despite the project achievements, some challenges have not been addressed, such as:

- Federation Agreements (SLA Definition & Evolution, Customization, Monitoring, Adaptation and Governance)
- Procurement process (Cloud Supplier Selection and Assessment) due to the lack of methods for comparison and assessment of Cloud Resources or Cloud Service Suppliers.
    - Offer Comparison: Price, Service, Quality
    - Sustainability and respectability of suppliers

5.3 SLA Negotiation

Automated Service Level Agreement (SLA) negotiation is a topic that is still in a quite early stage of research. So far, most commercial cloud offerings rely on a model similar to end-user license agreements commonly found with software packages, which are based on the principle that a customer can either accept the provider's terms and use the offering, or disagree with them, which in consequence means he cannot use the offering. This model can be extended in a way to be a bit more dynamic and user-friendly by offering several levels of service for different prices, such as different levels of reliability or computing power. However, these models are still a far step away from automated negotiation of SLA terms as intended for the EASI-CLOUDS project.

To be able to provide automated SLA negotiation, it is necessary to be able to express service descriptions, guarantee terms, resource usage, pricing, and everything else related to service usage in a formal manner. Currently, there are several major document formats available for this purpose.

5.3.1 SLA Document Models

- **WS-Agreement**: WS-Agreement is a standard by the Global Grid Forum from 2005, which models SLA guarantees and parameters and stores information relevant for SLA monitoring. It was created with the intent of providing automated SLA negotiation and monitoring and thus provides a detailed modelling of the SLA terms. However, other aspects of software usage are completely ignored by WS-Agreement, which requires it to be used in combination with other service description languages.

- **USDL**: the **U**nified **S**ervice **D**escription **L**anguage was an important attempt to unify the various aspects of service description (technical, commercial, legal, etc.…) into a common document. USDLs attempt of providing this is a structure of several modules that each provides a model for one of these aspects. In the EASI-CLOUDS project, USDL was chosen to describe services and SLAs, as the single-document approach of it seemed quite well suited for the task of exchanging documents between the architecture components. However, it was found out that custom extensions were required to provide detailed descriptions of service resources required for provisioning services. Furthermore, USDL was originally not intended for SLA negotiation, thus more extensions were required to mark SLA parameters as negotiable and provide allowed value ranges. To be able to use the existing WSAG4J framework as a base for the EASI-CLOUDS component implementation, a combination of WS-Agreement and

USDL document is currently employed in EASI-CLOUDS. A great weakness of USDL that showed in the project is the lack of semantic expressions in USDL, which makes it quite hard to link the different aspects of service usage to each other.

- **Linked-USDL** [151]: As the lack of semantic expressions in USDL showed in a lot of projects attempting to use the standard, a new version called Linked-USDL was proposed. Linked-USDL attempts to overcome USDL's limitations by using the semantic notation RDF as a base. There were intentions of using Linked-USDL for EASI-CLOUDS in the planning stage of the project, but as by then the standard was still being under development, the combination of USDL and WS-Agreement was chosen instead. A core release of the standard was finally published in June 2014, and should be investigated for future implementations of SLA negotiation and management.

5.3.2 SLA Negotiation

Once the service description and SLA terms are expressed in a formal model, a toolchain for (at least partially) automated SLA negotiation is required. Although the final decision authority will in all cases rest on human representatives of customer and provider, an automated negotiation of SLA terms between customer's needs and provider offerings can make cloud usage a lot easier and expand the use cases for cloud services. The current lack of an industry standard toolkit is closely related to the problem that no standard for service and SLA description has yet been adopted widely enough to become a standard. Furthermore, current large players in the cloud market tend to opt for in-house solutions rather than open standards, which may be attributed to the lack of established standards, but as well a way of making provider changes less attractive for their customers.

The EASI-CLOUDS architecture employs the **WSAG4J** framework [152] as a base for SLA negotiation. WSAG4J is built for the WS-Agreement standard. The automated negotiation capabilities it provides use the standard technologies available for XML datatype modelling. This means that it is able to perform checks if SLA term values are valid and within allowed ranges. Further negotiation abilities have to be added by developers using the WSAG4J framework. In general, the problem of a generic automated SLA negotiation very much depends on semantic expressions for service descriptions and SLA terms. As WS-Agreement does not offer these abilities, most of the negotiation still has to be implemented by framework users.

The lack of a generic framework for automated SLA negotiation widely known across the research community. Wu et al [153] proposed an approach using a knowledge base and negotiation policies to overcome the lack of semantic expressions, and a heuristic approach to automatically negotiate QoS parameters between a SaaS broker and a larger number of providers. The paper describes that the authors implemented a test framework and tested it using values gathered from real-world cloud offerings, however it is unclear if the authors plan to further develop their framework for SLA negotiation, or if their approach is applicable to scenarios with a more general approach than a small set of selected QoS parameters is.

5.3.3 SLA Monitoring and Policy Enforcement

The processes of Negotiation and Agreement of SLA guarantees would be useless without means to control that guarantees are fulfilled during a service's lifetime. In order to ensure that SLA guarantees are fulfilled, it is thus necessary that 1) the state of a service is evaluated against the SLA guarantees at runtime (SLA Monitoring), and 2) actions can be applied on guarantee violations, e.g. financial penalties or compensation actions to return the service to an allowed state (policy enforcement). The term "SLA Monitoring" is in this context used to describe the process of monitoring service states and evaluating them against SLA guarantees, as the literal meaning (monitoring the SLA documents) does not make much sense.

Current cloud offerings do not provide much in regards to SLA monitoring. Instead, they operate more or less in a way that customers have to trust providers that SLA guarantees are fulfilled, there is not much transparency offered by providers to show how the guarantees were kept. If a user has the suspicion that the provider is not fulfilling the guarantees, it might be hard to prove this, as most cloud providers only grant very limited access to monitoring data.

Again, a generic approach towards SLA monitoring depends a lot on the formal description of services and SLA guarantees. As a monitoring framework needs to collect runtime data from a service, the descriptions of SLA guarantees need to contain information about which context variables of the service should be evaluated in what fashion to judge if a guarantee has been violated. In consequence, a formal description of the context variables of a service that can be monitored is also required.

USDL contains definitions of context variables for services and a simple classification system that provides at least basic information how to interpret a context variable. The extensions to USDL made for EASI-CLOUDS allow to reference context variables in SLA guarantees, so that state variables can be evaluated directly in a generic fashion. USDL offers the definition of threshold values and a comparator function (equal, smaller/larger than) for a guarantee, which allows the evaluation of a context variable's value. However, this is largely dependent on a service implementation, as it relies on the service to expose the declared context variables to a monitoring system. Also, this approach is not feasible for guarantees that require evaluation over longer periods of time, e.g. an availability percentage value for a service. In the EASI-CLOUDS architecture, this problem can be partially solved by the concept of a ContextStore, which is a component that accumulates monitoring data from a service, performs context-specific calculations and exposes the resulting values again via monitoring. As the name indicates, the ContextStore needs to be implemented specially for each context.

To be able to perform such evaluations without the need to implement context-specific / service-specific components, a more meaningful semantic description of context variables is required than what USDL offers, furthermore, the description of how a guarantee state can be evaluated needs to be able to express more complex relations than simple threshold values. As **Linked-USDL** is based on RDF to be able to contain semantic information, thus future use of Linked-USDL could be a step towards a more generic approach on SLA monitoring.
SLA Policy Enforcement for specific domains / contexts has been a topic of research, such as the M.F.Bari et al. did for the domain of Software-defined Networks (SDNs) [154]. Most of these concepts and frameworks solve the problem of policy enforcement for the specific domain or context in a quite elegant way, by applying heuristic models, approaches from game theory or AI algorithms for negotiation between autonomous agents on the problem.

However, these approaches rely on a context-specific implementation of the policy enforcement algorithms. A generic approach is again dependent on the availability of semantic descriptions of services, policies and possible actions.

5.4 SaaS Enablement of Legacy Applications

The transfer of applications that are already web-based into a cloud is usually quite straightforward: The server landscape is rebuilt in the virtual data-center of the cloud provider and then works like on traditional servers. However, our challenge is to turn software that is not web enabled and not cloud-aware into a cloud service.

FreeSurfer, an open source software suite for processing and analysing brain MRI images, provided an excellent use-case to work on this problem. FreeSurfer itself consists of a set of command-line tools, which are orchestrated through shell scripts, and is not in itself cloud-aware.

5.4.1 The Challenge

In order to create a cloud service this tooling has to be integrated to become cloud-compatible for the control and monitoring mechanism for a cloud service. The service itself has to be designed to be scalable, and deployable, since in a cloud there is no fixed installation in general but only virtual resources that are created and destroyed after use.

5.4.2 The FreeSurfer Cloud Service

The FreeSurfer cloud service, depicted in Figure 5.1, is designed as a worker-based solution. The service end-point of FreeSurfer in the cloud internally acts as a controller node for a group of worker nodes. Using a REST interface, the controller takes jobs, which consist of an arbitrary number of tasks. It shares the work by calling up a number of worker nodes and distributing the tasks to them. Each worker node will then work on the tasks given to it, sequentially, until all tasks are completed.

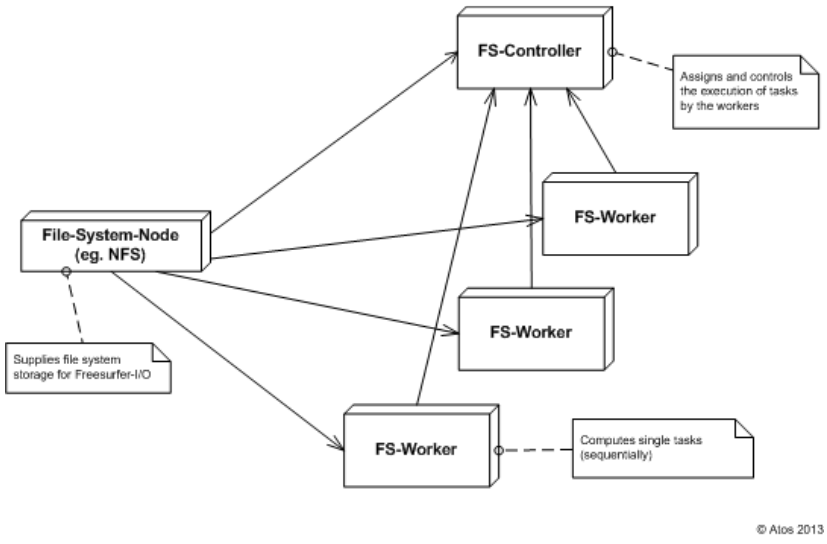

Figure 5.1:FreeSurfer deployment setup

Due to the nature of FreeSurfer, scaling can currently not go beyond task level, since it is not a multi-core application. This also influences what virtual resources are necessary.

In order to encapsulate the FreeSurfer tooling, a generic integration approach has been chosen: In order to capture the sequence of individual tools to be called, the behaviour of the tooling has been captured as a finite state machine. For the execution control of legacy (command-line) tools exists a generic class library, which is then extended with regard to the FreeSurfer specifics. With each transition in the finite state machine, specific context events can be generated. In our case these events report the progress of the single worker to the controller. The controller can process these context events into a summary and forward it to the cloud service manager, which may decide to allocate more resources, or free them.

Figure 5.2: Architecture of a FreeSurfer node

This integration layer has been realized using OSGi as modular platform, which integrates the services as bundles and allows a modular extension of the layer. Figure 5.2 shows a simplified view of a FreeSurfer node in the cloud. The ContextStore is used for service monitoring, while the REST interface is used for control.

6 Conclusion

In this document we have described some of the state of the art on different fields in the area of cloud computing. We have taken a view of the current cloud related business environment. Finally we have identified the key points of innovation that have been in the center of focus in the EASI-CLOUDS project.

EASI-CLOUDS has been about advancing the state of cloud computing in Europe, as well as in Egypt and Korea. It has also been about helping the entrance of new players in the fast growing European cloud market. To support these objectives, we have constructed a comprehensive, open-source, innovative, and validated cloud infrastructure that can be instantiated to set up an application type-specific cloud for use in a private, public, or hybrid setting.

The EASI-CLOUDS infrastructure has a powerful service composition and orchestration framework that facilitates cloud interoperability and federation, uses standardized interfaces to allow service portability, and supports advanced SLA (Service Level Agreement) management to help guarantee the required Quality of Service. It also provides facilities for capacity planning, provisioning, accounting, and billing that are needed to operate a true marketplace for cloud services and an efficient pay-per-use business model.

The EASI-CLOUDS project has proven the feasibility of cloud federation. The method chosen by the project is to use and extend the ACCORDS platform to suit the needs of brokering and federation. The ACCORDS platform was extended to support decentralized federation by enabling it to communicate with other ACCORDS platforms. The new features also include for example federation management, creation and resource management. The brokering capabilities were also enhanced, for example the support for different clouds environments and service registry features were added. Overall, the ACCORDS platform is capable of making cloud federation possible.

The previously available real-time billing services were often complex, sophisticated commercial systems that need to both tailored to specific systems and maintained. Due to this, they are often very expensive and smaller enterprises cannot afford them. The EASI-CLOUDS has developed a real-time Billing-as-a-Service approach, that is suitable for the cloud computing. The billing service can be made available for example as a SaaS service through a cloud marketplace and the service providers can customize the service to suit their pricing models and services. The integration of the real-time billing service is made through open APIs, which cover Customer Profile Management, Account Management and Payment functions.

Automated SLA Negotiation is an open research question. In the project, we have approached the problem by examining the current approaches and augmenting some of their solutions to suit the EASI-CLOUDS requirements. Our requirements include the need to be able to express service descriptions, resource usage and pricing in a formal manner that can be then automated. In addition, the actual SLA negotiation needs to be at least partially automated. The used framework presents a negotiated SLA to the customer who confirms the SLA. The framework can also take care of monitoring and enforcement of the SLA.

In the work with SaaS enablement of legacy applications, we have demonstrated the versatility of the framework. The work focused on converting a complex, computationally intensive, locally executed command-line application into a scalable, easy to use service that can be offered in a cloud software marketplace. Scalability was achieved through orchestrating instances of the application to be executed on a dynamic set of virtual machines, run in parallel, each working on an independent part from a collection of tasks that were assigned to the service.

In the EASI-CLOUDS project we were able to create an environment, which enables us to create a federated cloud ecosystem. The project also allows the developers to both create new applications and also enable legacy applications to be run in the cloud environment. We also examined the business- and revenue sharing models that the cloud domain brings to the companies. This allows different actors to benefit from the project work, be they service providers, businesses, application developers or investors.

The project was also successful in laying grounds for future investigation. In EASI-CLOUDS, we used USDL as the standard description language for resources, prices etc., and extended it for several cases where it was not originally designed to be used. The development of a newer standard, called Linked USDL, was underway at the same time as EASI-CLOUDS, and is worth revisiting for the same purposes for which we used USDL. We also encourage further investigation to solve challenges related to Federation Agreements, like best practices in forming and monitoring the SLAs between federation partners.

List of Figures

MIDEaaS Main Components 24
CoRED architecture 25
CoRED high level class architecture 26
Error line and marker in MIDEaaS 27
Schematic view of a federated hybrid clouds 33
USDL Overview 47
USDL modules and dependencies 48
Estimates of main cloud computing segments (excluding BPaaS). Source: Gartner[110] 52
Geographical distribution of the public cloud market according to IDC[116] 53
A simplified cloud value chain from a non-technical cloud consumer perspective with the focus of the EASI CLOUDS project highlighted. 56
Accounting process 61
Pricing types 62
Static vs. Dynamic price 62
Pricing models 63
Cloud federation 63
FreeSurfer deployment setup 75
Architecture of a FreeSurfer node 76

List of tables

Table 3.1: IaaS platforms available on the market 10
Table 3.2: Hypervisor-support by IaaS platform [11] 11
Table 3.3: Features of different hypervisors 12
Table 3.4: Storage services offered by different IaaS platforms 13
Table 3.5: Scheduling methods in IaaS platforms 14
Table 3.6: Supported interfaces in IaaS platforms 16
Table 3.7: Classification of different IaaS clouds smart placement algorithms. 20
Table 3.8: Comparison of on-line IDEs 28
Table 3.9: Cloud categories 32

Acronyms

AHP	Analytic Hierarchy Process
API	Application Programming Interface
BCR	Benefit to Cost Ratio
CAGR	Compound Annual Growth Rate
CLI	Command-Line Interface
CoRED	Collaborative Real-time Editor, the code editor used by MIDEaaS
CPU	Central Processing Unit, a generic term for a processor
CRUD	Create, Read, Update, Delete; Common operations for records in databases
CSB	Cloud Service Broker
CSP	Cloud Service Provider
CSS	Cascading Style Sheets
CSV	Comma Separated Values, a format for storing data
DPP	Discounted Payback Period
EC2	Elastic Compute Cloud, a service by Amazon
FLA	Federation Level Agreement
GRE	Generic Routing Encapsulation, a tunnelling protocol designed by Cisco
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure (HTTP over TLS/SSL)
I/O	Input/Output, generally writing or reading from a device such as storage
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IdM	Identity Management
IoT	Internet of Things
IP	Internet Protocol
JAR	Java ARchive, a container for distributing Java applications
JDK	Java Development Kit
KPI	Key Performance Indicator
KVM	Kernel-based Virtual Machines
LDAP	Lightweight Directory Access Protocol
LP	Linear Programming
LVM	Logical Volume Management
MCDM	Multiple Criteria Decision Making
MIDEaaS	Mobile IDE as a Service, a software development tool by TUT
NFS	Network File System
NIST	National Institute of Standards and Technology
NPV	Net Present Value
NRPE	Nagios Remote Plugin Executor, part of a remote monitoring system
OCCI	Open Cloud Computing Interface
OVF	Open Virtualization Format

PaaS	Platform as a Service
PAM	Pluggable Authentication Modules
PKI	Public Key Infrastructure
QoS	Quality of Service
RAM	Random Access Memory, the "working memory" of a computer
RDF	Resource Description Framework
REST	Representational State Transfer, software/interface architecture design
ROI	Return On Investment
RRD	Round-Robin Database, a format for storing time-series data
S3	Simple Storage Service (a service by Amazon)
SaaS	Software as a Service
SDK	Software Development Kit
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SSH	Secure Shell, a remote control protocol
SSL	Secure Sockets Layer, predecessor of TLS
SSO	Single Sign-On
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UML	Unified Modeling Language
USDL	Unified Service Description Language
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMFS	Virtual Machine File System
VMM	Virtual Machine Manager/Monitor
VNE	Virtual Network Embedding
XML	Extensible Markup Language

References

[1] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in *INFOCOM 2009, IEEE*, 2009, pp. 783–791.

[2] S. A. Baset, C. Tang, B.-C. Tak, and L. Wang, "Dissecting Open Source Cloud Evolution: An OpenStack Case Study," in *HotCloud*, 2013.

[3] "OpenStack Cloud Administration Guide." [Online]. Available: http://docs.openstack.org/admin-guide-cloud/.

[4] "OpenNebula 3.4 Guide / Virtualization," Apr-2012. [Online]. Available: http://archives.opennebula.org/documentation:archives:rel3.4:vmmg.

[5] "OpenNebula Ecosystem / Hyper-V Drivers." [Online]. Available: http://wiki.opennebula.org/ecosystem:hyperv.

[6] "Eucalyptus." [Online]. Available: https://www.eucalyptus.com. [Accessed: 29-Sep-2014].

[7] "VMware." [Online]. Available: http://www.vmware.com. [Accessed: 29-Sep-2014].

[8] "Nimbus." [Online]. Available: http://www.nimbusproject.org. [Accessed: 29-Sep-2014].

[9] "Apache Cloudstack." [Online]. Available: http://cloudstack.apache.org. [Accessed: 29-Sep-2014].

[10] "Citrix CloudPlatform | CloudStack Consultancy & CloudStack Support from ShapeBlue." .

[11] "Comparison of platform virtualization software," *Wikipedia*. 03-Jul-2014.

[12] "OpenStack Glance Documentation." [Online]. Available: http://docs.openstack.org/developer/glance/.

[13] "OpenStack Components / Storage." [Online]. Available: http://www.openstack.org/software/openstack-storage/.

[14] "Amazon S3." [Online]. Available: http://aws.amazon.com/s3/. [Accessed: 30-Sep-2014].

[15] "Ceph." [Online]. Available: http://ceph.com/. [Accessed: 29-Sep-2014].

[16] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, *Internet Small Computer Systems Interface (iSCSI)*. IETF, 2004.

[17] "EBS - Elastic Block Storage." [Online]. Available: http://aws.amazon.com/ebs/. [Accessed: 30-Sep-2014].

[18] "OpenStack Security Guide." [Online]. Available: http://docs.openstack.org/security-guide/.

[19] "Open vSwitch." [Online]. Available: http://www.openvswitch.org. [Accessed: 29-Sep-2014].

[20] "OpenStack Configuration Reference - Icehouse." [Online]. Available: http://docs.openstack.org/icehouse/config-reference/.

[21] "Ceilometer." [Online]. Available: https://wiki.openstack.org/wiki/Ceilometer. [Accessed: 30-Sep-2014].

[22] "übersmith." [Online]. Available: http://www.ubersmith.com/. [Accessed: 30-Sep-2014].

[23] "Amysta." [Online]. Available: www.amysta.com. [Accessed: 30-Sep-2014].

[24] "Amazon Elastic Compute Cloud (EC2)." [Online]. Available: http://aws.amazon.com/ec2/. [Accessed: 30-Sep-2014].

[25] "Open Cloud Computing Interface." [Online]. Available: http://occi-wg.org/. [Accessed: 30-Sep-2014].

[26] "OVF Open Virtualization Format." [Online]. Available: http://www.dmtf.org/standards/ovf. [Accessed: 30-Sep-2014].

[27] "Horizon, OpenStack Dashboard." [Online]. Available: http://www.openstack.org/software/openstack-dashboard/. [Accessed: 30-Sep-2014].
[28] "Nagios." [Online]. Available: http://www.nagios.org. [Accessed: 30-Sep-2014].
[29] *collectd - The system statistics collection daemon.* .
[30] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Comput.*, vol. 30, no. 7, pp. 817–840, 2004.
[31] "Zenoss." [Online]. Available: http://www.zenoss.org. [Accessed: 30-Sep-2014].
[32] A. Fischer, J. F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *Commun. Surv. Tutor. IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.
[33] J. Rayport and A. Heyward, *Envisioning the cloud: The next computing paradigm and its implication for technology policy*. 2011.
[34] J. Weinman, "Mathematical proof of the inevitability of cloud computing," *Cloudonomics Com Available Online Httpcloudonomics Wordpress Comlast Retrieved Novemb. 8 2010*, 2011.
[35] M. Klems, J. Nimis, and S. Tai, "Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing," *Lect. Notes Bus. Inf. Process.*, vol. 22, pp. 110–123, 2009.
[36] H.-L. Truong and S. Dustdar, "Composable cost estimation and monitoring for computational applications in cloud computing environments," *Procedia Comput. Sci.*, vol. 1, no. 1, pp. 2175–2184, 2010.
[37] T. Alford and G. Morton, "The Economics of Cloud Computing: Addressing the Benefits of Infrastructure in the Cloud," *Booz Allen Hamilt.*, 2009.
[38] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise it system to iaas," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 450–457.
[39] J. Koomey, K. Brill, P. Turner, J. Stanley, and B. Taylor, "A simple model for determining true total cost of ownership for data centers," *Uptime Inst. White Pap. Version*, vol. 2, 2007.
[40] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1–12.
[41] A. Khajeh-Hosseini, D. Greenwood, J. W. Smith, and I. Sommerville, "The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise," *Softw. Pract. Exp.*, vol. 42, no. 4, pp. 447–465, 2012.
[42] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam, "To move or not to move: The economics of cloud computing," in *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, 2011, pp. 5–5.
[43] F. Koch, M. D. Assunçao, and M. A. Netto, "A cost analysis of cloud computing for education," in *Economics of Grids, Clouds, Systems, and Services*, Springer, 2012, pp. 182–196.
[44] R. Hwang, C. Lee, Y. Chen, and D. Zhang-Jian, "Cost optimization of elasticity cloud resource subscription policy," 2013.
[45] Y. Kessaci, N. Melab, and E.-G. Talbi, "A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation," *Clust. Comput.*, vol. 16, no. 3, pp. 451–468, 2013.
[46] M. Bjorkqvist, L. Y. Chen, and W. Binder, "Cost-driven service provisioning in hybrid clouds," in *Proceedings of the 2012 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2012, pp. 1–8.

[47] M. M. Kashef and J. Altmann, “A cost model for hybrid clouds,” in *Economics of Grids, Clouds, Systems, and Services*, Springer, 2012, pp. 46–60.
[48] K.-T. Tran and N. Agoulmine, “Adaptive and Cost-effective Service Placement,” in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–6.
[49] L. F. Bittencourt and E. R. M. Madeira, “HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds,” *J. Internet Serv. Appl.*, vol. 2, no. 3, pp. 207–227, 2011.
[50] *Google App Engine*. .
[51] O. Bini, *JRuby & Ioke on Google App Engine for Java, video. Google I/O, 2009*. .
[52] A. Bedra, “Getting Started with Google App Engine and Clojure,” *Internet Comput. IEEE*, vol. 14, no. 4, pp. 85–88, 2010.
[53] J. D. Blower, “GIS in the cloud: implementing a web map service on Google App Engine,” in *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, New York, NY, USA, 2010, pp. 34:1–34:4.
[54] “AppScale.” [Online]. Available: http://www.appscale.com/. [Accessed: 30-Sep-2014].
[55] *Google partners*. .
[56] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski, “AppScale: Scalable and Open AppEngine Application Development and Deployment,” in *Cloud Computing*, vol. 34, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, pp. 57–70.
[57] “CapeDwarf.” [Online]. Available: http://www.jboss.org/capedwarf. [Accessed: 30-Sep-2014].
[58] S. Yang, “Running Google App Engine Apps in JBoss AS7: Aleš Justin Discusses CapeDwarf.” [Online]. Available: http://www.infoq.com/articles/capedwarf.
[59] M. Lazar, *Run Google App Engine Apps on the OpenShift PaaS*. .
[60] “CapeDwarf Documentation.” [Online]. Available: http://www.jboss.org/capedwarf/docs. [Accessed: 30-Sep-2014].
[61] J. Lautamäki, A. Nieminen, J. Koskinen, T. Aho, T. Mikkonen, and M. Englund, “CoRED: browser-based Collaborative Real-time Editor for Java web applications,” in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 1307–1316.
[62] M. Grönroos, *Book of Vaadin - 7th Edition - 2nd revision*. Uniprint, 2013.
[63] Oracle, “Java SE Documentation / Reflection.” [Online]. Available: http://docs.oracle.com/javase/6/docs/technotes/guides/reflection.
[64] T. Aho, A. Ashraf, M. Englund, J. Katajamäki, J. Koskinen, J. Lautamäki, A. Nieminen, I. Porres, and I. Turunen, “Designing IDE as a service,” *Commun. Cloud Softw.*, vol. 1, no. 1, 2011.
[65] “NIST Cloud Computing Security Reference Architecture,” National Institute of Standards and Technology, NIST SP 500-299, May 2013.
[66] “NIST Cloud Computing Reference Architecture,” National Institute of Standards and Technology, NIST SP 500-292, Sep. 2011.
[67] “The Role of CSB in the Cloud Services Value Chain,” Gartner, G00218960, Oct. 2011.
[68] Forrester, “Cloud Brokers will reshape the cloud – getting Ready for the future Cloud Business.” Sep-2012.
[69] F. Fowley, C. Pahl, and L. Zhang, “A comparison framework and review of service brokerage solutions for cloud architectures,” in *Service-Oriented Computing–ICSOC 2013 Workshops*, 2014, pp. 137–149.

[70] "State of the art and research baseline," *Broker@Cloud Project*. [Online]. Available: http://www.broker-cloud.eu/documents/deliverables/d2-1-state-of-the-art-and-research-baseline.
[71] A. B. Mohammed, J. Altmann, and J. Hwang, "Cloud computing value chains: Understanding businesses and value creation in the cloud," in *Economic models and algorithms for distributed systems*, Springer, 2010, pp. 187–208.
[72] M. Maurer, V. C. Emeakaroha, M. Risch, I. Brandic, and J. Altmann, "Cost and benefit of the SLA mapping approach for defining standardized goods in cloud computing markets," in *International Conference on Utility and Cloud Computing (UCC 2010) in conjunction with the International Conference on Advanced Computing (ICoAC 2010)*, 2010.
[73] J. Metzler and S. Taylor, "Cloud Computing: Reality vs. Fiction," 2008. [Online]. Available: http://www.networkworld.com/article/2216572/lan-wan/cloud-computing--reality-vs--fiction.html. [Accessed: 30-Sep-2014].
[74] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 973–985, 2013.
[75] J. Altmann, C. Courcoubetis, and M. Risch, "A marketplace and its market mechanism for trading commoditized computing resources," *Ann. Telecommun.-Ann. Télécommunications*, vol. 65, no. 11–12, pp. 653–667, 2010.
[76] E. Elmroth and L. Larsson, "Interfaces for placement, migration, and monitoring of virtual machines in federated clouds," in *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, 2009, pp. 253–260.
[77] A. Parameswaran and A. Chaddha, "Cloud interoperability and standardization," *SETlabs Brief.*, vol. 7, no. 7, pp. 19–26, 2009.
[78] "Libvirt." [Online]. Available: http://libvirt.org. [Accessed: 29-Sep-2014].
[79] M. Rak, S. Venticinque, T. Máhr, G. Echevarria, and G. Esnal, "Cloud application monitoring: The mOSAIC approach," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 758–763.
[80] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "Resource monitoring and management with OVIS to enable HPC in cloud computing environments," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1–8.
[81] A. Ciuffoletti, "(UNACCESSABLE) Describing a monitoring infrastructure with an OCCI-compliant schema." 2013.
[82] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. Fitó, and D. Henriksson, "A Multi-level Architecture for Collecting and Managing Monitoring Information in Cloud Environments.," in *CLOSER*, 2011, pp. 232–239.
[83] S. Clayman, A. Galis, and L. Mamatas, "Monitoring virtual networks with lattice," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, 2010, pp. 239–246.
[84] "Nagios documentation," 2010. [Online]. Available: http: //www.nagios.org/documentation.
[85] "OAuth." [Online]. Available: http://oauth.net/. [Accessed: 24-Jul-2014].
[86] OASIS, "Security Assertion Markup Language (SAML) v.2.0." [Online]. Available: https://www.oasis-open.org/standards#samlv2.0. [Accessed: 24-Jul-2014].
[87] OpenID Foundation, "OpenID." [Online]. Available: http://openid.net/. [Accessed: 24-Jul-2014].
[88] "WS-Security Specification." [Online]. Available: http://msdn.microsoft.com/en-us/library/ms951273.aspx. [Accessed: 24-Jul-2014].

[89] "Platform for Privacy Preferences (P3P) Project." [Online]. Available: http://www.w3.org/P3P/. [Accessed: 24-Jul-2014].
[90] "OASIS eXtensible Access Control Markup Language (XACML) TC." [Online]. Available: https://www.oasis-open.org/committees/xacml. [Accessed: 24-Jul-2014].
[91] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, and others, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Memb. Submiss.*, vol. 21, p. 79, 2004.
[92] IBM Research - Zurich, "Idemix Identity Mixer," REPLACE. [Online]. Available: http://www.zurich.ibm.com/security/idemix/. [Accessed: 24-Jul-2014].
[93] Microsoft Research, "U-Prove." [Online]. Available: http://research.microsoft.com/en-us/projects/u-prove/. [Accessed: 24-Jul-2014].
[94] ctk, "PRIME - Privacy and Identity Management for Europe — Portal for the PRIME Project." [Online]. Available: https://www.prime-project.eu/. [Accessed: 24-Jul-2014].
[95] "PrimeLife - Privacy and Identity Management in Europe for Life." [Online]. Available: http://primelife.ercim.eu/. [Accessed: 24-Jul-2014].
[96] "ABC4Trust EU Project." [Online]. Available: https://abc4trust.eu/. [Accessed: 24-Jul-2014].
[97] G. Menaga and S.Subasree, "Development of Optimized Resource Provisioning On-Demand Security Architecture for Secured Storage Services in Cloud Computing," *Int. J. Eng. Sci. Innov. Technol. IJESIT*, vol. 2, no. 3, May 2013.
[98] S. P. Praveen, U. Tulasi, B. Vishnu, and A. Yuvakrishna, "A New Approach for Optimizing Resource Provisioning In Cloud Computing Using OCRP Algorithm," *Softw. Eng. Technol.*, vol. 5, no. 12, p. 381, 2013.
[99] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *Serv. Comput. IEEE Trans. On*, vol. 5, no. 2, pp. 164–177, 2012.
[100] P. K. Sur, "Resource Reservation Strategy for Cost Minimization in Cloud Computing Environment," Jadavpur University Kolkata, 2013.
[101] V. Vignesh, K. Sendhil Kumar, and N. Jaisankar, "Resource Management and Scheduling in Cloud Environment," *Int. J. Sci. Res. Publ.*, vol. 3, no. 6, 2013.
[102] D. Funke, F. Brosig, and M. Faber, "Towards truthful resource reservation in cloud computing," in *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on*, 2012, pp. 253–262.
[103] Q. Li and Y. Guo, "Optimization of Resource Scheduling in Cloud Computing.," in *SYNASC*, 2010, pp. 315–320.
[104] S. D. Barnabas, "Resource Reservation in the Cloud based on Infrastructure as a Service (IaaS)," *Res. J. Comput. Syst. Eng. – RJCSE*, no. 4, Jun. 2013.
[105] C. Schulte, *Principles and Practice of Constraint Programming: 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*. Springer Berlin Heidelberg, 2013.
[106] "The Standardisation Environment for Cloud Computing." Federal Ministry for Economic Affairs and Energy, Feb-2012.
[107] "OneAPI | GSMA OneAPI." [Online]. Available: http://www.gsma.com/oneapi/. [Accessed: 27-Jun-2014].
[108] "Billing for cloud services: The impact on revenue management systems for operators." Analysys Mason.
[109] "Gartner Says Nearly Half of Large Enterprises Will Have Hybrid Cloud Deployments by the End of 2017." [Online]. Available: http://www.gartner.com/newsroom/id/2599315. [Accessed: 29-Sep-2014].
[110] "Gartner's Forecast Analysis: Enterprise Application Software, Worldwide, 2010-2016. (Author's calculations)." [Online]. Available: http://blogs-

images.forbes.com/louiscolumbus/files/2013/02/public-cloud-forecast.jpg. [Accessed: 30-Sep-2014].
[111] P. Mell and T. Grance, "The NIST definition of cloud computing," *Natl. Inst. Stand. Technol.*, vol. 53, no. 6, p. 50, 2009.
[112] "Business Process as a Service (BPaaS) - Gartner IT Glossary," *Business Process as a Service (BPaaS) - Gartner IT Glossary*. [Online]. Available: http://www.gartner.com/it-glossary/business-process-as-a-service-bpaas. [Accessed: 29-Sep-2014].
[113] "IT Spending Report | Gartner Worldwide IT Spending Forecast | Gartner Inc." [Online]. Available: http://www.gartner.com/technology/research/it-spending-forecast/.
[114] "Forrester Research□: Research□: The Public Cloud Market Is Now In Hypergrowth." [Online]. Available: https://www.forrester.com/The+Public+Cloud+Market+Is+Now+In+Hypergrowth/fulltext/-/E-RES113365. [Accessed: 30-Sep-2014].
[115] "Roundup Of Small & Medium Business Cloud Computing Forecasts And Market Estimates, 2013," *A Passion for Research*. .
[116] "Public Cloud Services Spending Is Being Driven by Enterprise Applications Solutions, According to IDC," *www.idc.com*. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS24977214. [Accessed: 29-Sep-2014].
[117] "Forrester lowers its 2014 global IT spending forecast," *Computerworld*, 13-Aug-2014. [Online]. Available: http://www.computerworld.com/article/2491170/it-management/forrester-lowers-its-2014-global-it-spending-forecast.html. [Accessed: 29-Sep-2014].
[118] "Microsoft and IBM Chase Amazon while Google Falls Off the Pace | Synergy Research Group." [Online]. Available: https://www.srgresearch.com/articles/microsoft-and-ibm-chase-amazon-while-google-falls-pace.
[119] "SalesForce.com Annual Report 2014." .
[120] M. Porter E., *Competitive Advantage*. New York: The Free Press, 1985.
[121] "Global Consulting Market · Consultancy.uk." [Online]. Available: http://www.consultancy.uk/consulting-industry/global-consulting-market. [Accessed: 29-Sep-2014].
[122] "Market Share Analysis: Consulting Services, Worldwide, 2013." [Online]. Available: https://www.gartner.com/doc/2733920/market-share-analysis-consulting-services. [Accessed: 29-Sep-2014].
[123] B. Levine, "Forrester Report: Brighter Days Coming for Tech Spending," *CMSWire.com*. [Online]. Available: http://www.cmswire.com/cms/information-management/forrester-report-brighter-days-coming-for-tech-spending-018989.php. [Accessed: 29-Sep-2014].
[124] "IDC Market Scape: Worldwide Cloud Professional Services 2013 Vendor Analysis." .
[125] L. Badger, R. Bohn, S. Chu, M. Hogan, F. Liu, V. Kaufmann, J. Mao, J. Messina, K. Mills, A. Sokol, J. Tong, F. Whiteside, and D. Leaf, "US Government Cloud Computing Technology Roadmap Volume II Release 1.0 - Special Publication 500-293 (draft)." NIST, 2011.
[126] "The OnApp Federation | OnApp." [Online]. Available: http://onapp.com/federation/.
[127] "IDC | A Passion for Research." .
[128] "Services and SaaS trends," *PwC*. [Online]. Available: http://www.pwc.com/gx/en/technology/publications/global-software-100-leaders/saas-trends.jhtml. [Accessed: 29-Sep-2014].
[129] "VMWare, INC Annual Report 2013." .

[130] "Citrix Reports Fourth Quarter and Fiscal Year Financial Results." [Online]. Available: http://www.citrix.com/content/dam/citrix/en_us/documents/news/citrix-reports-fourth-quarter-and-fiscal-year-financial-results-2013.pdf. [Accessed: 30-Sep-2014].
[131] "Research and Markets: Global Colocation Market Report 2013-2018: Market will Grow from $25.72 Billion to $43.34 Billion," *Reuters*, 12-Dec-2013.
[132] "Gartner says worldwide IT spending on pace to grow 2.1 percent in 2014 | Channel Post MEA." .
[133] D.-W.-I. S. Lehmann and P. Buxmann, "Pricing strategies of software vendors," *Bus. Inf. Syst. Eng.*, vol. 1, no. 6, pp. 452–462, 2009.
[134] J. O. Kephart, J. E. Hanson, and A. R. Greenwald, "Dynamic pricing by software agents," *Comput. Netw.*, vol. 32, no. 6, pp. 731–752, 2000.
[135] Y. Masuda and S. Whang, "Dynamic pricing for network service: Equilibrium and stability," *Manag. Sci.*, vol. 45, no. 6, pp. 857–869, 1999.
[136] P. R. Dasgupta and Y. Hashimoto, "Multi-attribute dynamic pricing for online markets using intelligent agents," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2004, pp. 277–284.
[137] P. Dasgupta and R. Das, "Dynamic pricing with limited competitor information in a multi-agent economy," in *Cooperative Information Systems*, 2000, pp. 299–310.
[138] J. Altmann and L. Rhodes, "Dynamic Netvalue Analyzer-A Pricing Plan Modeling Tool for ISPs Using Actual Network Usage Data," in *Advanced Issues of E-Commerce and Web-Based Information Systems, 2002.(WECWIS 2002). Proceedings. Fourth IEEE International Workshop on*, 2002, pp. 143–148.
[139] J. Altmann, B. Rupp, and P. Varaiya, "Internet demand under different pricing schemes," in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 9–14.
[140] R. Harmon, H. Demirkan, B. Hefley, and N. Auseklis, "Pricing strategies for information technology services: A value-based approach," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, 2009, pp. 1–10.
[141] K. B. Monroe, *Pricing: Making profitable decisions, 3rd Edition*. McGraw-Hill New York, 2003.
[142] S. Flake, J. Tacken, and C. Zoth, "Real-time Billing as a Service – A standard-based proof-of-concept implementation," presented at the 8th International Workshop on Service-Oriented Cyber-Physical Systems in Converging Networked Environments (SOCNE 2014), Barcelona, Spain, 2014.
[143] FI-WARE Consortium, "D.2.6.2: State of the Art Analysis – Emerging Technologies (Document ID: ICT-2011-FI-285248-WP2-D.2.6.2)." Jun-2013.
[144] FI-WARE Consortium, "Store Open RESTful API Specification." .
[145] Mobile Cloud Computing Consortium, "D5.1 Design of Mobile Platform Architecture and Services." Nov-2013.
[146] ETSI, "Charging architecture and principles.," *3GPP TS 32240*.
[147] ETSI, "Diameter charging applications," *3GPP TS 32299*.
[148] C. Fiehe, A. Litvina, J. Tonn, J. Wu, M. Scheel, A. Brinkmann, L. Nagel, K. Narayanan, C. Zoth, H.-J. Golz, S. Unger, W. Thronicke, and F. Pursche, "Building a Medical Research Cloud in the EASI-CLOUDS Project," presented at the 6th International Workshop on Science Gateways (IWSG 2014), Dublin, Ireland, 2014.
[149] E. Elmroth, F. G. Marquez, D. Henriksson, and D. P. Ferrera, "Accounting and billing for federated cloud infrastructures," in *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, 2009, pp. 268–275.

[150] "Linked USDL, comprising USDL-Core, USDL-Pricing and USDL-SLA." [Online]. Available: http://www.linked-usdl.org/.

[151] C. Pedrinaci, J. Cardoso, and T. Leidig, "Linked USDL: A Vocabulary for Web-Scale Service Trading," in *The Semantic Web: Trends and Challenges*, Springer, 2014, pp. 68–82.

[152] "WSAG4J - WS-Agreement for Java," *WSAG4J - WS-Agreement for Java*. [Online]. Available: http://wsag4j.sourceforge.net/site/index.html.

[153] L. Wu, S. K. Garg, R. Buyya, C. Chen, and S. Versteeg, "Automated SLA negotiation framework for cloud computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 235–244.

[154] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks," 2013, pp. 1–7.