

2 Background

This chapter introduces the fundamentals and major terms of this thesis. Since cloud-based services are the focus of the present work, Section 2.1 discusses different definitions of cloud computing and describes the main characteristics of cloud-based services. Furthermore, Section 2.1 elaborates on the technologies enabling cloud-based services and describes related paradigms. In order to provide an overview, explaining how guarantees for certain quality levels of cloud-based services can be obtained and how it can be ensured that they are maintained during service usage, Section 2.2 outlines the activities of Service Level Management. Furthermore, it introduces the term Service Level Agreements and elaborates on how they relate to Quality of Service (QoS) parameters well known from the area of communication networks. Section 2.3 presents mechanisms and concepts on how to automate the negotiation of quality guarantees. Finally, Section 2.4 describes objectives and principles when it comes to measure the current performance of a cloud-based service against these guarantees.

2.1 Cloud Computing

2.1.1 Definition

Although cloud computing has become a major computing paradigm of our time, no widely accepted definition exists, as yet. Basically, the term *cloud computing* was coined by today's famous cloud providers Amazon and Google in late 2006 (cf., e.g., [147, 218, 221]). Although it is controversial, which of them was the first propagating this new computing paradigm, the term *cloud computing* itself has its origins in industry. Research in academia started later in 2007 [218]. Nevertheless, the underlying idea of provisioning computing resources as a utility is not new and has been a vision since the era of mainframes (cf., e.g., [25, 147, 221]). Early in 1961, John McCarthy envisaged that “‘*computation*’ *might someday be organized as a public utility*” [147]. In 1969, the same vision was again uttered by Leonard Kleinrock, a chief scientist of the Advanced Research Project Agency Network (ARPANET) project, the predecessor of today's Internet [111]:

“As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of

‘computer utilities’, which, like present electric and telephone utilities, will service individual homes and offices across the country.”

Kleinrock already envisioned some of the main characteristics of cloud-based services today, namely, that computing resources will be provided on-demand with ubiquitous access in the form of web-based services (cf., e.g., [25, 111]). These characteristics already emphasize some of the main advantages of cloud computing from a consumer’s point of view. Since cloud-based services are provided in a manner similar to other common utilities, such as electricity, consumers do not have to maintain their own complex IT infrastructure anymore and only pay for the services based on their usage [25]. In addition to the advantages on the consumer side, there are also several advantages on the provider side. Some of them have already been mentioned in one of the first definitions of cloud computing by Ian Foster (co-inventor of the grid computing paradigm as a preceding paradigm of cloud computing) and others in 2008 [72]:

“A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.”

According to Foster et al., the most important aspects in this definition are that clouds are “*massively scalable*”, provide “*different levels of services*”, are “*driven by economies of scale*”, and that “*the services can be dynamically configured [...] and delivered on demand*” [72]. All these aspects provide major advantages to service providers and consumers. By offering services at different levels and enabling their dynamic configuration, a high level of customization of cloud-based services can be achieved. Furthermore, massive scalability allows the consideration of varying business requirements and the handling of unforeseen peak behavior. Last but not least, by aggregating cloud computing resources into large-scale resource pools, economies of scale can be realized on the provider side. That is, cost savings can be achieved by conducting the same operations more efficiently on a larger scale [85]. Cloud service providers are then able to pass on these cost savings to their customers. All these advantages make cloud computing highly attractive to the industry. Nevertheless, the characteristics of cloud computing, especially the high dynamics, scalability, and flexibility, also raise several new challenges (cf., e.g., [9, 17, 130, 197, 221]) that have been addressed in academic research since 2008. A major number of these scientific works is based on the definition by the National Institute of Standards and Technology (NIST). The thesis at hand also follows the NIST definition, which reads as follows [131]:

“Cloud computing is a model for enabling ubiquitous, 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models.”

The NIST definition incorporates all the aspects of cloud computing mentioned above. Moreover, it also provides a further characterization of cloud computing by defining its essential characteristics, service models and deployment models, which are described in detail in the following.

On-demand self-service According to their demand at particular timeslots, consumers are able to obtain computing capabilities, such as software, storage, or CPU time, automatically from a service provider without human interaction [45, 131]. The automated resource provisioning also enables service providers to adapt rapidly to dynamic changes, such as flash crowd effects [221].

Broad network access Unified access to computing capabilities over the Internet permits the usage of heterogeneous clients, such as mobile phones or laptops [131, 197]. In addition, many of today’s cloud providers place their data centers at different locations worldwide in order to improve network performance [221].

Resource pooling The computing resources are aggregated to large-scale resource pools and dynamically shared among multiple consumers based on their demand. In doing so, virtual computing resources assigned to different consumers can be co-located on the same physical machine, referred to as multi-tenancy [131, 197]. In general, consumers are not aware of the exact location of their leased resources, but may specify location preferences at a higher level, e.g., country or data center [131].

Rapid elasticity From a consumer’s perspective, there seem to be unlimited resource capacities within a cloud and additional/less resources can be rapidly acquired in line with a consumer’s demand by, e.g., scaling up or scaling down [45, 131]. More dimensions of scalability are typically distinguished: scale-up/-down and scale-out. While the former requires an application following a concurrent programming model and only allows the resources to be scaled-up to the resource capacities of a single machine, scale-out requires an application following a distributed programming model and allows the combination of the resources of multiple machines resulting in a single virtual machine [76].

Measured service Resource usage is tracked by metering capabilities allowing for an automated control and optimization of resources and providing transparency for cloud providers and consumers [131]. Typically, a pay-per-use model is applied, which lowers operating costs for both, providers and consumers, but increases the complexity in controlling these operating costs from a provider's perspective [221].

The essential characteristics in the NIST definition represent generic properties of cloud-based systems. However, within a cloud-based system, the computing resources provided to consumers differ. Basically, different levels of services are offered, such as “*computing power, storage, platforms, and services*” [72], and several proposals for a universal classification of these service models have been made so far (cf., e.g., [45, 57, 131, 160, 220, 226]). Among all these proposals, the NIST definition specifies the following three service models, which are the most common. Additional service models are usually a special case of one of these.

Software as a Service On the Software as a Service (SaaS) level, applications running on a cloud provider's infrastructure are provided to consumers. Access to these applications can be gained through either thin client interfaces (e.g., a web browser) or program interfaces. Besides being able to configure certain application settings, consumers are neither able to control the underlying cloud infrastructure, such as the network, server, or operating system, nor individual capabilities of the application [131]. Popular examples are the Customer Relationship Management software of Salesforce¹ and Google Docs² [197].

Platform as a Service According to the NIST definition [131], the Platform as a Service (PaaS) level comprises capabilities to develop and deploy applications that are build “*using programming languages, libraries, services, and tools supported by the provider*”. Similar to SaaS, consumers do not have control over the underlying cloud infrastructure, but, in contrast, are able to control the deployed applications and may possibly configure settings of the hosting environment. Popular examples are Google App Engine³ and Microsoft Azure⁴ [197].

Infrastructure as a Service The Infrastructure as a Service (IaaS) level incorporates the provisioning of fundamental computing resources, such as processing power, networks, and data storage, that can be used by consumers in order to deploy and run applications and operating systems. Again, consumers do not control the

¹ <http://www.salesforce.com/crm>

² <http://www.docs.google.com>

³ <http://cloud.google.com/appengine/>

⁴ <http://azure.microsoft.com>

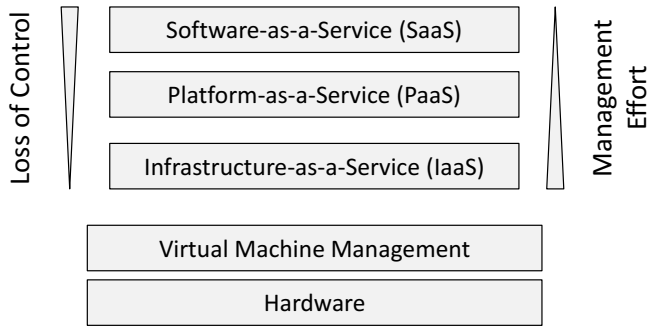


Figure 2.1: Cloud service stack and service models

underlying cloud infrastructure⁵, similar to SaaS and PaaS, but are able to manage applications, operating systems, data storage, as well as selected network components, such as a host firewall [131]. Although the IaaS model provides more flexibility to consumers than PaaS, it also implies an increased management effort on the consumer side regarding, e.g., updates of the operating system. Popular examples are Amazon’s Elastic Compute Cloud (EC2)⁶ and Amazon’s Simple Storage Service (S3)⁷ [197].

The three service models form a layered model in a cloud architecture as depicted in Figure 2.1 and are located on top of a virtual machine management layer, which is responsible for the management of the virtualized computing resources above (cf. Section 2.1.2), and the physical hardware. In general, when moving from the bottom (i.e., IaaS) to the top (i.e., SaaS) of this layered model, the loss of control for consumers increases while the management effort decreases. In addition to the different service models, clouds can be further distinguished based on their ownership, target user group(s), and location (cf. [197]) resulting in the following deployment models as proposed by the NIST definition [131].

Public cloud A cloud infrastructure that is provided to “*the general public*” and that is located on the premises of a certain cloud provider. That provider owns, manages, and operates the cloud and may be an organization from the industrial, academic, or governmental sectors, or a combination of these [131]. The economic benefit of utilizing resources from public clouds is often described as turning capital expenditure into operational expenditure [9]. However, consumers also loose fine-grained control over issues such as data, network and security concerns [221].

⁵ It should be noted that the term ‘cloud infrastructure’ in this thesis denotes the physical infrastructure of a cloud and, thus, is different from the virtual IaaS layer.

⁶ <http://aws.amazon.com/ec2/>

⁷ <http://aws.amazon.com/s3/>

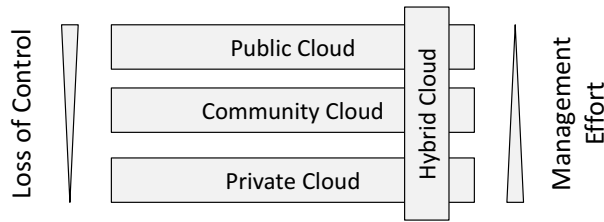


Figure 2.2: Cloud deployment models

Community cloud A cloud infrastructure that is provided exclusively to a specific community comprising consumers of different organizations sharing similar concerns. One or more organizations being part of that community, a third party, or a combination of both may be the owner, manager, and operator of that cloud and it may be located on or off the premises of the collaborating organizations [131].

Private cloud A cloud infrastructure that is provided exclusively to a single organization that may comprise multiple consumers, such as different business units. The organization itself, a third party, or a combination of both may be the owner, manager, and operator of that cloud and it may be located on or off the premises of that organization [131]. Running a private cloud offers the highest level of control over aspects such as performance, security concerns, reliability, and mission-critical tasks, but does not allow, for example, to eliminate up-front investments in contrast to public clouds [45, 221].

Hybrid cloud A cloud infrastructure that is a combination of two or more of the other deployment models. Each of the clouds being part of the hybrid cloud remains a discrete entity, although being able to move data and applications to the other clouds by applying appropriate (i.e., standardized or proprietary) technology [131]. The hybrid cloud deployment model can be beneficial in specific scenarios (cf. [45, 131, 197]). In case of a bursting scenario, for instance, a private cloud can acquire more resources from a public cloud. Also, in order to optimize resource utilization, a hybrid cloud permits keeping core business activities in a private cloud, while moving other, less critical activities into a public cloud.

Besides these four, most common deployment models, as depicted in Figure 2.2, also a fifth type, namely, a Virtual Private Cloud (VPC) is mentioned in literature. Zhang et al. [221] describe a VPC as “*a platform running on top of public clouds*” that permits own network topologies and security settings (e.g., firewall rules) to be established by exploiting virtual network technology. That is, the underlying communication network is also virtualized in addition to servers and applications. In doing so, a traditional

private network infrastructure can be easily rebuild within a public cloud infrastructure while keeping the privacy of the resulting communication network.

According to Dillon et al. [45], a VPC is a mixture between a private and a public cloud. On the one hand, it utilizes resources from a public cloud provider. On the other hand, it provides a secure connection to these resources from an organization's corporate infrastructure and isolates these resources from other public cloud resources. The authors mention Amazon Virtual Private Cloud⁸ as the first example for this kind of deployment model.

2.1.2 Enabling Technologies

In general, cloud computing is considered to be a paradigm (cf., e.g., [25, 72, 130, 147, 207, 218, 220, 221]) and, thus, it is not bound to a specific implementation. Nevertheless, it is based on certain technologies (cf., e.g., [25, 72, 209, 221]). Existing works often make no explicit distinction between enabling technologies and other computing paradigms that are related to cloud computing. However, since enabling technologies refer to the concrete realization of a cloud-based system and computing paradigms refer to abstract principles and certain concepts of an IT-based architecture, an explicit separation between both concerns is made in the following. While enabling technologies are described in this section, related paradigms are discussed in the subsequent Section 2.1.3. Concerning enabling technologies, the work by Erl et al. [57] contains the most comprehensive enumeration and description of these technologies.

According to Erl et al. [57], the following cloud-enabling technologies can be distinguished. The authors state that although each of these technologies existed before cloud computing gained momentum, some of them further evolved with the advent of cloud computing.

Broadband Networks and Internet Architecture The existence of global networks, such as the Internet, enables remote provisioning of computing capabilities by cloud providers and permits ubiquitous access to these capabilities for cloud consumers [57, 131]. Hence, the quality of cloud-based systems depends highly on the performance of these networks. Therefore, enterprise consumers are faced with the decision whether to choose a private cloud deployment in their own data center and access its capabilities through the organization's own network or to choose a public, Internet-enabled cloud deployment model. In the former case, the network is usually completely managed and controlled by the respective organization. In the latter case, the cloud provider assumes these responsibilities and can provide ubiquitous access to the computing capabilities through an Internet connection (cf. Section 2.1.1). While a private cloud deployment is less affected by connectivity, bandwidth, or latency issues when accessing the computing capabilities through a corporate network, public

⁸ <http://aws.amazon.com/vpc>

cloud deployments facing these issues usually provide superior Internet connectivity at additional network usage charges [57].

Data Center Technology Aggregating IT resources in large data centers allows power-sharing, IT resources, and IT personnel and, thus, allows for higher efficiency [57]. Typically, IT resources in a data center comprise servers, storage, networking components, and software (cf., e.g., [57, 131]), and can be physical or virtual in nature. The physical resources build the physical infrastructure of a data center, which serves as a hosting environment for virtual resources. The virtual resources represent physical resources that are virtualized in order to, e.g., facilitate corresponding management tasks, such as their allocation, operation, release, monitoring and control [57].

Virtualization Technology Virtualization is an inherent technology of cloud-based systems and, thus, already mentioned in several definitions of cloud computing (cf., e.g., [72, 131]). The concept of virtualization allows the abstraction from the details of physical resources and the consideration of these as resource pools. In doing so, a logical view on the physical resources can be established and they can be managed as a whole. Hence, (multiple) virtual resources can be dynamically assigned to them on-demand according to particular business requirements [12, 221]. In general, different kinds of resources can be virtualized, such as servers, storage, networks, power, and software [12, 27, 57]. A virtualized server is denoted as Virtual Machine (VM) and multiple of these logical VMs can be hosted in parallel on a single physical machine [25, 221]. Major advantages of this kind of abstraction are as follows [25, 72]. First of all, virtualization permits encapsulation, so that, e.g., different operating systems can be run on different VMs residing on the same physical machine. A second major advantage is improved resource utilization, since each VM can utilize a different amount of the underlying physical resources. Finally, virtualization enables dynamic resource provisioning, which comprises the automation of activities such as starting, stopping, deploying, and migrating VMs. In general, different techniques for server virtualization exist, from which hardware-based virtualization is the most common [27]. The logical layers in case of hardware-based virtualization are depicted in Figure 2.3. An implementation of this type of virtualization is based on a so-called hypervisor. Basically, a hypervisor represents a meta operating system with minimal system requirements. It establishes a VM management layer between a host and the guest operating systems running different applications in order to distribute the physical resources among the different VMs and to coordinate their hardware access [12, 27, 57].

Web Technology Since cloud computing is based on the existence of broadband networks, Web technology is used for implementing and managing cloud services and their functionality is exposed to consumers through Web-based interfaces [26]. Web

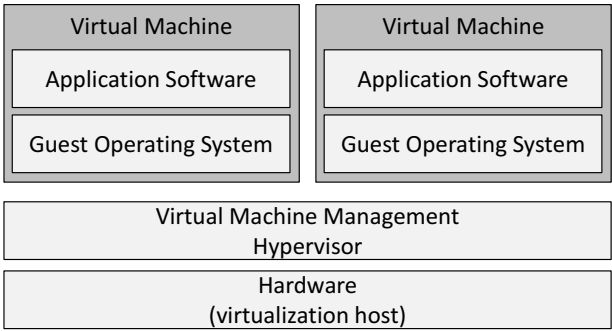


Figure 2.3: Logical layers of hardware-based virtualization (based on [12, 57])

standards encompass the Uniform Resource Locator Uniform Resource Locator (URL) to identify and locate Web-based resources, the Hypertext Transfer Protocol (HTTP) to exchange requests with a resource host, and the Hypertext Markup Language (HTML) and Extensible Markup Language (XML) to express Web-based data and metadata, respectively [57]. Besides these well-known Web technologies, a set of new, so-called Web 2.0 technologies has gained momentum. Web 2.0 enables building interactive and flexible applications based on new technologies, such as Asynchronous JavaScript and XML (AJAX), Really Simple Syndication (RSS), and well-known service technology such as Web services [26, 72]. In doing so, Web 2.0 allows the provision of Desktop-like applications over the Web. A popular example of a Web 2.0-based cloud application is Google Docs⁹[26].

Multi-tenant Technology A multi-tenant application enables a user (denoted as tenant) to use the same application concurrently with multiple other users without interfering with the other users due to isolated operational environments. A dedicated instance is provided to each user with, e.g., a customized user interface, individual application logic, and own data. The users are sharing various artifacts, such as middleware and databases, while adhering to certain security levels so that tenants are not able to access private data of other users. The isolation of the operational environments allows the implementation of metering capabilities that track each tenant’s usage separately and supports the scalability when the number of tenants increases. SaaS typically implements this kind of application [57].

Service Technology Clouds have adopted service technologies in order to provide a predefined Application Programming Interface (API) for using, configuring, and programming cloud services [57, 72]. These service technologies comprise Web services

⁹ <https://www.google.com/intl/en/docs/about/>

and Representational State Transfer (REST). The basic Web service technologies are as follows (cf., e.g., [4, 55, 57]). First of all, they incorporate the Web Service Description Language (WSDL) to describe the interface, operations, and message types of a Web service. Furthermore, the XML Schema Definition Language (XSD) is used to specify the data structures of the message types within a WSDL document. Simple Object Access Protocol (SOAP) (formerly denoted as the Simple Object Access Protocol) constitutes another standard and represents a messaging format for requests and response messages. Finally, Universal Description, Discovery and Integration (UDDI) is a specification for service registries that can be used to publish WSDL documents and to query for particular Web services. Further, enhanced Web service technologies denoted as WS-* extensions or WS-* stack exist concerning additional areas, such as message-level security and reliable messaging (cf., e.g., [55, 57, 154]). Besides the traditional Web service technologies, REST evolved as an alternative approach for implementing services. REST was originally introduced as a new “*architectural style for distributed hypermedia systems*” by Roy Thomas Fielding in his doctoral dissertation in the year 2000 [65]. RESTful Web services are perceived to exhibit less complexity due to the fact that they leverage core Web technologies, such as HTTP and the Uniform Resource Identifier (URI) for global addressing [57, 154]. Erl et al. [57] further mention service agents and service middleware in the context of service technology. According to the authors, both can be operated in a cloud-based environment. Service agents represent active or passive event-driven programs that take further actions, such as monitoring and reporting, based on the contents of intercepted messages. In contrast, service middleware, such as an Enterprise Service Bus (ESB) and an orchestration platform, is designed in order to accomplish tasks like service brokerage and service composition.

2.1.3 Related Paradigms

The idea of providing computing resources as a utility is not new (cf. Section 2.1.1) and several other computing paradigms (cf., e.g., [12, 25, 26, 45, 72, 130, 205, 209]) have been proposed, further approaching this vision prior to the emergence of the cloud computing paradigm [25]. Among these paradigms, service-oriented computing and grid computing are the most related to cloud computing and, thus, they will be elaborated in more detail in the following.

Service-oriented Computing The concept of delivering computing resources in the form of different “*as a Service*” models (cf. Section 2.1) already indicates a relationship of cloud computing to service orientation. Indeed, Service-oriented Computing (SOC), where services represent the main blocks for building applications, can be considered to be a reference model for cloud-based systems [26]. However, although sharing certain concepts, both paradigms differ. Therefore, a few research

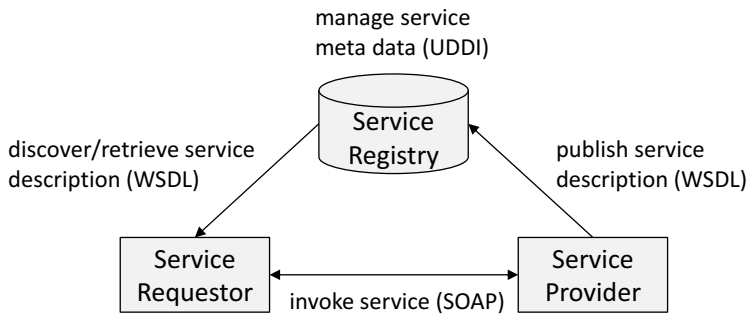


Figure 2.4: Interaction in an SOA using Web services (based on [54], p. 75)

works are discussing how cloud computing can benefit from service-oriented computing (cf. [45, 213]). The term service-oriented computing is often used synonymously with the term 'service-oriented architecture' (SOA) [55] and SOA is also perceived as a foundation for cloud-based systems (cf., e.g., [12, 209]). However, while service-oriented computing comprises several basic design principles around service-oriented solution logic, SOA represents an architectural paradigm supporting the realization of these principles [55]. The Organization for the Advancement of Structured Information Standards (OASIS) defines SOA as *"a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains"* and services as *"the mechanism by which needs and capabilities are brought together"* [145]. Hence, an SOA shall facilitate the integration and orchestration of multiple functional capabilities in the form of services from different organizations. In doing so, SOA is not bound to a specific technology, but often implemented using Web services (cf., e.g., [12, 55, 196, 209]). The basic interaction between organizations following the SOA paradigm is typically represented in the form of a triangle and depicted in Figure 2.4 assuming an implementation using Web service technology. In detail, a service provider may advertise a service by registering a corresponding service description in a service registry. The service can then be discovered by service requesters searching this registry. Having retrieved the service description containing the location of the desired service, the service can then be invoked by directly exchanging messages with the respective service provider. Basically, a service encapsulates a certain business logic and abstracts from the underlying implementation by providing well-defined interfaces resulting in autonomous units (cf., e.g., [53]). As a consequence, services can be loosely coupled in order to build larger business processes. Thus, SOA enables the creation of flexible business process infrastructures that can be easily adapted according to changing business requirements. Summarizing, SOA defines a uniform architectural model for offering, discovering, and interacting with distributed capabilities encapsulated as services and allows the creation of new

services from existing ones [145]. In comparison to SOC and SOA, cloud services also encapsulate a certain business logic as a re-usable, autonomous service, which is provided over the Web and expose their functionality through Web service technology, such as SOAP and REST (cf. Section 2.1.2). Hence, SOC and SOA are most related to the SaaS model. However, most of the current cloud applications are provided as entire (single-purpose) applications that do not require the flexibility that SOA provides (cf. [196]). Although following SOC and SOA principles is not mandatory in order to build cloud applications, SOA can serve as a blueprint, such that cloud applications can reach their full potential when being composed of loosely-coupled services (cf., e.g., [26, 196, 209]).

Grid Computing The *grid computing paradigm* was coined in the mid 1990s by Ian Foster and others and describes a distributed computing infrastructure that provides computing power on-demand, originally intended for resource sharing among scientific collaborations [70, 72]. According to Foster et al. [71], the underlying concept aims at “*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*”. A virtual organization, thereby, denotes “*individuals and/or institutions*” that adhere to clearly defined resource usage policies while sharing computing resources, such as computers, data, and software, as if these resources belong to the same organization [71, 72]. The description of the underlying concept of grid computing already reveals some characteristics shared with cloud computing. In a grid, different computing resources are also aggregated to large, logical resource pools with networked access (cf. [56]) and provided on-demand. In contrast to cloud computing, these resources generally belong to different administrative domains, which are geographically distributed and the resources are heterogeneous in capacity and dynamic so that their availability varies (cf. [72]). The architecture of a grid is based on the definition of standard protocols and the provisioning of middleware like the Globus Toolkit [69], considered to be the de facto standard implementation [72]. The middleware provides features such as load-balancing and autonomic configuration management and cloud computing is therefore perceived as a descendant of grid computing [56]. However, in comparison to cloud computing, grids have not initially been driven by commercial offers, so that grids are either build by sharing idle resources or users pay a fixed rate for each service [205]. Hence, in contrast to clouds, grids do not fully realize the idea of providing resources as a utility, similar to, e.g., electricity based on a pay-per-use model. Finally, hardware virtualization is also not part of the grid computing paradigm. As a consequence, grids do not support performance isolation, so that performance issues of one user or virtual organization can interfere with the performance of other users, which is one of the reasons that prevented grids adoption [209].

2.2 Service Level Management

2.2.1 Definition

Basically, Service Level Management (SLM) comprises specific activities of an organization with the aim to ensure an agreed level of quality for each of its provisioned IT services to service consumers [167]. In general, SLM is part of the IT governance of an organization. According to the IT Governance Institute, IT governance is the responsibility of the business management for specifying and managing structures and processes within an organization in order to align the organization's IT with the organization's strategy and objectives [100]. Hence, IT governance provides a generic framework for an organization's IT. Several concrete frameworks specifying best practices in the field of IT governance have been defined so far, among which the Information Technology Infrastructure Library (ITIL) is considered to be the de facto standard for IT service management. In ITIL, SLM is explicitly defined as a separate process of the service design phase, which encompasses the design of new services and the adaptation of existing services (cf. [19]). An official definition of SLM is provided by ITIL as follows [167]:

“The process responsible for negotiating achievable service level agreements and ensuring that these are met. It is responsible for ensuring that all IT service management processes, operational level agreements and underpinning contracts are appropriate for the agreed service level targets. Service level management monitors and reports on service levels, holds regular service reviews with customers, and identifies required improvements.”

It can be deduced from the ITIL definition, that two of the main activities of SLM are the negotiation and monitoring of service level agreements (SLAs), which are also the focus of the present work. Moreover, SLM aims to maintain and improve the quality of an IT service continually and, thus, also incorporates reporting on service levels, studying the generated reports with customers during service review meetings, and taking appropriate corrective or improvement actions if required (cf. [22, 167]). The reports are compiled based on measurements obtained from monitoring services and allow to assess whether the agreed service level targets have been met or not during a certain period of time. Besides SLAs, also agreements with internal and external providers have to be made in order to ensure compliance with the corresponding service level targets defined in an SLA. An internal agreement, denoted as Operational Level Agreement (OLA), are made between or within IT departments and external agreements, denoted as underpinning contracts, are made with external or third-party vendors or suppliers (cf. [22, 167]).

The generic activities performed during the lifecycle of an SLA are grouped into five different phases by Keller and Ludwig [109] as depicted in Figure 2.5. Although their

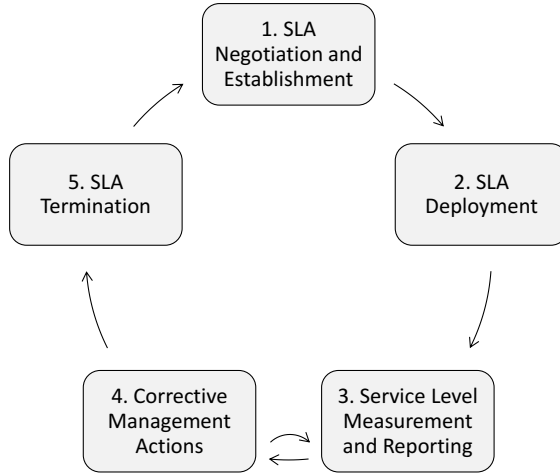


Figure 2.5: SLA lifecycle model (based on [109])

lifecycle model has been established in the context of Web services, it is applicable to IT services in general and to cloud services as well, since service technology is considered to be an enabling technology of cloud computing (cf. Section 2.1.2). The five phases according to Keller et al. [109] are described in detail in the following:

- *1. SLA Negotiation and Establishment:* An SLA is negotiated with regard to certain parameters and the resulting SLA document is made available for deployment.
- *2. SLA Deployment:* The validity of an SLA is checked, configuration information is accordingly generated, and the system configures the respective components.
- *3. Service Level Measurement and Reporting:* SLA parameters, such as availability or execution time, are computed by retrieving corresponding metrics, the measured parameters are compared against the agreed thresholds specified in the SLA, and the associated management system is notified.
- *4. Corrective Management Actions:* In case, that an SLA violation has been detected, corrective actions are executed in order to resolve the violation.
- *5. SLA Termination:* An SLA is terminated, either when certain conditions are satisfied or an expiration date has been reached as specified in the SLA.

The five phases are considered to form a loop in this thesis, since consumer requirements may change, which makes it necessary to negotiate new SLAs for a given

service or service properties may change allowing to provide improved SLAs to consumers. In addition, corrective management actions performed in the fourth phase may either be successful or not. Depending on the preferences of the contractual parties, the system can proceed with service level measurement and reporting afterwards or an SLA can be terminated.

2.2.2 Service Level Agreements

As described in the previous section, SLAs are the focus of SLM in order to guarantee a certain level of quality for a given IT service. Basically, different definitions exist for the term *service level agreement*. According to Marilly et al. [128], for example, an SLA can be defined as follows:

“A Service Level Agreement (SLA) is a contract between Service Providers or between Service Providers and Customers that specifies, usually in measurable terms, what services the Service Provider will furnish and what penalties the Service Provider will pay if he cannot meet the committed goals.”

Hence, an SLA is a binding agreement that specifies what a service provider guarantees to deliver and can be offered to other service providers or to consumers. Furthermore, these guarantees have to be defined in such a manner that the adherence to these guarantees can be monitored later on during service usage. Therefore, the goals represented by these guarantees are usually expressed in the form of thresholds for certain parameters, so that the current values of the corresponding parameters can be determined at runtime. Often, financial penalties are imposed for the case that a service provider deviates from an agreed SLA. A more detailed definition of an SLA can be found in ITIL (e.g., [22, 167]):

“An agreement between an IT service provider and a customer. The SLA describes the IT service, documents service level targets, and specifies the responsibilities of the IT service provider and the customer. A single SLA may cover multiple IT services or multiple customers.”

It can already be deduced from the definition in ITIL that SLAs can be defined in different ways. Indeed, ITIL further distinguishes *customer-based SLAs* that are defined for a set of services of a certain customer and *service-based SLAs* that are defined for all customers using a specific service (cf., e.g., [19, 22]). Furthermore, ITIL mentions a combination of these two types resulting in a so-called *multi-level SLA approach*, in which elements of services that concern all customers are defined in a *corporate-level SLA*. Besides these different types of SLAs, the definition in ITIL also already indicates that an SLA consists of different parts incorporating the service itself,

service level targets as well as responsibilities of the contracting parties. Among these parts, the part addressing service level targets, also often referred to as service level objectives, is denoted as Service Level Specification (SLS). According to Marilly et al. [128], the SLS contains metrics and corresponding thresholds to be guaranteed for technical parameters, such as network availability, latency, or throughput, and specifies according measurement methods, periods, and provided reports. The authors further mention the possibility to define an SLS template and to instantiate this template for a specific customer by setting the desired value for each threshold. Formal SLA specification languages are suitable for this purpose. In the field of Web services, different standards of such a formal language evolved for specifying SLAs in a machine-readable format as a prerequisite towards automated SLA management. The most popular ones are WSLA [109] developed by IBM and WS-Agreement [8] specified by the Open Grid Forum. In the field of cloud computing, automated SLA management is still an open issue (cf. Section 1.1). However, several approaches in research exist exploring the application of these Web service-related standards to cloud computing, for example, how WSLA can be adapted to a cloud computing context (cf. [152]).

2.2.3 Quality of Service

Service level targets specified within an SLA express some guarantees given by a service provider and usually concern technical, i.e., non-functional properties of a particular service (cf. Section 2.2.2). Hence, SLAs assure that a service exhibits a certain overall quality, also often referred to as Quality of Service (QoS). However, no single, common definition exists for the term QoS in general. Furthermore, QoS is not necessarily bound to software-based services and existing definitions mainly differ with respect to the considered application area. A generic definition of QoS in the field of communication networks is given by Schmitt et al. in [173]:

“QoS is the well-defined and controllable behavior of a system with respect to quantitative parameters.”

The definition by Schmitt et al. indicates that QoS goes beyond a set of well-defined quantitative parameters of a service. Moreover, these parameters can be controlled in such a manner that they lead to a desired behavior of a system. Another definition for QoS by the International Telecommunication Union (ITU) that has also been published in a standard provided by the International Organization for Standardization (ISO) is given as follows [101]:

“A set of quality requirements on the collective behavior of one or more objects. Quality of service may be specified in a contract or measured and reported after the event. The quality of service may be parameterized.”

The aforementioned definition reveals the relation of QoS to service quality. Following that definition, QoS implies a particular behavior of certain objects under consideration, similar to the definition by Schmitt et al., thereby fulfilling specific quality expectations. Moreover, QoS, and accordingly the desired behavior, can be described by the use of parameters, in terms of quality attributes expressing certain quality requirements. These quality attributes can be subject of a contract, i.e., part of an SLA, and can be monitored at runtime. Since the perception of QoS highly varies depending on the subject under consideration, several different QoS attributes have been considered in different application areas, such as communication networks (cf., e.g., [81]), multimedia (cf., e.g., [198, 199]), peer-to-peer systems (cf., e.g., [34, 92, 200]), and Web services (cf., e.g., [15, 158]). Since Web services are an enabling technology for cloud computing (cf. Section 2.1.2) and also realize the concept of Internet-based service delivery, the non-functional properties for Web services also apply to cloud-based services. However, there are some attributes that are either new or of major importance in the field of cloud computing. Thus, the Service Measurement Index (SMI) has been proposed by the Cloud Services Measurement Initiative Consortium in order to provide a set of standardized key performance indicators for comparing cloud-based services with respect to their business and technical requirements [42]. Besides non-functional properties concerning a service, the SMI also contains functional properties as well as attributes that apply to the cloud service provider. Therefore, a model for cloud-based services focusing only on non-functional properties of a service is presented in the following (cf. [89]). The model adapts the classification of non-functional properties for Web services by Berbner [15] and incorporates non-functional attributes of the SMI framework [42]. The resulting model of non-functional properties of cloud-based services is depicted in Figure 2.6 and the different properties will be elaborated on in more detail in the following.

Performance When it comes to performance in the context of computer systems, a variety of metrics exists (cf., e.g., [103]). In the field of Web services, performance is expressed by the speed in order to complete a service request [157]. Usually, that speed is represented in terms of the amount of time required from sending a request until receiving the response, denoted as *response time*, and the number of successful requests within a certain time interval, denoted as *throughput* (cf., e.g., [132, 157]). Response time, in turn, can be further split into *execution time* and *latency* (cf., e.g., [133]). While execution time encompasses the time for processing a request on the server side, latency comprises the time required for a one-way delivery of a message, i.e., a request or response, over the network (cf., e.g., [133, 150, 157]).

Assurance This category incorporates attributes that express the degree to which a service will behave as expected [42]. Concerning non-functional properties, *availability*, *reliability*, and *robustness* are usually considered (cf. e.g., [42, 80, 150, 157]).

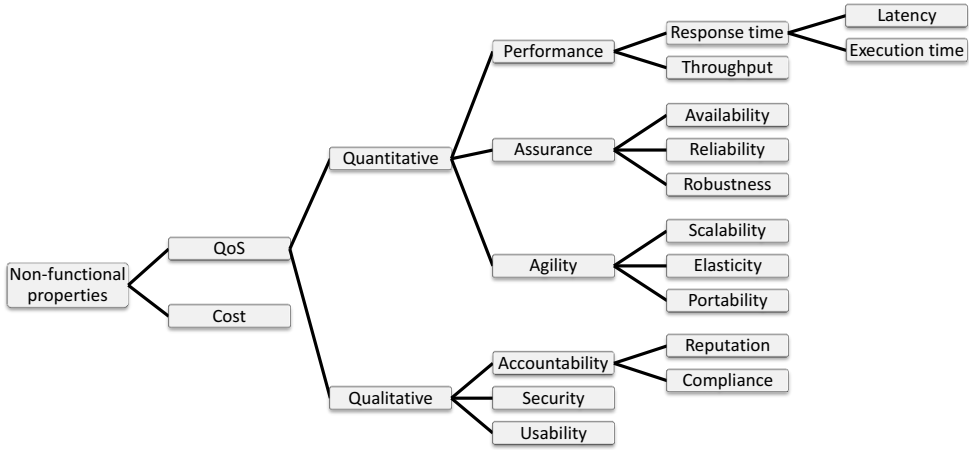


Figure 2.6: Non-functional properties of cloud-based services (based on [15, 42, 89])

Availability represents the probability that a service is up and running and can serve service requests. Basically, different metrics exist to express availability, such as the percentage of uptime or mean uptime (cf., e.g., [74, 80, 103, 132]). Reliability indicates, whether a service is able to work correctly over a certain time interval. Hence, it is very much related to availability and can be measured by the number of failures over a certain period of time, the probability of failures, or the mean time between or to failures (cf., e.g., [80, 103, 157]). Robustness represents the ability of a service to continue working correctly under unforeseen circumstances. Such circumstances may occur, for example, due to unexpected input that is invalid, incomplete or interferes with other input [157]. In the field of computing, no common metric for robustness exists and each work defines its own measure focusing on different aspects [219]. In the present work, the terms robustness and resiliency are used interchangeably.

Agility In contrast to former computing paradigms, cloud computing provides a very high level of flexibility allowing for dynamic resource provisioning according to a consumer’s demand (cf. Section 2.1.1). In this regard, the well-known attribute *scalability* and the two new attributes *elasticity* and *portability* are mentioned in the literature and characterize this agile behavior. Basically, all these attributes are indicators for the ability of a service to adjust quickly to changing user requirements with minimal disruption [42]. Scalability expresses, whether a service is able to sustain increasing workloads while maintaining adequate performance at an acceptable cost in terms of, e.g., response time or resource consumption [18, 94]. Further dimensions of scalability, namely scale-up/-down and scale-out, are distinguished in the context of cloud computing (cf. Section 2.1.1). Elasticity represents the degree to which a service is able to adapt to changing workloads by automatically adding or removing

resources that closely match the consumers' demand at each point in time [94]. Hence, elasticity is closely related to scalability. In order to allow for scalability, additional resources are occupied or have to be provisioned and elasticity represents the ability to determine and to switch to that required amount of resources in an automated manner. Portability indicates, whether a service can be easily moved to a different cloud provider [42]. In cloud computing, the foundation for portability is established by the use of virtualization technology. However, while the option of moving VMs within a cloud is usually given, a successful migration to another cloud provider depends highly on the interoperability between the two clouds and still represents an obstacle in today's cloud environments (cf. [123, 156]). Nevertheless, several activities¹⁰ exist, which address the development of cloud standards.

Accountability Within the SMI framework, this category addresses properties concerning the cloud service provider (cf. [42]). Nevertheless, the two attributes *reputation* (in the SMI framework denoted as “ease of doing business” expressing client satisfaction) and *compliance* can also be applied to a cloud service itself. In this regard, reputation is a measure of collective trustworthiness in a community based on the individual referrals or ratings derived from personal experience or second hand referrals of the community members [108]. Hence, reputation reflects the degree of satisfaction of all consumers with a particular cloud service and their strength to recommend this service. Compliance indicates the adherence to legal and regulatory requirements, such as standards, policies, processes, and established SLAs (cf., e.g., [42, 80, 150]). It is obvious, that the level of compliance may also have an influence on the reputation.

Security Concerns related to security are considered to be major obstacles in cloud adoption (cf., e.g., [64]). This is mainly due to the fact that the utilization of cloud-based services is accompanied with a shift of responsibility to the cloud provider. Furthermore, cloud computing enables the portability of cloud-based services as well as related data to other locations. In this regard, virtualization has a major impact on cloud security. First of all, VMs are stored as a file, so that they could be moved to a different host or network without the knowledge of a consumer. Second, multiple VMs can be run on the same host, so that an attacker could gain access to other VMs on that machine [184, 203]. Solutions for cloud security basically aim to address concerns, such as access control, confidentiality, data integrity, as well as threat and vulnerability management (cf., e.g., [42, 80, 84, 132, 150]).

Usability This category represents how easy it is to use a service. It comprises attributes, such as accessibility indicating whether a service can be used by consumers

¹⁰ An overview of the activities can be obtained from <http://cloud-standards.org/>.

with disabilities, installability expressing the effort to get a service ready for use, and learnability representing the effort to learn how to use a service [42].

Cost In contrast to all other aforementioned attributes, cost is not considered to be a QoS attribute (cf., e.g., [80]). Cost further represents a separate important non-functional property when comparing different cloud-based services. In this regard, cost shares a competitive relationship with QoS in terms of customer or provider utility, since a higher quality is usually associated with a higher price, and vice versa. In general, several options for pricing in cloud computing are possible. Among them, pay-as-you-go and subscription pricing are the most widespread models in cloud computing today [120].

The model in Figure 2.6 distinguishes qualitative and quantitative non-functional attributes of cloud-based services. While the latter can be expressed using numerical values, the former are usually described in terms of existing or non-existing properties, unless a quantification of the corresponding properties is conducted. However, there currently exists no commonly accepted quantification method for the qualitative attributes. Hence, only quantitative non-functional properties and cost are considered in the present work. Numerical values are required in order to be able to compare the non-functional attributes of different services against each other when negotiating SLAs and to detect SLA violations later on in terms of deviations from the negotiated thresholds.

In the field of communication networks, the concept of QoS evolved, since the prevailing service in the Internet has been, and still is, *best effort*. That means that, all packets are treated equally without providing any resource guarantees, which is critical for applications that require a minimum amount of resources in order to operate effectively, such as video conferencing or Internet telephony [212]. In addition, the applications provided over the Internet ranging from simple mail services to business-critical, transaction-based SAP systems highly differ in their requirements with respect to the underlying network [173]. In order to enable QoS in the Internet, different architectures and mechanisms allowing for resource assurance and provisioning of different levels of service according to customer requirements have been developed [212]. Ferguson and Huston [63] point out that such mechanisms allowing to treat one customer's traffic differently than another customer's, can also provide economic advantages, which can be exploited by developing according economic models. The same applies for the provisioning of customized quality guarantees in terms of SLAs for cloud-based services (cf. Section 1.1).

2.3 Automated Electronic Negotiations

Negotiations are generally conducted in case of conflicting interests. In cloud computing, conflicting interests occur between consumers and providers, since consumers want to obtain cloud services that exhibit a certain level of quality at low costs, while providers aim to maximize their profit [185]. To date, dynamic negotiations of SLAs are not supported or only offered with restrictions by today's cloud providers [25]. However, automated negotiations have been considered in several research works in different areas, such as service-oriented computing (e.g., [43, 143, 169]), grid computing (e.g., [23, 121, 206]), and cloud computing (e.g., [6, 25, 216]), all envisioning market-based service provisioning.

2.3.1 Negotiation Mechanisms

Following other research works as mentioned above, an automation of negotiations can be realized by applying negotiation mechanisms. Such a negotiation mechanism usually comprises a negotiation protocol and negotiation strategies of the negotiating parties. Basically, a negotiation protocol specifies the rules for interaction, i.e., the types of participants, negotiation states, events changing states, and actions that are valid in particular states [104]. A selection of commonly-used negotiation protocols in the field of automated electronic negotiations (cf., e.g., [3, 43, 46, 117, 121, 161, 223, 225]) is depicted in Figure 2.7 and described in the following.

Take It Or Leave It This is a very simple negotiation protocol over a single round. One party proposes an offer with a fixed content and the other party can accept it or not [104].

Alternate Offers This is a multi-round negotiation protocol proposed by Rubinstein [166]. The parties alternate in making offers and counter-offers until the parties come to an agreement or decide to break off negotiations. This protocol serves as a foundation for many other works (e.g., [117, 190, 216]).

Extended Contract Net Protocol The Contract Net Protocol (CNP) as proposed by Smith [193] was originally designed for task distribution. Using this protocol, a central node specifies a task and offers the task to all contractor nodes in a system. The contractor nodes then respond with an offer and the central node assigns the task to the contractor node with the best offer. In contrast, the extended CNP, also being used in several research works (e.g., [3, 43]), combines a multi-round alternate-offer phase with a two-phase commit phase, so that proposals are initially accepted preliminarily, and then have to be finally accepted in order to reach an agreement.

Besides negotiation protocols, negotiation strategies define the sequence of actions planned to make during a negotiation [126]. Hence, negotiation strategies represent the decision models of the negotiating parties and are based on their objectives, their method to valuate proposals, and their tactics applied during negotiation. Usually, the negotiating parties' objective is to reach an agreement while maximizing their individual utility. Thereby, their reserve utility defines the lowest utility level, below which they are not willing to continue to negotiate. Tactics define how an offer is generated for a single issue based on a particular criterion and the tactics for all issues are combined to form a strategy [58]. In the present work, we use the terms 'tactic' and 'strategy' interchangeably, if a strategy only comprises a single tactic. Three different types of tactics are usually distinguished: *time-dependent tactics*, in which the negotiating parties have (individual) time limits and make concessions during negotiation based on that deadlines; *resource-dependent tactics*, in which negotiating parties also make concessions depending on the remaining amount of available resources; and *behavior-dependent tactics*, in which a negotiating party is not under pressure and imitates the tactics of its opponent in order to prevent being

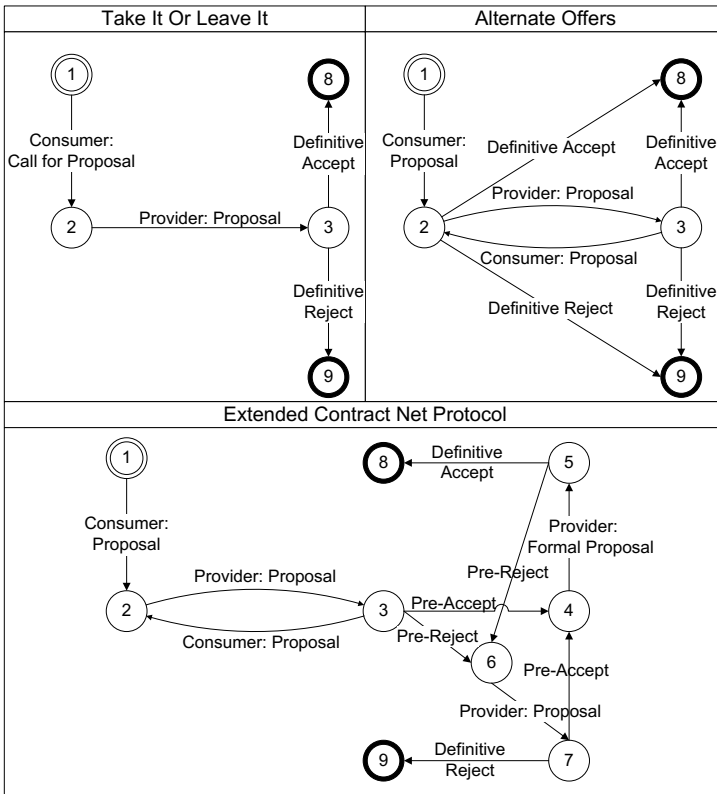


Figure 2.7: Commonly-used negotiation protocols

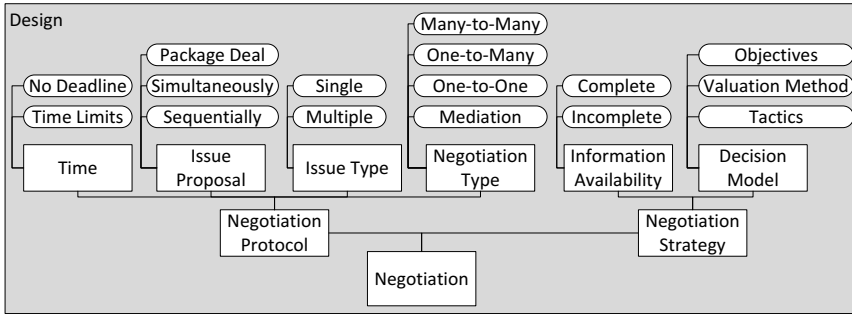


Figure 2.8: Aspects of negotiation design

exploited [58]. Such types of tactics are applied in case of self-interested, competitive parties negotiating without any information on their opponents' preferences while also keeping their preferences private. However, the negotiating parties may also entrust the task of determining an agreement to a third-party mediator, which usually obtains a part of their private information for that purpose (e.g., [143]). In case of a complete information setting, a mediator is able to determine an optimal solution. Hence, the quality of the negotiation outcome also depends on the amount of information that is available. A mediator-based negotiation is only one of the possible negotiation types. Depending on the number of participating parties, one-to-one (so-called bilateral), one-to-many, and many-to-many negotiations are typically distinguished, where the last two types can also be arranged as concurrent negotiations in a bilateral manner [126]. Negotiations can also be distinguished concerning the number of negotiated issues. Basically, a negotiation can concern single or multiple parameters (so-called issues), where the latter can be either negotiated sequentially, simultaneously, but independently of each other, or as a package deal considered as a bundle [62]. The aforementioned aspects of negotiation design are summarized in Figure 2.8.

2.3.2 Utility Theory and Pareto-Efficiency

During a negotiation, the negotiating parties have to decide whether a received proposal is acceptable to them or not. Furthermore, in case of multiple proposals, when negotiating over several rounds and/or with multiple participants, the negotiating parties must be aware of the conditions under which a certain proposal is preferred over the others. For this purpose, each negotiating party has its individual valuation method (cf. Section 2.3). Usually, utility functions, which assign a numerical value to each proposal, thereby expressing a user's utility, are applied as valuation method (cf., e.g., [59, 118, 121, 143, 168, 190, 195, 217, 222, 224]). In doing so, the concept of utility is used when comparing several alternatives in order to determine the level of satisfaction, denoted as utility, provided by each alternative [202]. Nevertheless,

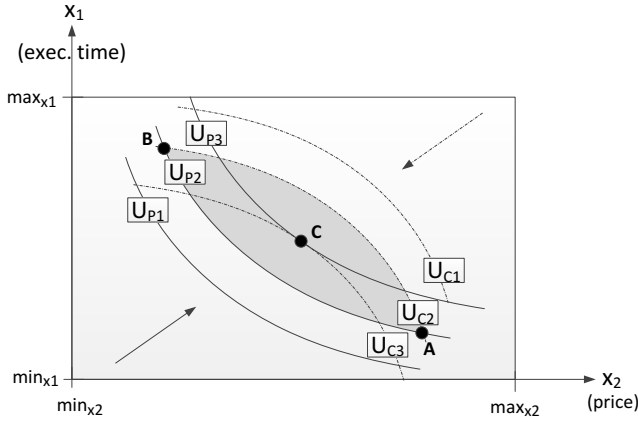


Figure 2.9: Pareto-efficient solutions and room for improvement

negotiations can also be conducted without having explicit utility functions. According to Chalamish and Kraus [30], the minimal requirement on a suitable valuation method is to express a user's preferences in such a manner that a party is able to compare and rank alternatives. However, the authors state that the advantage of using utility functions is that exact numerical values are associated with each result, so that they also indicate to which extent a certain result is preferred over another.

When negotiations are conducted over multiple issues, trade-offs are possible between the different issues. By making a trade-off, the value of a certain attribute is decreased while the value of another attribute is increased, so that the overall utility for one or both negotiating parties increases [59]. As a consequence, several offers with different attribute values can be composed that exhibit the same utility from a proposing party's point of view, but provide different levels of utility to the opponent. In microeconomics, this situation can be expressed by using indifference curves. Basically, an indifference curve expresses the preferences of a certain party and represents different combinations of attribute values where each combination provides the same level of utility to that party (cf., e.g., [214]). In this regard, the Marginal Rate of Substitution (MRS) mathematically describes the exchange ratio of the attributes. In detail, at a given point of an indifference curve, the absolute value of the slope of the curve corresponds to the MRS (cf., e.g., [214]). Now, in order to compare different solutions against each other, the notion of Pareto efficiency is usually applied. Basically, a solution is denoted as Pareto-optimal or Pareto-efficient, if there is no other solution that makes a party better off without making another party worse off (cf., e.g., [46, 126, 214]). In order to determine such efficient solutions, intersection points of the negotiating parties' indifference curves can be analyzed. This demands convex indifference curves, which are usually assumed (cf., e.g., [214]). Such

a situation is illustrated in Figure 2.9, in which the solid and dashed lines represent the indifference curves of the provider and consumer, respectively. Each indifference curve thereby expresses a certain level of utility provided to the consumer (i.e., U_{Ci}) or to the provider (i.e., U_{Pi}). Assuming, for example, that a consumer negotiates with a provider over the two attributes execution time as x_1 and price as x_2 of a given service. In that case, proposals on lower indifference curves (i.e., a lower execution time and a lower price) would be more beneficial from a consumer's perspective and proposals on higher indifference curves (i.e., a higher execution time and a higher price) would be preferred from the provider. Hence, both parties have to make concessions in order to reach an agreement. Although, the solution points A and B are intersection points of a consumer's and provider's indifference curve, they do not represent an efficient solution. When moving from these points to other intersection points within the dark gray area, improvements are still possible for the consumer and provider. Solely in tangent points of the indifference curves, such as solution point C , no further improvement can be made, so that tangent points represent Pareto-efficient solutions (cf., e.g., [214]).

2.4 Monitoring

Besides the negotiation of SLAs, holistic SLM also incorporates measuring the values of the SLA parameters at runtime in order to assess whether a provider adheres to the agreed thresholds as specified within an SLA or not (cf. Section 2.2). The objectives and principles related to the corresponding task of service monitoring are explained in detail in the following. Prior to that, a definition of the term *monitoring* is applied.

2.4.1 Definition

As with the term QoS, no single, common definition of the term monitoring exists. However, there are certain activities that most of the existing definitions have in common. In the field of distributed systems, Dodd and Ravishankar [47], for instance, define monitoring “*as the measurement, collection, and processing of information about the execution of tasks in a computer system*”. Hence, monitoring deals with the observation and analysis of certain activities in a system. A more detailed definition concerning monitoring for performance analysis in computer systems is provided by Jain. According to Jain [103], monitoring is a key activity in performance measurements and monitors, that are applied for this purpose, can be defined as follows:

“A monitor is a tool used to observe the activities on a system. In general, monitors observe the performance of systems, collect performance statistics, analyze the data, and display the results. Some also identify problem areas and suggest remedies.”

It can be deduced from the definition by Jain, that the activities related to monitoring may also incorporate the presentation of results, the detection of problems, as well as the proposal of appropriate countermeasures. An even broader definition is given by Brown [24] in the context of business process monitoring in SOAs. Brown considers the following three activities as essential when it comes to monitoring in general: gathering data, identifying deviations from benchmark standards, and initiating a response in case of a deviation. When comparing the aforementioned definitions against the SLA lifecycle model (cf. Section 2.2), the term *monitoring* corresponds to at least the service level measurement and reporting phase and also incorporates the phase of corrective management actions in the definitions presented by Jain and Brown. However, while the former phase focuses on observing and assessing service levels, the latter already considers control actions and is therefore not addressed by several definitions. In the present work, we also concentrate on monitoring activities related to the first phase and follow the definition by Molina-Jimenez et al. [135]:

“Monitoring of contractual SLAs involves collection of statistical metrics about the performance of a service to evaluate whether the provider is delivering the level of QoS stipulated in a contract signed between the provider and the consumer.”

The definition by Molina-Jimenez et al. specifies the data to be collected in more detail as “*statistical metrics about the performance of a service*”. These metrics typically correspond to certain QoS parameters of a given service. In general, the QoS parameters to be monitored differ depending on the application area and the object under investigation as outlined in the next section.

2.4.2 Monitoring Objectives in Cloud Computing

As already mentioned in the previous section, monitoring activities provide particular information about certain activities in a system and, thus, are a prerequisite for certain management tasks. In cloud computing, monitoring is conducted for several reasons in different areas as outlined in the following [1, 60]:

Capacity and resource management Cloud services require at least a certain amount of resources in order to work properly. Moreover, in order to provide some guarantees on the performance of services in terms of SLAs, appropriate capacity and resource planning is required [91, 180]. In this regard, monitoring the relevant parameters at runtime enables prediction of the need for additional resources or to determine wasted resources [60]. Metrics relevant in this area may incorporate the current utilization of memory, CPU, and storage.

SLA management Measuring QoS parameters during the lifecycle of an SLA is necessary for a cloud provider in order to assure compliance with the negotiated thresholds (cf. Sections 2.2.1 and 2.4.1). Furthermore, QoS assurance is essential for customer satisfaction and business continuity [60]. QoS parameters of cloud services are listed in Section 2.2.3.

Accounting and billing Measured service as one of the essential characteristics of cloud-based services (cf. Section 2.1.1) already indicates the need for metering capabilities that allow to track cloud usage and to design specific pricing models [1, 60]. In this regard, monitoring provides the means to capture the resource consumption of virtual resources and to measure specific properties of an application, such as used bandwidth and compute hours [49].

Fault and performance management Due to the high complexity of a cloud infrastructure comprising several layers and components, the detection of faults or performance issues and their root cause is a big issue [1]. A reliable and timely monitoring system not only enables cloud providers to locate faults and performance issues inside their infrastructure, but also allows customers to determine whether a failure or performance concern is caused by the cloud provider, the network, or the application itself [162]. Common metrics comprise the current load of a system and the availability of its components.

Security management Security is still considered to be one of the major obstacles in cloud adoption (cf. Section 2.2.3, [1]). Therefore, monitoring is required in order to support cloud security management by, e.g., detecting security breaches [60].

2.4.3 General Monitoring Principles

Besides the requirements derived from the application area, the design of monitoring systems also depends on the object under investigation. Based on the characteristics of the object to be observed, the following questions have to be answered in general [135]:

1. Should the collection of metrics happen in an active or passive manner?
2. From what points in the system should the metrics be collected?
3. Who should collect the metrics?
4. What information should be gained based on the collected metrics?

Concerning the first question, an active collection of metrics implies intrusive monitoring, while a passive collection allows for non-intrusive monitoring (cf., e.g. [140]).

This is due to the fact that active measurements are conducted by, e.g., actively sending synthetic requests to a component in order to gather information (also denoted as probing), and passive measurements are performed by, e.g., observing already existing messages exchanged in a system (cf., [135]).

The second question deals with monitor placement, i.e., the number of monitors to be used as well as their location. Basically, the monitor location is closely related to the location where the object under investigation resides, e.g., whether a monitor should be placed within the network in order to conduct network monitoring, and the perspective from which the object should be observed, e.g., from the provider's or the consumer's side. Regarding the number of utilized monitors, three approaches can be distinguished in general [24]: Monitoring can be either conducted at one point or at multiple points with or without correlation. While more information may be gathered when monitoring at multiple points, the monitoring tasks must be synchronized in order to achieve a reference to time.

The third question depends on who is interested in, or responsible for collecting the data and who has the expertise in collecting the data or access to the observed objects. Furthermore, for the case that data collection is performed by a third-party, the third question also depends on whether to trust that party or not.

Finally, concerning the fourth question, the information that can be obtained based on the collected data mainly results from the design of the monitoring system itself and from an appropriate elicitation of the relationships between collected metrics.



<http://www.springer.com/978-3-658-18772-9>

Service Level Management in Cloud Computing
Pareto-Efficient Negotiations, Reliable Monitoring, and
Robust Monitor Placement

Holloway, M.

2017, XVII, 250 p. 45 illus., Hardcover

ISBN: 978-3-658-18772-9