2021

Autonomic Management of Service Level Agreements in Cloud Computing

Frey, Stefan Edwin Karl

http://hdl.handle.net/10026.1/16887

http://dx.doi.org/10.24382/517

University of Plymouth

UNIVERSITY OF
PLYMOUTH

Autonomic Management of Service Level Agreements in Cloud Computing

By

STEFAN EDWIN KARL FREY

A thesis submitted to the University of Plymouth in partial
fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

NOVEMBER 2020

Dedication and Acknowledgements

This work was made possible due to funding from the Federal Ministry of Education and Research of Germany and the Faculty of Computer Science at Furtwangen University. I wish to thank both organisations for their support. I would like to thank all team members at the Cloud Research Lab Furtwangen and the Plymouth CSCAN group for their support.

I would like to express my special appreciation and thank my Director of Studies, Dr. Christoph Reich for his tireless effort in steering me through the PhD process and for encouraging my research and for allowing me to grow as a research scientist. Thanks also go to my supervisors, Dr. Nathan Clarke and Dr. Martin Knahl, who have spent a lot time proof reading papers and my thesis, in addition to providing helpful experience and guidance throughout my studies. Additionally, i would like to thank all my colleagues at Furtwangen University for the excellent working atmosphere, their professionalism and support.

A very big thank you goes to my Mum and my Dad for supporting me throughout my entire life, as well as my entire family.You are the source of my joy and happiness, and words cannot express how grateful I am. Lastly, I would like to thank all of my friends who supported me during this time and inspired me to strive towards my goal.

The research topic "Autonomic SLA Management as a Service (ASLAMaaS)" presented in this thesis, was supported by a research grant from the German Federal Ministry of Education and Research (BMBF) within the FHprofUnt2012 program. The grant was issued for the period of three years, starting October 2012 and is referenced under the grant number 03FH046PX2.[1]

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award, without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

This study was financed with the aid of a research grant from the Federal Ministry of Education and Research (BMBF) of Germany.

Relevant seminars and conferences were regularly attended at which work was presented and several papers prepared for publication, details of which are listed in the appendices.

Word count of this PhD thesis: 48028

Signed .... Date 30.11.2020

Abstract

Stefan Edwin Karl Frey

Autonomic Management of Service Level Agreements in Cloud Computing

Cloud computing has been one of the major trending topics of recent years in the Information Technology (IT) industry. In contrary to the previous IT service delivery models, where all the services and resources were hosted locally, the idea behind cloud computing is to deliver computing resources or services on-demand over a network on an easy pay-per-use business model. This paradigm change included almost all areas of the modern IT landscape, be it the storage and sharing of data for example with services like Dropbox, Microsoft OneDrive or Google Drive or creating complete computational environments with Amazon Web Services or the Microsoft Azure Cloud Platform. Due to the lower upfront costs, rapid provisioning, elasticity and scalability, the adoption of cloud services is steadily increasing. Thereby cloud computing nowadays for many companies has become a practical alternative to locally hosted resources and IT services. However, in the currently offered state of cloud computing, there are significant shortcomings in regard to guarantees and the Quality of Service (QoS). In order to make cloud services effectively usable and reliable for enterprises, Service Level Agreements (SLA) are needed, which state the precise level of performance, as well as the manner and the scope of the service provided. This practice, which is widespread in the area of IT services, but is currently of limited use for cloud computing, due to the fact that existing cloud environments offer only rudimentary support and handling of SLAs, if any. Moreover, the classic SLA management approach is a rather static method, whereas due to the dynamic character of the cloud, the QoS attributes respectively service levels must be monitored and managed continuously. In addition, performance indicators and measurement methods for service level objectives in cloud computing have been studied inadequately. The proposed research focuses on cloud specific SLA management as well as the associated functional and non- functional QoS parameters and measurement methods. To address the shortcomings of the current state of the art cloud computing SLA landscape and enable cloud users to dynamically create, adjust and monitor SLAs for their cloud services, an architecture for providing autonomous management of cloud services together with an easy to use SLA control interface is introduced, to monitor the agreed service performance, facilitate SLA compliance and enable dynamic customer- specific adaptation of SLAs for cloud services. Together with the specially tailored machine readable Adaptable Service Level Objective Agreement language (A-SLO-A), and highly the specialized scaling mechanisms, as well as a SLA scheduling module, the proposed research builds a holistic approach of enabling QoS and SLAs in Cloud Computing.

TABLE OF CONTENTS

Page

List of Tables IX

List of Figures XI

1 Introduction 1

1.1 Motivation . . . 1

1.2 Aim & Research Questions . . . 3

1.3 Outline . . . 4

2 Cloud Computing & Service Level Agreements 7

2.1 Cloud Computing . . . 7

2.1.1 Cloud Computing Advantages and Disadvantages . . . 10

2.2 Service Level Agreements . . . 12

2.2.1 SLA Life Cycle . . . 13

2.2.2 SLA Content . . . 14

2.2.3 Service Level Objectives . . . 16

2.3 SLAs in the Cloud . . . 17

2.3.1 Current Cloud SLA Landscape . . . 17

2.3.2 KPIs for Cloud Services . . . 20

2.3.3 General Service KPIs . . . 21

2.3.4 Service Specific KPIs . . . 22

2.3.5 QoS Control in the Cloud . . . 23

2.4 Conclusion . . . 24

3 Critical Analysis of Prior Art 25

3.1 Infrastructure Measurements and Cloud Monitoring . . . 25

3.2 SLA Modeling and Description Languages . . . 26

3.3 Performance Prediction . . . 28

3.4 Scheduling Mechanisms . . . 29

3.5 SLA Enforcement . . . 31

3.6 Autonomic Computing . . . 32
3.7 Conclusion . . . 32

4 Autonomic Cloud Computing SLA Management Approach 33
4.1 The Autonomic Computing Approach . . . 33
4.2 Autonomic SLA Management Architecture . . . 35
4.2.1 Fault tolerance and resilience . . . 43
4.3 SLA Creation Process . . . 43
4.4 Prediction, Detection and Management of Service Levels . . . 44
4.5 Autonomic SLA Management Evaluation . . . 45
4.6 Conclusion . . . 46

5 Implementation and Evaluation 47
5.1 SLA Frontend . . . 47
5.1.1 GUI . . . 47
5.1.2 KPIs and Infrastructure Data . . . 49
5.1.3 A-SLO-A . . . 49
5.2 QoS Management Modules . . . 60
5.2.1 Threshold Value System . . . 61
5.2.2 Machine Learning Algorithms . . . 70
5.3 Results . . . 84
5.4 Holistic SLA Management . . . 84
5.5 Conclusion . . . 87

6 Conclusion and Future Work 89
6.1 Achievements of the Research . . . 89
6.1.1 Key Contributions . . . 90
6.1.2 Limitations . . . 91
6.2 Lessons Learned . . . 91
6.3 Future Work . . . 92

References 95

Appendix A - Published Papers & Outputs 113

Appendix B - Further Background Information 115
History of Cloud Computing . . . 115
Cloud Computing Definition . . . 116
Cloud Models . . . 118
Reference Architecture . . . 120

List of Tables

TABLE Page

4.1 ASLAMaaS SLA Manager Components 40
4.2 ASLAMaaS QoS Manager Components 41

5.1 Infrastructure Service Parameter 72
5.2 Simulation Input Neurons 74

1 Cloud Computing Actors from NIST [2] 120

List of Figures

Figure **Page**

1.1 Worldwide Public Cloud Service Revenue Forecast (Billions of U.S. Dollars) Source: Gartner [3] . . . 2

2.1 Hype Cycle Phases . . . 8
2.2 Gartner Group Technology Hype Cycle 2014 [4] . . . 9
2.3 Gartner Group Cloud Hype Cycle 2018 [5] . . . 9
2.4 SLA Life Cycle . . . 13
2.5 SLA Structure . . . 15
2.6 WS-A and SLA(T) Structure . . . 18
2.7 Sample Agreement in WS-A and SLA(T) . . . 19

4.1 Autonomic Control Loop (MAPE-K) [6] . . . 34
4.2 Autonomic Computing Control Loop from [7] . . . 35
4.3 Early Iteration of the Autonomic SLA Management Approach . . . 36
4.4 Architectural Entity Relationships . . . 37
4.5 Information Flow between ASLAMaaS Modules . . . 38
4.6 ASLAMaaS Architecture Overview . . . 39
4.7 ASLAMaaS SLA Manager MAPE-K Loop . . . 40
4.8 ASLAMaaS QoS Manager MAPE-K Loop . . . 41
4.9 Flowchart of the ASLAMaaS SLA Manager . . . 42
4.10 ASLAMaaS SLA Management Frontend Overview . . . 44

5.1 First GUI Mock-up . . . 48
5.2 Final GUI . . . 48
5.3 SLA Violation Report . . . 49
5.4 Ganglia Cluster Overview . . . 50
5.5 Cluster Load Aggregation . . . 50
5.6 Model Based Development of SLO-A . . . 51
5.7 Overview of SLO-A Format . . . 51
5.8 Example of References within SLO-A . . . 52

5.9 Overview of SLA Template, a modified and extended SLA(T) model [8] 53
5.10 Overview of Agreement Terms 55
5.11 Pricing Business Term Classes 57
5.12 Business Level Guaranteed Action Classes 57
5.13 ServiceTime adjustment by the customer 60
5.14 Service Offering for CPUs 61
5.15 Threshold Value System 62
5.16 Fuzzy Controlled Scaling Architecture 63
5.17 Example: Weekly Load of the HFU Learning Management Platform 64
5.18 Input Fuzzy Set for Load Deviation 65
5.19 Simulator Module Diagram 66
5.20 Different Usage Load Graphs 68
5.21 Load Graph 68
5.22 Conventional Rule Set Results 69
5.23 Fuzzy & "High" Prediction Results 69
5.24 Fuzzy & "High" Prediction & Slope Results 70
5.25 Simple 3-tier Feedforward Multilayer Perceptron. 71
5.26 Feedforward Multilayer Perceptron Architecture. 71
5.27 Simulation Architecture. 73
5.28 IF THEN rules for threshold system. 74
5.29 Storage allocation results for threshold rules. 75
5.30 Storage allocation results for NN. 75
5.31 NN Scenario 1: 0s-100s 78
5.32 NN Scenario 1: 0s-20ss 78
5.33 SVM Scenario 1: 0s-100s 78
5.34 SVM Scenario 1: 0s-20ss 79
5.35 Linear Regression Scenario 1: 0s-100s 79
5.36 Linear Regression Scenario 1: 0s-20ss 79
5.37 NN Scenario 2: 0s-100s 80
5.38 NN Scenario 2: 65s-100ss 80
5.39 SVM Scenario 2: 0s-100s 81
5.40 SVM Scenario 2: 65s-100ss 81
5.41 Linear Regression Scenario 2: 0s-100s 82
5.42 Linear Regression Scenario 2: 65s-100ss 82
5.43 NN Scenario 3: 0s-300s 82
5.44 SVM Scenario 3: 0s-300s 83
5.45 Linear Regression Scenario 3: 0s-300s 83
5.46 Cloud Instances Abstraction 85

5.47 Holistic SLA Scheduling Model 86
5.48 Holistic SLA Scheduling Algorithms 86

1 Evolution of Cloud Computing from [9, p.4] 116
2 NIST Cloud Computing Model cf. [10] 117
3 Cloud Pyramid of Flexibility and Management 118
4 Cloud Computing Reference Architecture from NIST [2] 122

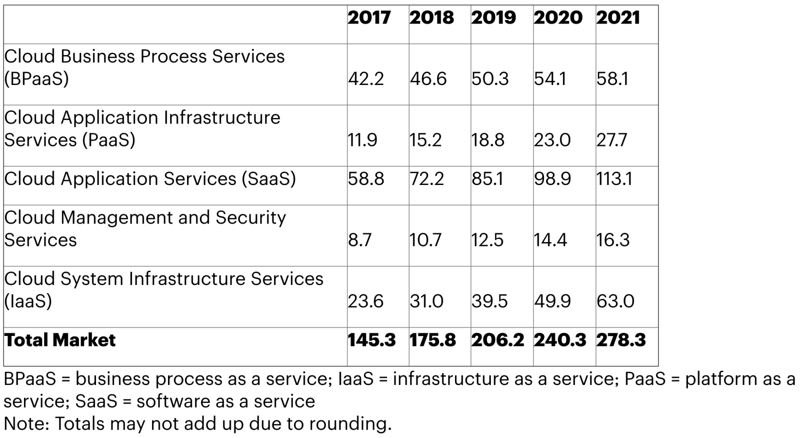
Chapter 1

Introduction

The following chapter starts with the motivation, why the autonomic management of service level agreements in cloud computing is a demanded capability (Section 1.1). Afterwards, the limitations and shortcomings of the current cloud computing landscape, as well as the aim and objectives of this thesis are summarized (Section 1.2). Finally, the structure is outlined (Section 1.3).

1.1 Motivation

Cloud computing has been one of the major trending topics of recent years in the Information Technology (IT) industry. A recent study about cloud adoption in Germany [11] has shown that 73% of the interviewed companies already use cloud solutions. In contrary to the older IT service delivery models, where all the services and resources were hosted locally, the idea behind cloud computing is to deliver computing resources or services on-demand over a network on an easy pay-per-use business model [10]. This paradigm change included almost all areas of the modern IT landscape, be it the storage and sharing of data for example with services like Dropbox [12], Microsoft OneDrive [13] or Google Drive [14], or complete computational environments like Amazon Web Services [15] or the Microsoft Azure Cloud Platform [16]. Nearly anything today is capable of being transferred to the cloud. As a result, the investment costs for IT infrastructures or services can be lowered, since it is now obtained from a cloud provider and not bought or provided by the user himself. At the same time, modern cloud providers possess an almost inexhaustible amount of computing resources pooled for their customers, so that nearly all requested amounts of resources can be provided within minutes. Due to the lower upfront costs, rapid provisioning, elasticity and scalability, the adoption of cloud services is steadily increasing [17]. Thereby cloud computing nowadays for many companies has become a practical alternative to locally hosted resources and IT services. According to the analyst view of Gartner, the worldwide public cloud services market is projected to grow by 17.3% in 2019 to total $206.2B, up from $175.8B [3]. This can be interpreted as a clear trend towards the cloud business.



	2017	2018	2019	2020	2021
Cloud Business Process Services (BPaaS)	42.2	46.6	50.3	54.1	58.1
Cloud Application Infrastructure Services (PaaS)	11.9	15.2	18.8	23.0	27.7
Cloud Application Services (SaaS)	58.8	72.2	85.1	98.9	113.1
Cloud Management and Security Services	8.7	10.7	12.5	14.4	16.3
Cloud System Infrastructure Services (IaaS)	23.6	31.0	39.5	49.9	63.0
Total Market	**145.3**	**175.8**	**206.2**	**240.3**	**278.3**

BPaaS = business process as a service; IaaS = infrastructure as a service; PaaS = platform as a service; SaaS = software as a service
Note: Totals may not add up due to rounding.

Figure 1.1: Worldwide Public Cloud Service Revenue Forecast (Billions of U.S. Dollars) Source: Gartner [3]

However, in the current offered state of cloud computing, there are significant shortcomings in regard to guarantees and the quality of offered services. But such guarantees for bought services are desperately needed, especially by enterprises, to make could computing effectively usable [18] and reliable [19]. For this purpose so-called Service Level Agreements (SLA) are needed, which state the precise level of performance, as well as the manner and the scope of the service provided. This practice, which is widespread in the area of IT services, is currently of limited use for cloud computing, due to the fact that existing cloud environments offer only rudimentary support and handling of SLAs, if any. Therefore most providers do not offer any kind of SLA or just generic versions of standardized guarantees, such as availability or service and help desk. For example, Amazon Web Services [15], as one of the biggest players in cloud services only gives availability guarantees for their Elastic Compute Cloud (EC2) [20], which is simply stated as if the achieved monthly availability is below 99.95% the customer will be given back 10% of the monthly fee in service credits. Additionally, up to 30% of the monthly fee will be credited if the availability falls below 99.0%. There are no further performance or quality guarantees given so far. This means that without even having to pay credits back the EC2 cloud service can be down for 21.56 minutes every month plus maintenance.

This is widespread for SLAs in the current cloud computing landscape [21] [22] and does not offer any sufficient protection for the customer. In addition, practically unusable services, due to poor performance, are not even taken into account. Arguably the compensation through service credits is in no proportion to the expected actual financial damage that a company can suffer due to poor availability, which can render this whole SLA meaningless [23]. Cloud users should be given the opportunity to configure related services according to their needs and to obtain customized guarantees. So that users in need can protect particularly important services. This means that cloud providers must be able to provide the functionality of custom SLAs and be able to guarantee the corresponding quality of service parameters and demand compensation.

1.2 Aim & Research Questions

In this section, the aim and the research questions addressed in this thesis are presented. Section 1.1 has shown the overview of the challenges motivating the research. Additionally, literature review on related work in the area of cloud computing guarantees, quality of service and Service Level Agreements shows, that the following limitations exist: The classic SLA management approach is a rather static method, whereas due to the dynamic character of the cloud, the QoS attributes respectively service levels must be monitored and managed continuously [24]. Performance indicators [25] and measurement methods for service level objectives in cloud computing have been studied inadequately [26].
The aim of the presented research has been to investigate the integration of SLAs into cloud computing environments to promote their reliability, transparency, trust and mitigate business concerns. Therefore cloud specific performance indicators, in dependence of cloud attributes such as, on-demand availability, elasticity of cloud resources and deployability were investigated. Following, an research of traditional SLA management and its ability to be integrated into the cloud has shown how there is need for changes. To address the identified problems autonomic management of SLAs for cloud environments has been proposed.

Therefore, we concretely address the following research questions:

- **How can Quality of Service Management and Service Level Agreements be adapted and built-into Cloud Environments?** For cloud computing, the quality and reliability of the services become an important aspect, as customers have no direct influence on the services. Due to the dynamic character and complex nature of the cloud environment, creating SLAs for the cloud can be very difficult. Therefore an adaptive autonomic architecture for cloud environments is needed.

- **How can a Cloud provider manage and guarantee individual Cloud SLAs and Service objectives?** Service requirements stated in Service Level Agreements need to be monitored and the corresponding resources need to be managed in order to guarantee them. In cloud systems, resources are being provided dynamically, which means the quality of a service can be directly dependent on the provisioning mechanism. Hence cloud computing services QoS monitoring, provisioning strategies, as well as detection and prediction of possible SLA violations must be proposed.

- **How can SLAs be enforced within the Cloud?** The fulfilment of the agreed upon SLA objectives heavily depends on the manageability of the Cloud resources in the environment. In order to efficiently provide theses resources an continuous autonomic control is necessary. Furthermore, in case of SLA violations an superior management is needed.
- **How can we prevent and minimize SLA violations in the Cloud?** SLA violations can happen especially due to varying demands and the up scaling delay of the infrastructure or economical limits. To minimize the number of SLA violations and to better guarantee the QoS, an behaviour, load and performance prediction based control model is needed. Additionally, a violation management model is needed to optimise the outcome in cases where a violation could not be prevented.

1.3 Outline

The remainder of this thesis is structured as follows.

Chapter 2 states background information to this study, starting with cloud computing its history, definitions and a reference architecture to build a common knowledge base. Furthermore, the foundations of Service Level Agreements, the process of SLA management as well as the guidelines on the content and preconditions will be given and the SLA life cycle will be introduced. Additionally, the autonomic computing paradigm is introduced and discussed. The current cloud computing SLA landscape will be illustrated, introducing cloud specific key performance indicators as basis for cloud service SLAs. Furthermore, the monitoring of cloud quality of service parameters and the controls and management will be elaborated.

Chapter 3 presents the related work, which is divided into 6 categories: Infrastructure Measurements and Cloud Monitoring, SLA Description Languages, Performance Prediction, SLA Enforcement, Autonomic Computing and Scheduling Mechanisms. The related works of each section is analysed and distinctions to the contributions in this thesis are drawn.

Chapter 4 describes the presented approach for autonomic SLA management in cloud computing, starting wiht the introduction of the architecture and its modules. Subsequently, the workflow from SLA creation to SLA monitoring and execution is illustrated to demonstrate the application of the autonomic SLA management.

Chapter 5- Implementation and Evaluations describes the developed prototype and its components. The Chapter follows the structure of the main research phases and presents the four consecutive development stages of the prototype:

- Stage 1 - "ASLAMaaS Front-end" - The graphical user interface to build Cloud SLAs and exported in the machine readable A-SLO-A language.

- Stage 2 - KPI respectively SLA monitoring and enforcement with various management and prediction techniques.
- Stage 3 - Cloud Simulator integration for evaluation and proof of concept.
- Stage 4 - Holistic SLA management based on provider prioritisation.

Each section ends with a demonstration of its specific part of the development stage and an evaluation based on three scenarios is given in the end .

Chapter 6 concludes the presented work and shows the limitations of the proposed solutions. Afterwards, the lessons learned during this study and possible future work arising from the research contributions of this thesis is described.

The Appendix shows the contributed research activities accompanying thesis with the published papers and reports.

Nomenclature within this thesis technical and legal terms are introduced at their first appearance. The terms cloud user, cloud consumer and cloud customer are used synonymously throughout this work.

CHAPTER 2

CLOUD COMPUTING & SERVICE LEVEL AGREEMENTS

This chapter considers the fundamental concepts and terms, which are being utilized in the following chapters. The aim is to give brief explanations and definitions that are necessary for further comprehension of the topic. The concept, history and the current state of cloud computing and presents its different cloud models are further described in Appendix B. Section 2.2 discusses the foundations of Service Level Agreements in the IT Industry, as well as to introduce structural and content-based pre-conditions and the SLA lifecycle. Section 2.3 illustrates the current use of SLAs in the Cloud and introduces KPIs. Section 2.4 summarizes the basic concepts and scientific classifications given in this chapter.

2.1 Cloud Computing

After an initial hype, cloud computing is establishing itself as a feasible means of providing resources or services on demand [27]. Never before in the history of computers has it been so easy for users to acquire enormous quantities of computational power and resources and put them to use almost immediately. Through this novel way computing users do not have to manage and maintain the IT assets they use and are no longer bound to the limited local resources they own. In comparison to the conventional approach of using IT resources, a user in cloud computing gets charged by the provider based upon the extent and time he utilizes or keeps the resource or service reserved. Through this pay-per-use model, and the rapid provisioning of cloud computing resources new opportunities in terms of dynamization, reduction of costs and investments emerge. The remainder of this chapter will point out the history of cloud computing and will indicate why this presents a relevant research topic in information technology. The introduction of a reference architecture for cloud computing and the different service models will create the technical background for this report. Finally, a discussion of the advantages and

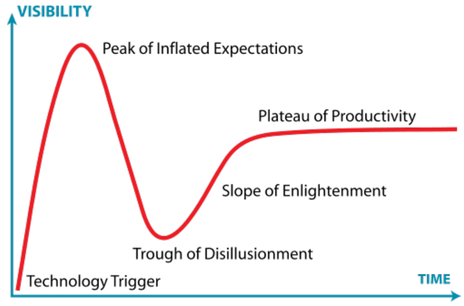


Figure 2.1: Hype Cycle Phases

disadvantages shall complete the state of the technology as it is.

As predicted by the Gartner Research Group [4], cloud computing is moving towards productivity and establishing itself as new means of getting IT resources and computational power. Figure 2.2 shows the Technology Hype Cycle for 2014, where it can be seen that after the initial peak interest and the following disillusionment cloud computing was expected to reach maturity and the plateau of productivity within the 2 to 5 years. In 2018 the Gartner Research published a stand-alone hype cycle for cloud computing [5]. This can be seen as a strong indicator of the increasing importance of cloud computing. In Germany, according to the German Association for Information Technology, Telecommunications and New Media (BITKOM) [28], in 2013 40% of all German companies were using cloud services. In 2018 this number has risen to 73% [11]. Especially large companies with over 2.000 employees increasingly rely on this technology. Overall cloud computing is immensely popular. Juniper Research [29] estimated the overall global cloud service users in 2013 to be 2.4 billion. Cloud computing is therefore a focal point for researchers in computer science.

As can be seen in Figure 2.1, a hype cycle consists of different phases, which range from an initial euphoria through a subsequent disillusionment to maturing and the productive use of a technology. While back in 2014 the Gartner Hype Cycle in Figure 2.2 shows Cloud Computing as a single point in the abyss of disillusionment, in 2018 Gartner published a stand-alone Cloud Computing hype cycle (as seen in Figure 2.3) to better represent all individual facets and uses of this core technology. So it becomes clear that cloud computing has now firmly established itself.

A further description and representation of the underlying technologies of cloud computing and their history can be viewed in Appendix B. For the further part of this thesis, these are considered to be known.

Related Technologies to Cloud Computing

As described earlier cloud computing has evolved from technologies that date back early in the history of computer science. Therefore it is related to different technologies by sharing similar

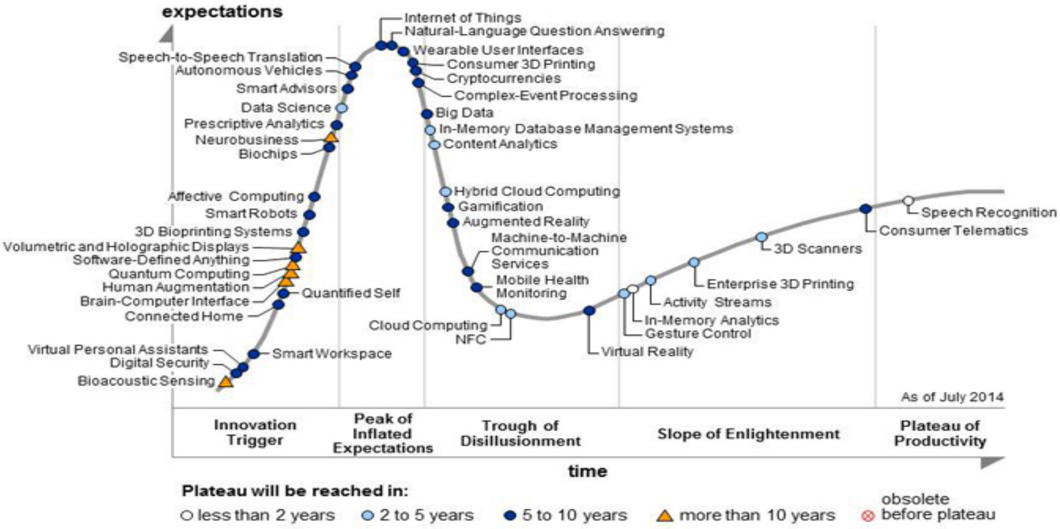


Figure 2.2: Gartner Group Technology Hype Cycle 2014 [4]

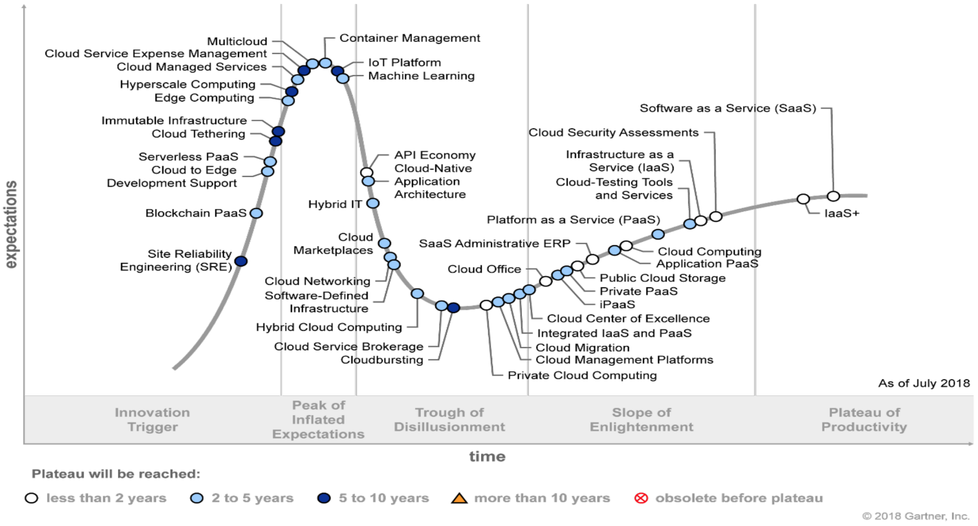


Figure 2.3: Gartner Group Cloud Hype Cycle 2018 [5]

aspects. Following the shared aspects with virtualization, grid computing, utility computing and autonomic computing will be shown.

Virtualization presents itself as one of the core technologies of cloud computing. Virtualization abstracts layers of physical resources from one another, thus making it possible to compose virtual systems out of virtual resources. Through the aggregation of large computational resources across multiple physical machines, virtual resources can be pooled by the provider. Therefore virtualization forms the foundation of cloud computing, as it provides the capability of assigning and reassigning virtual resources dynamically to cloud customers on-demand [30].

Grid Computing aims at creating supercomputer like computing resources by coordinating and utilizing distributed network-connected resources. Grid computing initially was driven by scientific applications like protein folding [31] or particle simulation [32] or the search for pulsars [33]. Grid computing always serves a specific task and all participating resources aim to achieve this common objective. Cloud computing appears very similar and shares a lot of the attributes from grid computing, for example, the use of distributed resources and abstracted utilization. But it differs in term of dynamic resource provisioning, the abstractions, the business model and applications [34].

Utility Computing contains the business model of "pay-per-use" for computing services. Much like for example, electricity the provided computing services are metered and the customer is billed based on his usage. Cloud Computing adopts this billing model and therefore can be seen as a realization of utility computing.

Autonomic Computing was coined by IBM in the early 2000s, where autonomic refers to self-managing computer systems. The aim was to create computing systems that were able to react and respond to internal and external observations without any human interaction. Cloud computing shares some of the aspects considering self-management, like autonomic provisioning of resources with autonomic computing. But instead of aiming for fully autonomous systems with for example, self-correction the aim in cloud computing is to reduce management involvement and improve the on-demand character.

2.1.1 Cloud Computing Advantages and Disadvantages

Cloud computing comes with many benefits, especially for small and medium sized business users. In this section, the advantages are described both from the perspective of the provider and consumer. As one of the major benefits of cloud computing the increased agility and adaptability can be seen. Here customers can take advantage of extending their IT capabilities based on the demand to fit the changing requirements. In addition, the customer does not have to manage or maintain the underlying infrastructure, which reduces his personnel training efforts and costs and therefore can stronger focus on his core strengths. Cloud computing also makes it possible to obtain enormous computing resources within very short times without having to bear the cost of acquisition. Due to this low up-front investments cloud computing offers a low entry barrier for young companies and simultaneously enabling rapid testing and development. For cloud providers using the virtualization technologies enables them to better utilize their infrastructure, by dividing the distributed physical hardware into virtual resources, since not all of the services in usually use the full provided resources. This reduces the overall Total Cost of Ownership (TCO) [35]. Because providing cloud services is the core competence of the cloud provider the offered services usually provide better performance and availability. Especially for SMEs it would be very expensive to reach the same high availability levels of modern cloud computing providers and depending on the usage cloud computing reduces the operating and maintenance cost of the customers services [36] [37]. This is due to the on-demand character of the cloud where the customer can reduce the rented resources of his services with low usage to a minimum. This reduces costs for the customer, while he still is being able to quickly adapt them back in case of increasing demand so that business risks of under-performance is reduced. In addition, the further risks like recovery due to outage or hardware failure is also moved to the cloud provider.

Besides the aforementioned benefits cloud computing also bears some disadvantages and risks. The main concern of many cloud users still is security [38] [39] [40]. Additionally, many users fear the loss of control over data or the infrastructure ([41], [42], [43]). This is due to lack of a proper, cloud provider independent security model. In addition the often unknown physical location of the rented resources (especially in public clouds) and the absence of cloud security standards and improper backups bare a risk of data loss. Besides that, the use of cloud computing does not always mean a reduction in operational costs. For permanent use classic hosting offerings are significantly cheaper. For example a streaming server based on a t3a.xlarge Amazon EC2 instance with 4 vCPU cores (AMD EPYC processors with an all core turbo clock speed of up to 2.5 GHz), 16GB memory, 500GB SSD, located in Frankfurt would costs around 140 USD [44]. In contrast, a conventional dedicated hosted server with 2x Intel Xeon E5 2.1+ GHz CPU, 16 GB of memory, 2x 1000GB RAID, with a 1 Gbit/s connection and unlimited traffic, would cost about 260 EUR [45]. Additionally, all cloud services are heavily network dependent, respectively Internet dependent. So the availability and performance of the connection is a core requirement for using cloud services. In case of missing or bad performing connections the provider is not necessarily responsible, since he only has to provide his services towards his edge of the network, so users have to ensue reliable connections.

An additional risk may be the dependency on the vendor and possible vendor lock-in. This occurs due to the lack of interoperability between cloud providers. Virtualization adds abstraction between the users and the physical hardware, but since not all virtualization technologies are interoperable moving from one cloud provider to another may be difficult. And depending on the local law, administrative regulations and the region legal concerns may arise [46].

2.2 Service Level Agreements

A service level agreement (SLA) in general is a formal bilateral legal contract between a provider of services and the consumer of such services, that defines the scope and quality expected that the provider commits to deliver to the customer. The Information Technology Infrastructure Library defines SLA as the following:

Service Level Agreement- A formal, negotiated document that defines (or attempts to define) in quantitative (and perhaps qualitative) terms the service being offered to a Customer. Confusion must be avoided over whether the quantitative definitions constitute thresholds for an acceptable service, targets to which the supplier should aspire or expectations that the supplier would strive to exceed. Any metrics included in a Service Level Agreement (SLA) should be capable of being measured on a regular basis and the SLA should record by whom. ITIL V2 [47]

The aim of such contracts is to create a reliable, legally binding arrangement from witch both the customer and the provider can benefit. The content of a SLA typically will cover general service related information like service hours and availability, but also will include the performance expectations and desired service outcomes in term of functionality and responsiveness. This also may include incurring costs, information about security and additional terminology. For customers, it offers the possibility to guarantee their expected service outcome and in case of non-compliance enable financial compensation. For providers, it states clearly all the agreed upon deliverables, as well as the method of delivery, occurring costs, verification and sanctions they have to comply to. So that the provider can deliver his services based on that and charge the customer without any discrepancies. The other side of the coin is, the provider needs to establish SLA management so that he can satisfy the contracts.

Since Service Level Agreements specify the promised respectively the expected performance characteristics of a service, the most important part is the exact description of the service quality (service level). But the creation of Service Level Agreements provides requirements to customers and providers. Customers need to be able to meet certain requirements in order to successfully define SLAs, which are listed briefly below [48].

A customer must:

- Understand the roles and responsibilities that are regulated by the SLA.
- Be able to describe precisely and in detail the service to be controlled by the SLA.
- Know the requirements of the controlled services, and define the matching key figures.
- Specify service levels based on the critical performance characteristics of the service.
- Understand the process and procedures of regulated service.

These requirements are necessary so that the customer is able to put in the correct SLAs values, and to understand the implications of his decisions. Furthermore, a SLA should fulfil the following tasks:

- Describe the services accurately.
- Specify the service quality to be provided in detail.
- Describe in detail the key performance indicators, metrics and service levels.
- Breakdown transparently all the costs.

2.2.1 SLA Life Cycle

The life cycle of a service level agreement involves several steps and is mandatory for the successful use of SLAs [49]. There are different views on whether the negotiation phase of the SLA is one of the stages of the life cycle or not, since this can also be counted among the preconditions (cf. [50] and [51]). Figure 2.4 shows the SLA life cycle derived from [52].

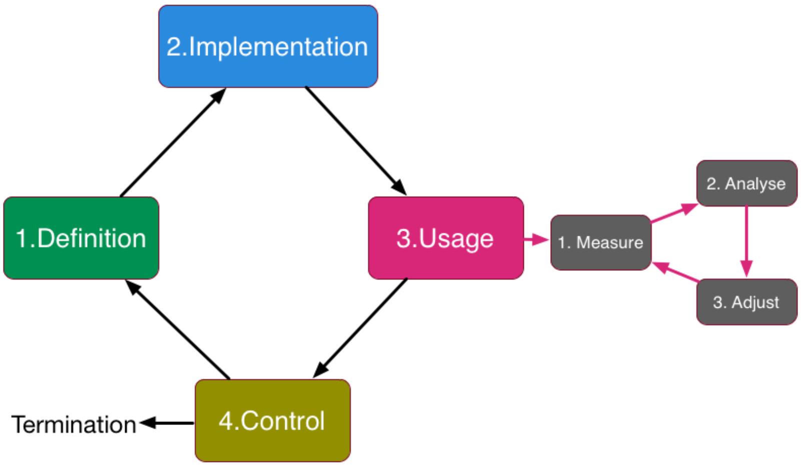


Figure 2.4: SLA Life Cycle

The preconditions for this life cycle is that the negotiation phase is integrated within the definition, where the deliverable service levels and the costs are negotiated with the provider. This negotiation process usually includes several rounds of proposals.

- **Definition Phase** Here the SLA content, which is agreed upon by both parties, is written down as the deliverable service level. Every aspect of the service, from creation to termination has to be stated precisely and must be accounted for. After this is done the contract is then signed by both parties.

- **Implementation Phase** Here the provided services are provisioned and the agreements are communicated and fitted into the organizations. The validity of the agreement is marked by the signatures of both partners.

- **Usage Phase** During the usage phase the customer uses the service according to the notions stated in the SLA. While in usage, the service performance is monitored and controlled by its own management cycle. Here during runtime service quality is assessed against the agreed service levels. If needed, corrective actions are executed and reports and documentations are created for the partners.

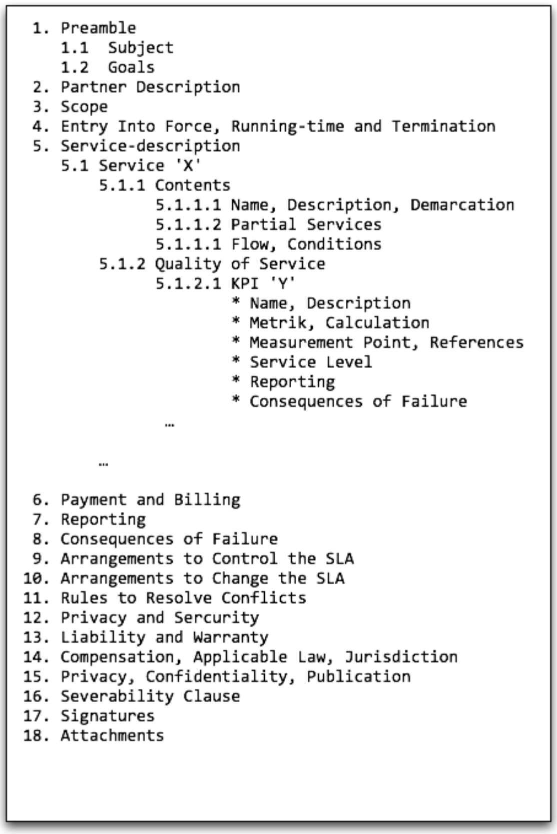
- **Control Phase** This marks either the end of the usage by the customer, if they are no longer in need of the service, which initiates the decommission of the service. Or the SLA is checked against the altered requirements and then a new definition process is started.

2.2.2 SLA Content

The structure of service level agreements are generally very scenario specific and can not be easily generalized. However, there are some basic elements that should be present in every SLA. The following remarks are not intended to be used to create a universal pattern for SLAs, but rather give an outline for most current contents of SLAs. The contents of a SLAs can be generally divided into the following four categories: (see [53]) *agreement-related elements, service-related elements, document-related elements and*
management-related elements.

The **agreement-related elements** contain the basic rules of the agreement and include, among others, the subject of SLAs, objectives, partners, as well as the scope, entry into force, duration and termination of SLAs. Often these elements are shown in practice in the form of a preamble or introduction. The subject of SLAs introduction here describes the content and context as well as a description and demarcation of the services being controlled by the SLA.The objectives of the SLAs reflect the specific objectives of both parties and serve, among other things, as a basis for future success control.

The **service-related elements** represent those elements which describe the regulation of a service. These must be specified individually for each service. The content is basically to describe who, when, where, and what services are provided. The description of the service should be generally understandable. The description of the quality of a service is the central role of the SLA. The negotiated quality of service is often represented by technical Key Performance Indicators (KPIs), which form the basis for the Service Level Objectives" (SLOs). These indicators include a label next to the calculation or metric, and a reference area and measurement point. KPIs also often target organizational objectives of the service provider and not only technical aspects. It should be noted that not every metric automatically relates to a Key Performance Indicator. KPIs are bound to organizational or services goals and must aim towards attainability.



```
 1. Preamble
    1.1  Subject
    1.2  Goals
 2. Partner Description
 3. Scope
 4. Entry Into Force, Running-time and Termination
 5. Service-description
    5.1 Service 'X'
        5.1.1 Contents
              5.1.1.1 Name, Description, Demarcation
              5.1.1.2 Partial Services
              5.1.1.1 Flow, Conditions
        5.1.2 Quality of Service
              5.1.2.1 KPI 'Y'
                      * Name, Description
                      * Metrik, Calculation
                      * Measurement Point, References
                      * Service Level
                      * Reporting
                      * Consequences of Failure
              …

        …

 6. Payment and Billing
 7. Reporting
 8. Consequences of Failure
 9. Arrangements to Control the SLA
10. Arrangements to Change the SLA
11. Rules to Resolve Conflicts
12. Privacy and Sercurity
13. Liability and Warranty
14. Compensation, Applicable Law, Jurisdiction
15. Privacy, Confidentiality, Publication
16. Severability Clause
17. Signatures
18. Attachments
```

Figure 2.5: SLA Structure

Document-related elements include administrative and editorial elements, which play a minor role inside a SLA and are mainly there to improve the handling, understanding and readability. These elements are, e.g., version, the date of last modification, revision history, table of contents, the index or glossary. These elements increase the readability by underpinning the context and explain the background.

The **management related elements** include the aspects that have to do with the administration and control of SLAs. These represent a very important section of the contents of a SLA, since both the customer notification and the procedure in case of problems or failures to meet the service levels are regulated. Furthermore, penalties and compensation in case of damage which may occur due to deviations from service levels are regulated.

Based on the presented elements, a generalized structure of a SLA has been created, which can be seen in Figure 2.5. Here, it is clear that the service descriptions, or service level objectives form the central aspect of each SLA. These and their contents are described in more detail in the following sections. Likewise, it comes clear that even small SLAs mean large administrative overhead and the creation is a lot of work.

2.2.3 Service Level Objectives

Service Level Objectives (SLOs) are a central element of every service level agreements (SLA), which include the negotiated service qualities (service level) and the corresponding Key Performance Indicators. SLOs contain the specific and measurable properties of the service, such as availability, throughput or response time and often consist of combined or composed attributes. SLOs should thereby have the following characteristics: cf. [54]

- **achievable / attainable** - Only objectives that can be achieved realistically may be chosen, otherwise the contract cannot be fulfilled.
- **measurable** - Objectives must be able to be measured otherwise no state or change in quality can be detected.
- **repeatable** - The measurement of the objective must be repeatable so actions, changes and adaptions can be documented and checked.
- **understandable** - The objective must be described in a way that they are general understandable, so even third parties can check them.
- **significant** - The objective must have a significant impact for the usage or quality and have a relation to the service.
- **affordable** - Providing the quality of service stated in the objective must be affordable for the provider, so there should be a limit.
- **controllable** - The providers actions must have a direct influence on the objective.
- **mutually acceptable** - Both provider and customer must agree on the same terms.

A SLOs should always contain a target value or desired service level, a metric and corresponding measurement period, as well as the type and location of the measurement. For this purpose, usually KPIs with associated service level values are stated. KPIs should contain information about the measurement process, measurement destination and unit as well. A valid SLO specification might, for instance, look like this: *The IT system should achieve an availability of 98% over the measurement period of one month. The availability represents thereby the ratio of the time in which the service works with a response time of less than 100ms plus the planned downtime to the total service time, measured at the server itself.* From such a description, the actual performance values can be compared with the reference values of the SLOs and the achieved availability result is calculated. Based on this, further measures can be carried out or correction measures can be conducted if necessary.

2.3 SLAs in the Cloud

As mentioned before, SLA contracts describe the exact service quality a user can expect, how fast a provider must respond in case of problems and what redress the provider has to give when the SLA contract gets violated and the user suffers a loss of business. SLAs, therefore, are the cornerstones of every IT service provider to reliable deliver services to his customers. According to a survey by Vanson Bourne Technology Market Research, commissioned by Compuware [55], German companies suffered heavy losses due to poor performance in cloud applications. More than half a million euros, is the average annual loss due to lack of SLAs according to the study. In order to make cloud services effectively usable [17] and reliable for enterprises [19], service level agreements (SLA) are needed, which state the precise level of performance, as well as the manner and the scope of the service provided. To include this functionality into existing cloud infrastructure specialized SLA management is needed. This section will introduce the current situation of SLA in cloud computing and gives an overview on related research efforts and introduce identified KPIs for the use within cloud SLAs.

2.3.1 Current Cloud SLA Landscape

As the usage of cloud service by companies continues to grow, the need for SLAs is increasing. NIST [2] has pointed out the necessity of SLAs, SLA management, definition of contracts, orientation of monitoring on Service Level Objects (SLOs) and how to enforce them. A basic discussion of SLA management and cloud architectures can be found in Service Level Agreements for Cloud Computing [26], but it is mainly concerned about SLA definitions and negotiations.

At present, most cloud computing providers only offer a generic SLA. Thereby guarantees for QoS characteristics like, bandwidth, data backup, etc. are given on the best-effort principle. Companies require QoS, monitoring and control of the cloud services at any time, as stated in the "Architecture of Managing Clouds" [18] and others (e.g., Study Group Report of Cloud Computing [56]). For cloud computing, the quality and reliability of the services has become an important aspect, as customers have no direct influence on the services. Therefore, service level agreements cloud be fundamental to an effective cloud utilization. For the users of cloud services, especially small and medium sized businesses, it would be very desirable to find a cloud provider who can guarantee the quality of the provided services by offering and enforcing SLAs.

Cloud infrastructures offer the potential to enable individual SLA negotiation and enforcement by adapting services during runtime, but this is currently not utilized. In the current cloud landscape large providers such as Amazon are currently not offering customer-specific SLAs and provide their customers with only rudimentary one size fits all general agreements. In the case of Amazon, for example, this means a guarantee to all customers of a availability of 99.95% for the EC2 service but for their Elastic Block Store (EBS) no service quality guarantee is given, which can lead to problems.

During the life cycle of a service the customer is often confronted with alternating and changing demands, which is often reflected in a change of the agreed service quality, emergency procedures, costs, legal compliance, and so on. The classical approach of negotiating SLAs is a very static based process. Unfortunately, this can not keep up with the dynamic character of cloud computing and therefore is unsuitable. For this new methods have to be created the cope with the fast-paced dynamics, but so far there is no solution on this field. Automation in term of SLA management would require SLAs to be presented in a machine readable form. In recent years, a significant amount of research has been performed on the standardization and creation of machine-readable formats. There are two major specifications for describing SLAs, WSAL [24] and WS-A [57].

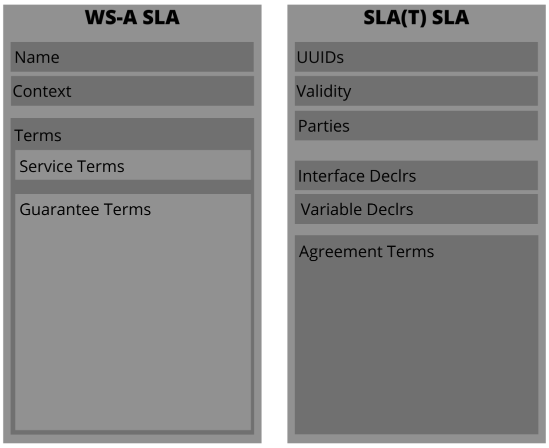
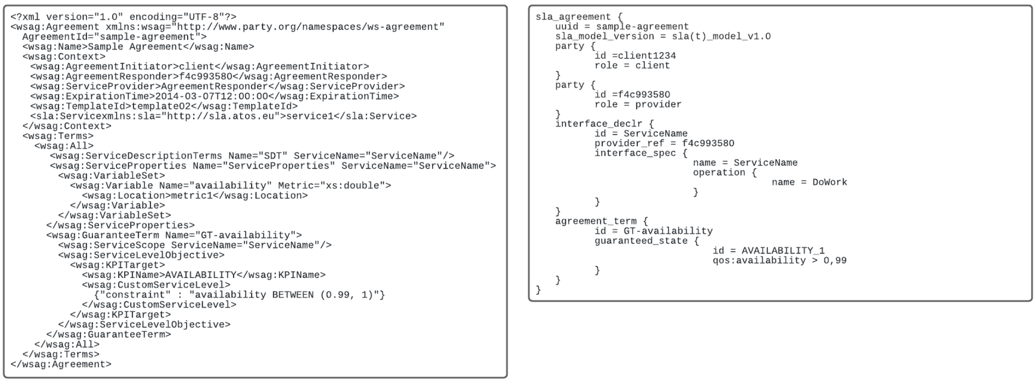


Figure 2.6: WS-A and SLA(T) Structure

The Web Service Agreement Language (WSLA) [24] was developed by IBM with the focus on performance and availability metrics. It has been mainly developed for Web services and the usage in other fields is questionable. It shows significant shortcomings regarding content as it was focused mainly on technical properties. WS-Agreement (WS-A) [57] was developed by the Open Grid Forum in 2007. The newest update, which is based on the work of the European SLA@SOI project, was done in 2011. Although it has been enhanced within the SLA@SOI project [58], the development is unclear, because the SLA@SOI project developed its own model SLA(T), which is supported by the European IT industry. Historically, the SLA(T) was developed as a generalisation and refinement of the web service-specific XML standards: WS-Agreement, WSLA, and WSDL. Figure **??** shows the content structure of WS-A and SLA(T). Instead of web services, however, the SLA model deals with services in general, and is representation language independent by offering an abstract syntax description instead of a specific language such as XML. The model, therefore, is based on vocabularies (e.g. for QoS metrics or constraints) and implemented as an abstract syntax that can be instantiated, in whole or in part, by an appropriate concrete syntactic format (e.g. XML, OWL, or human-readable formats), thus being language and technology independent. The model provides a specification content at a fine-grained level of detail, which is both richly expressive and inherently extensible: supporting controlled customisation to arbitrary domain-specific requirements. Both, WSLA and WS-A can be mapped in SLA(T) the other way round, however, this is not as easy to do.[59]. Figure **??** shows a simple sample agreement written in WS-A and SLA(T). This is mainly due to the fact that domain-specific extensions have to be modelled in the respective languages. For example, the recurring execution of a service based on a schedule cannot be directly mapped, but can be modelled using additional agreement terms like several execution time constraints.



```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:wsag="http://www.party.org/namespaces/ws-agreement"
  AgreementId="sample-agreement">
  <wsag:Name>Sample Agreement</wsag:Name>
  <wsag:Context>
   <wsag:AgreementInitiator>client</wsag:AgreementInitiator>
   <wsag:AgreementResponder>f4c993580</wsag:AgreementResponder>
   <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
   <wsag:ExpirationTime>2014-03-07T12:00:00</wsag:ExpirationTime>
   <wsag:TemplateId>template02</wsag:TemplateId>
   <sla:Servicexmlns:sla="http://sla.atos.eu">service1</sla:Service>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:ServiceDescriptionTerms Name="SDT" ServiceName="ServiceName"/>
      <wsag:ServiceProperties Name="ServiceProperties" ServiceName="ServiceName">
        <wsag:VariableSet>
          <wsag:Variable Name="availability" Metric="xs:double">
            <wsag:Location>metric1</wsag:Location>
          </wsag:Variable>
        </wsag:VariableSet>
      </wsag:ServiceProperties>
      <wsag:GuaranteeTerm Name="GT-availability">
        <wsag:ServiceScope ServiceName="ServiceName"/>
        <wsag:ServiceLevelObjective>
          <wsag:KPITarget>
            <wsag:KPIName>AVAILABILITY</wsag:KPIName>
            <wsag:CustomServiceLevel>
              {"constraint" : "availability BETWEEN (0.99, 1)"}
            </wsag:CustomServiceLevel>
          </wsag:KPITarget>
        </wsag:ServiceLevelObjective>
      </wsag:GuaranteeTerm>
    </wsag:All>
  </wsag:Terms>
</wsag:Agreement>
```

```
sla_agreement {
    uuid = sample-agreement
    sla_model_version = sla(t)_model_v1.0
    party {
            id =client1234
            role = client
    }
    party {
            id =f4c993580
            role = provider
    }
    interface_declr {
            id = ServiceName
            provider_ref = f4c993580
            interface_spec {
                        name = ServiceName
                        operation {
                                name = DoWork
                        }
            }
    }
    agreement_term {
            id = GT-availability
            guaranteed_state {
                            id = AVAILABILITY_1
                            qos:availability > 0,99
            }
    }
}
```

Figure 2.7: Sample Agreement in WS-A and SLA(T)

Although much research has been done in the direction of SLA formats, the contents of SLAs remain a further field for investigations. The fact that SLAs are always very scenario-specific makes it difficult to generalize their contents. KPIs, as a central component of service level objectives, are increasingly offered in KPI libraries. [60]. However, these are mostly rudimentary content and are not suitable for implementation. In order to describe the QoS of cloud services, metrics must fit the requirements and orientation of the to be measured service. Currently, there are no standardized cloud-specific KPIs and therefore fitting cloud KPIs standards have to be introduced. The International Organization for Standardization presented SLAs for cloud computing in 2018. There a definition of SLA content and usable metrics as part of the project ISO/IEC NP 19086-2 [61] have been publicized. Cloud SLAs have been covered in the ISO/IEC 19086-1:2016 Service level agreement (SLA) framework. Security and privacy overview and concepts are published in ISO/IEC 19086-4:2019. And the ISO/IEC 19086-2 Metric model establishes common terminology, defines a model for specifying metrics for cloud SLAs, and includes applications of the model with examples. In addition, problems can arise due to the distributed character of the cloud. If for example, a breach of contract occurs the jurisdiction may lie outside of the EU and has to be handled internationally. Therefore, analysts of the Experton Group [62] recommend German business customers to choose cloud providers with German contracts and service level agreements (SLAs), and local jurisdiction.

Additionally, current cloud services are hard to monitor for the customer, because none or only sparse information is given by the providers. However, for businesses, it is essential to monitor their services and check on compliance with their SLAs. Especially in cloud computing, due to the dynamic cloud character, the QoS attributes must be monitored and managed consistently. [63]

The negotiation of SLA deals with the technique of setting up the agreement between the provider and the customer.There are lots of effort going on in this area to address this issue like [64] [65] [66]. The area of SLA negotiations is not in the focus of this work. In this thesis, we mainly focus on developing techniques and mechanisms for SLA creation, management and enforcement.

Recently, different cloud providers have started offering additional services for their cloud offerings. For example, Rackspace [67] is now offering a managed cloud service with three optional service levels (Managed Infrastructure, Managed Operations - SysOps, and Managed Operations - DevOps Automation). These services mainly include arrangements that involve the support and help desk, such as 24x7x365 support systems with different response times or integration and security guidance. However, in the largest available configuration, the management of the backup service and the performance by scaling, as well as cloud monitoring is also included. This can be considered as the first step towards SLA integration into cloud services since thus cloud customers will be given at least some degree of control over the obtained services. However, if you look deeper into the offered service, it can be recognized that no state within respect to the quality of the performance that has to comply is stated. Likewise, it is not possible to negotiate individual service levels, there are currently only prescribed levels available by the provider. So it remains questionable whether this is sufficient enough to offer these services with enough dynamism and reliably for business users.

2.3.2 KPIs for Cloud Services

Together with industry partners the ASLAMaaS Project [1] identified over 90 possible cloud service KPIs. The utilized KPIs include, both general QoS parameters such as availability, response time or bandwidth, which are commonly used for almost all services today, but also cloud-specific KPIs, such as the deployment-times for PaaS, virtual image management or scaling schemes. This section will outline the structure of the proposed KPIs and introduce selected sample KPIs. For a complete overview and in detail description see [68].

The cloud KPIs are generally grouped in two domains, the Service Specific KPIs and General Service KPIs. The latter represent all the basic needs of each service has to run efficiently. Service Level Agreements must always be tailored to fit the service that shall be controlled. Nevertheless, there are some KPIs, which rules can be used in various SLA. These include, for example the availability, security aspects, service times and helpdesk, as well as monitoring and reporting. These are basic requirements for every purchased service. The Service Specific KPIs differ strongly depending on the cloud service model and comprised resources.

2.3.3 General Service KPIs

The General Service KPIs form the basic requirements every cloud service bares. They include the availability which is defined at the time the service is usable, the Mean Time Between Failure and Mean Time To Repair, which specify the time intervals at which to expect failures and how long it takes to repair them. Additional security KPIs regulate for example which software version levels shall be used, how long it should take until an update is implemented, as well as the scope and frequency of security audits. Other important KPIs control the encryption of data, the use and timeliness of anti virus software and the isolation and logging. Service and Helpdesk KPIs control the times at which assistance is provided, which support methods are applied or how many calls are received per week. Similarly, the qualification of the support personnel and the duration is given to problem solving. Finally, monitoring and reporting KPIs are used to define in which determined intervals performance date id to monitor and how to handle the resulting reports.

A sample KPI definition out of this group would be the availability, which is composed as follows: Availability of an IT system, the status of the error-free usage of the functionality of a system under specified conditions within a defined time frame. The error-free usage refers to the defined functionality of a system, such as sending of e-mails, so the service would already be unavailable if no e-mails can be sent, even though the system is reachable and the receipt of e-mails would be still possible. In practice, is the fact that services are switched off at certain times to perform, for example, maintenance or updates needs to be considered. Therefore, the availability is calculated as follows:

$$Availability = \frac{Servicetime + PlanedDowntime - UnplanedDowntime}{Total\,Time}$$

The *Servicetime* here refers to the time in which a system could be used without errors. This is also often referred to as operational time. The downtime is the non error-free time. A distinction is made between planned and unplanned downtimes. As planned downtime refers to maintenance, updates, and so on, which has been agreed on in advance with the customer. The total time indicates the reference period for the measurement in which this calculation is conducted. The benefits of such agreement is strongly dependent on the definition of the reference period. For example, an availability of 98.5% guaranteed for a reference period of one week results in a permissible outage of 2.52 hours per week. However, if the reference period is set to one year the allowed downtime would be about 5.5 days (131.4 hours) per year. In practice, the availability is usually provided in relation to one year. Another critical factor is the definition of the type of measurement. It is important to note that the wanted functionality of the system has to be imaged. For example, a simple ping or a request-response time below a certain limit can be considered "available". In complex distributed system there is a additional requirement that it has to be determined whether the availability of a individual component equals the aggregated system overall availability.

2.3.4 Service Specific KPIs

The service-specific KPIs are based directly on the service model of the services and its resources. Particularly for cloud computing, the network has a strong significance, as all provided resources and services are available through a network. Here, the **network** has to be considered both as a pure transmission medium for other services as well as independent service itself. For the described KPIs, the entry point of the provider network is usually chosen as a measure point, as the guarantees of the provider refer only to this area of the network. A sample KPI from the network group involves the *Round Trip Time*, *Response Time*, *Packet Loss*, as well as *Bandwidth* and *Throughput*. These are sole technical parameters.

Cloud Storage KPIs can be distinguished within cloud computing in two basic types. First, Storage as a service itself, that is obtained as a memory from pre-existing infrastructures. On the other hand storage can be used as part of another service such as a backup or data storage for cloud services. Typical KPIs here are the *Response Time* being the interval between sending a request to the storage and the arrival of the response at the output interface. Usually measured in milliseconds. The *Throughput:* Number of transmitted data per time unit. Here, a specified amount of data is transferred to the storage and measured the needed time from a given point. The size of the data set and package sizes are important factors for the validity of this measure. Furthermore, the network and its utilization must be considered. The *Average Read / Wirte Speeds*, which in contrast to the throughput, refers to an individual hard drive or hardware type. This value indicates how fast data can be read or written from/to the hardware. In RAID systems or virtual storage solutions, this figure is expected of interconnected hard drives. *Random and Sequential Input / Outputs per second (IOPS)*, which give the number of possible random / sequential input / output operations per second for different block sizes. The higher the IOPS value, the faster the disk. This value is also important to measure how many concurrent accesses can be handled by a system. More cloud relevant KPIs are the *Provisioning Type* and the *Average Provisioning Time*. The type of provisioning where at "thin provisioning" the client gets the storage not permanently assigned but it is dynamically allocated at runtime. In contrast, the thick-provisioned storage is allocated to the customer immediately and the time, the provider needs to provide a defined amount of data volume growth.

Backup and Restore KPIs refer to both the storage, i.e., the stored data, as well as services, for example, VMs or SaaS services. Here the *Backup Interval* is a major KPI together with the *Backup Type* . An exact specification along with the backup type and a description of the scoop has to be given to to the provider in order to protect from data loss. *Time To Recovery* specifies the minimum and maximum time from the failure of a storage, to the successful restore from an existing backups. Together with the *Backup Media* and *Backup Archive* they form a complete definition for cloud services.

Depending on the service model **Infrastructure as a Service KPIs** refers not only to the service itself but also to the virtual machines used. For this, additional VM KPIs are specified as well. For infrastructures the *VM CPUs*, meaning the number and type of CPUs used by the virtual machine hast to be defined. Additionally, information about the overbooking of the provided CPU resources shall be given. Here the shared resources are allocated with more capacity than is physically available. Thus, no real physical allocation of resources takes place. Actual performance is dependent on the overall consumption of the system. The KPI *CPU Utilization* enables performance management by being based on the proportion of CPU resources in use to the total number of resources provided per time unit. Also the CPU queue, which indicates the number of open requests to the CPU should be considered. Similar KPIs for the memory exist with *VM Memory* and *Memory Utilization* likewise here information about the overbooking of allocated memory resources should be stated. Cloud service type specific KPIs like the *Migration Time* and the corresponding *Migration Interruption Time* define the maximum time in which a customer has no access to a resource while migration. *Logging* gives retention of log data, which specifies how long log data to be stored by the provider and specification of what level to be logged. (e.g., INFO, DEBUG, etc.)

These KPIs presented here represent a small selection from the wealth of the identified potential cloud KPIs. However, the selection of a service-related and the specific management can not be generalized.

2.3.5 QoS Control in the Cloud

In order to guarantee the QoS in a SLA a provider must be able to control the respective QoS parameters. Otherwise a provider can only measure the performance of the service he offers but can not actively adjust its quality, which makes the offering of SLAs outrageously. The control of the relevant QoS parameters always requires the adaptation of the underlying infrastructures or its resources. So for example, the improvement of CPU utilization requires either the control over the virtual CPU resources provided or to the entire infrastructure. In the first case a simple alteration of the assigned virtual CPUs can change the utilization of the entire system. In the other case the overall utilization could be altered by adding additional VM instances which then get the occurring load distributed by a load-balancer. In both cases the service of the cloud user does not notice an continuously keeps running.

2.4 Conclusion

In this chapter the historic development and essential attributes of cloud computing were introduced, as well as a definition of cloud computing together with a reference architecture were given. The main cloud service and deployment models were presented and related technologies were introduced and delimited from cloud computing. Afterwards the advantages as well as the disadvantages of cloud computing were discussed. Additionally, the essential attributes and contents of service level agreements have been stated as well as a definition based on ITIL V2. The main SLA elements were discussed and the related life cycle was introduced and illustrated. The current Cloud SLA landscape and developments towards SLA for cloud services have been presented. The general SLA management process and its adoption to the cloud computing model has been outlined. It could be shown that SLAs and QoS in the cloud have not yet been implemented sufficiently extensively. This is due to the lack of translation of performance metrics into SLAs and to the magical management of such KPIs. Cloud providers are unable to provide customers with extensive and detailed SLAs, as these require automated management of the resources and relevant parameters for the metrics. In addition, the transparent and inter compatible description of such cloud SLAs poses a problem, as there are various standards for describing and disagreeing with the content and scope of such cloud SLAs. Although recent efforts have been made to standardize this, there is no general standard yet. Finally, a generalized structure and service level objectives were introduced. Along side with the here introduced self-management respectively autonomous systems , this forms the foundations for the system presented.

Chapter 3

Critical Analysis of Prior Art

The related work section presented in this thesis, focuses on the state of the art in Cloud Computing in regard to it's monitoring, scheduling and SLA management capabilities. Therefore this chapter was divided into five categories, namely: (1) Infrastructure Measurements and Cloud Monitoring, (2) SLA Modelling and Description Languages, (3) Performance Prediction, (4) Scheduling Mechanisms and (5) SLA Enforcement, as well as (6) Autonomic Computing.

3.1 Infrastructure Measurements and Cloud Monitoring

Cloud Computing today delivers nearly unlimited scalable on demand computing resources within a few clicks. But the monitoring and government of such resources may come with some requirements. Monitoring is one of the fundamental parts of every cloud platform, since monitoring is needed to scale applications or instances correctly based on their resource utilization. It is also needed to detect and discover defects and limitations, such as bottlenecks in the infrastructure environment. Additionally monitoring can give viable insights to both Cloud users and providers by revealing usage patterns and trends. Without any form of monitoring Cloud providers would not even be able to invoice their customer correctly, since they would not be able to tell how much resources and for how log the customer has used them. Cloud Monitoring has many ties into different Cloud Management fields such as capacity and resource planning and management, SLA management, incident management and troubleshooting, performance management and billing. For the presented approach, monitoring and measurements of cloud systems is an important foundation. Without reliable and continuous monitoring of the status of the resources and the system, neither suitable management nor reporting can take place. An early example towards distributed resources monitoring is NetLogger [69], a distributed monitoring system, which could monitor and collect network information. Published in 2000, application could use NetLoggers API to gather load information of the network and react accordingly. With GridEye [70] a service-oriented monitoring system, with an flexible, scaleable architecture and forecasting algorithm for performance prediction on the basis of traditional MA(k) and ExS(alpha) models was purposed. In 2009 Sandpiper [71] was proposed, a system, which automatically could monitor and detect hotspots and based on that remap or reconfigure VMs if necessary. The described system marks a first step towards autonomous SLA management, since the internal remapping algorithm considered SLA violations and to avoid them. In recent years various academic and commercial Cloud monitoring solutions have been proposed. In 2014 Jonathan Stuart Ward and Adam Barker [72] published a taxonomy of Cloud Monitoring stating scalability, cloud awareness, fault tolerance, aromaticity, comprehensiveness, timeliness and multiple granularity as core requirements towards effective Cloud monitoring. Fatema et al. [73] and Aceto et al. [74] analysed the most common open source and commercial monitoring solutions, such as Nagios, Collectd, Ganglia, IBM Tivoli, Amazon Cloud Watch, Azure Watch, PCMONS, mOSAIC, CASViD and many more, according on how they relate with these requirements. Recently Manasha Saqib [75] apportioned these requirements along the seven layers of Cloud Computing accoring to [76] [77] [78]: Facility, Network, Hardware, Operating System (OS), Middleware, Application and User. These studies provided a comprehensive overview of the available monitoring tools and their ability to support Cloud operational management, but also stated their shortcomings in the different areas. Cloud Monitoring in this thesis is considered as an enablement technology. The proposed architecture and algorithms are able to work with different monitoring solutions. The main focus thereby lies within the acquisition and enhancement ot such methodologies with additional information. An additional topic especially in the area of PaaS is the application monitoring and application performance monitoring in the cloud. The definition of application performance management by Menasce [79]: APM is a collection of management processes used by organizations to ensure that the QoS of their e-business applications meets the business goals. So APM directly aims at the application life cycle and management processes such as monitoring, resource management, performance-management, reporting and so on, of software systems. Tthere are several established big names in APM such as IBM, Oracle, BMC and so on, that are getting challenged by innovative start-ups such as AppDynamics or Dynatrace, and Correlsense. APM can be used as performance analysis and monitoring agents in SaaS and PaaS, and as such will be a topic for future work.

3.2 SLA Modeling and Description Languages

In order to enable dynamic and self-service-based, autonomous management of SLAs in the cloud, machine-processable and understandable SLAs are required. These have to be created in a formal model and a language to be universally understandable and reliable. Since these requirements also exist for regular electronic SLAs, there are already many approaches and projects aimed at this. SLA attributes and its criteria are investigated for cloud computing by Alhamad et al. [80]. This study determined the individual SLA metrics for SaaS, PaaS and IaaS separately. Although specific metrics are presented based on their layer requirements, they never considered the SLAs relation and hierarchical nature of cloud computing. In 2003 IBM developed the Web Service Agreement Language (WSLA) [24] which is still available in the Version 1.0 that dates back to the same year. It has not been further developed since. WSLA focused on performance and availability metrics. It seriously lacks expression therefore it is not powerful enough. The required flexibility is also missing which is needed for dynamical changes at run time. It is closely connected to their XML schema and not very useful to determine a conclusion. WSLA has been mainly developed for Web services, its usage in other fields is questionable. It shows significant shortcomings regarding content as it focusses mainly on technical properties. The structural requirements, however, are met as discussed in Spillner et.al. [81]. Patel et al. [82] proposed the WSLA framework for SLA enforcement in cloud computing. This framework is not enough adapted for cloud needs, cloud computing nature. Some other frameworks include RESTful [83], SALmonADA [84], SLA@SOI [58], which all have put aside the hierarchical nature of cloud computing. They have directly applied the SLA structure in the cloud computing area from SOA and grid computing. The SSV architecture proposed by Kertesz et al. [85] which contracted the SLA between an individual service provider and consumer after the negotiation process. However, they did not consider the hierarchical relation of SLAs in the cloud environment. Although research by Undheim et al. [86] attempted to deploy the on-demand SLAs with QoS details on different levels, it followed the common SLA structure with stated shortcomings. NIST [87] has also pointed out the necessity of SLAs, SLA management, the definition of contracts, the orientation of monitoring on Service Level Objects (SLOs) and how to enforce them. However, a clear definition of a reference of a specific format is missing. This is also the case with the Internet Engineering Task Force (IETF) [88]. Besides these approaches of the companies and organisations, there are further efforts to develop and to realize cloud management architectures and systems. A basic discussion can be found in the book from Wieder et.al. [26] mainly concerned about SLA definitions and negotiations. CSLA [89] (Cloud Service Level Agreement) language was developed with the main focus on dealing with QoS uncertainty and performance fluctuations in cloud services. CSLA specification can be defined in different programming languages and are not XML restricted. The Foundation of Self Governing ICT Infrastructures (FOSIl)-Project [90] is another research project which aims at the usage of autonomic principals for information and communication systems. Self-determining infrastructures should be realized and made available through cloud-based services. Within the LoM2HiS autonomic SLA management is realized by translation of system parameters to abstract KPIs and SLOs [91]. The SLA specification is based on WSAL. The Texo project [92] attempts within the THESEUS research program to realize a conception of service descriptions, contract management, end to end marketplaces and monitoring from a business perspective. In addition, the development and use of WS-A based SLAs are needed. Looking at the cloud interfaces that describe the management of resources within the cloud, it becomes obvious that they are exclusively designed with the purpose of system monitoring. They do not provide direct monitoring of SLAs. Therefore the placement of machine-readable SLAs is extremely difficult. This is also the case with existing monitoring tools. SLA handling in clouds, resulting from the EU project OPTIMIS, is discussing negotiating and creating Service Level Agreements between infrastructure providers and service providers [93]. Although it is enhanced within the SLA@SOI project [58] its development is unclear because the SLA@SOI project develops its own format SLA(T) [94] and supported by the European IT industry. A comprehensive project result has been published on their web page. No independent analysis of the advantages or disadvantages of the SLA(T) format is available at the moment. It provides all structural requirements of an SLA and it has the greatest intersection with regard to content. SLA(T) is the basis of the proposed approach in this paper. Further, it should be accentuated that a meta-model SLA* [95] is defined which simplifies the extension and adaptability for SLA(T). SLA* is proposed as part of SLA@SOI European project to generalize the XML based web services standards WSLA, WS-Agreement and WSDL. The WS-Agreement (WS-A) was developed by the Open Grid Forum in the year 2007 (Version 1.0) The last update which was based on the work of the European SLA@SOI project was done in 2011. Its advantages are the expandability and the adaptability which is, on the other hand, also one of its greatest disadvantages because it has not been specified in details by Kearney [57]. It is based on technical transformation, the structural transformations, however, have not been taken into account. SLAC [96] is based on WS-Agreement and inherits many features and structures from it. The SLA-Ready [97] common reference model describes and promotes the uptake of cloud service level agreements, by providing a common understanding of SLAs for cloud services.

3.3 Performance Prediction

In order to give a reliable guarantee for a service, the only way is to be able to estimate the performance of the service in order to provide and adjust it to meet the given criteria. For this purpose, predictions are extremely useful because ideally, a problem does not have to arise in order to be able to react to it. Furthermore, resources can be better used and planned with good estimates. Neural Networks are widely used in forecasting problems. One of the earliest successful application of ANNs in forecasting is reported by Lapedes and Farber [98]. They used a feed-forward neural network with deterministic chaotic time series generated by the Glass-Mackey equation, to predict such dynamic non-linear systems. Artificial Neural Networks are proven universal approximators [99] [100] and are able to forecast both linear [101] and non-linear time series [102]. Adya and Collopy investigated the effectiveness of Neural Networks (NN) for forecasting and prediction [103]. They came to the conclusion that NN are well suited for the use of prediction, but need to be validated against a simple and well-accepted alternative method to show the direct value of this approach. Since forecasting problems are common to many different disciplines and diverse fields of research, it is very hard to be aware of all the work done in this area. Some examples are forecasting applications such as: temperature and weather [104] [105] [106], tourism [107], electricity load [108] [109], financial and economics [110] [111] [112] [113] and medical [114] [115] to name a few. Zhang, Patuwo, and Hu [102] show multiple other fields where prediction by ANN was successfully implemented. Similar research with different focus has been conducted in the past for the use of machine learning in cloud environments. Prevost et al. used a Neural Network (NN), as well as a Linear Predictor [116] to anticipate future workloads by learning from historical URL requests. Although both models were able to give efficient predictions, the Linear Predictor was able to predict more accurately. Li and Wang proposed their modified Neural Network algorithm nn-dwrr in [117]. The application of this algorithm led to a lowered average response time compared to the application of traditional capacity-based algorithms for scheduling incoming requests to VMs. In similar research, Hu et al. [118] have shown that their modification of a standard Support Vector Regression (SVR) algorithm can lead to accurate forecasting of CPU Load what can be used to achieve a better resources utilization. Another algorithm, which is renowned for providing good results in similar scenarios, is Linear Regression. Although the results are often weaker compared to Neural Networks or Support Vector Machines (SVM) in cases of workload prediction [119] [120], the fast training and deployment time of models built with Linear Regression should not be underestimated. Those examples show that there are a variety of optimization challenges in cloud environments which can be tackled by applying machine learning algorithms. What separates the current work from previous research is a detailed examination of specific characteristics of three different machine learning algorithms and presenting the results in a visual way. The choice to evaluate Neural Networks, Support Vector Machines and Linear Regression was made because those algorithms earned promising results in previously conducted research.

3.4 Scheduling Mechanisms

Scheduling refers to making a sequence of allocation decisions for a specific time frame. This enables good planning in advance and better utilization of the existing infrastructures. In the area of cloud computing, in particular, various methods play an important role for optimal allocation and distribution, as with the pay per use character cloud instance should be used as efficiently and cost-effectively as possible. Most commonly found scheduling approaches for the Cloud, like proposed by Mao and Humphrey [121] or Panda et al. [122], aim towards resource optimization based task scheduling. Therefore, cloud instances are regarded as computing tasks that have to be processed within certain resource constraints. However, this basic view is only suitable for cloud operations with already known tasks or performance requirements. In the real cloud environment, however, these are usually of no importance since cloud customers either do not know them or they are highly variable. The approach shown seems to have more applicability in grid computing. Many publications have been published in recent years regarding Grid Scheduling. This suggests a special interest in research in this area. In particular, the methods of nature-based algorithms, such as the evolutionary algorithms, are frequently used, cf. [123] [124]. Heuristics, however, play an equally important role in practice, cf. [125]. The main approach in grid scheduling is to split tasks (workloads) in so-called jobs, which are distributed on the basis of their computational requirements (MIPS), so that the total processing time is minimal. The publication of [126] compares four popular approaches to grid scheduling. For this purpose, the ant algorithm, a genetic algorithm, a particle-swarm algorithm and simulated annealing were implemented on the basis of the same preconditions. Anyhow, the approaches of grid scheduling are not or only very difficult to transfer to the problem area of cloud scheduling or the scheduling of virtual machines, since these instances are based on time rather than tasks. The big difference is that jobs are displayed in a one-dimensional unit, mostly in MIPS. However, instances in cloud scheduling are defined by a two or more-dimensional combination of resources and time. For this, the solution methods must be strongly adapted and the expected results will deviate from the findings described above. With the ever-increasing popularity of cloud computing, the number of research papers on the topic of cloud scheduling is increasing. For the problems of this thesis especially the work on the topics of virtual machine scheduling and cloud scheduling in the context of SLA awareness are interesting. Often grid scheduling approaches like [127] and [128], are transferred to the cloud. Here a genetic algorithm is used but also simple algorithms like First-In-First-Out (FIFO) are used. The basis here, as in the case of grid scheduling, is the division into jobs that have a specific computing time requirement, which are then distributed to existing virtual machines within the cloud. Another approach of [129] shows how loads can be distributed to an external cloud, such as the Amazon Elastic Compute Cloud, by means of a so-called Bursting Scheduler. Here, too, jobs are represented as computing tasks. Seoyoung et al. [130] deals with cloud scheduling for scientific applications. It should be noted that these applications have special requirements in terms of resources and computing time. Based on OpenNebula and Haizea [131], it could be proven that a shift of the scientific application into the cloud brings clear advantages. The same applies to [132] where the CloudSim Toolkit introduces a scheduling process for the processing of so-called process units (PE). The tasks are distributed to individual cloud instances to minimize the total processing time. The approach involves techniques such as backfilling, where smaller jobs may advance as long as they do not delay others' execution, and first-come-first-serve and round-robin algorithm, unfortunately, the approaches used are not transferable to the scheduling of cloud instances, but they show the power of the cloud computing approach itself. Haizea [133] itself is a resource manager or resource scheduler. Haizea can manage a set of computers (typically a cluster), allowing users to request exclusive use of those resources described in a variety of terms, such as "I need 10 nodes, each with 1 GB of memory, right now" or "I need 4 nodes, each with 2 CPUs and 2GB of memory, from 2pm to 4pm tomorrow". Haizea provides reservation and deadline sensitive type of leases along with the traditional immediate and best effort policies [134] [135] [136]. Haizea uses simple allocation policies for deadline sensitive leases. It tries to find out a single slot of required time between startTime and endTime of the given lease which can allocate the requested resources. If it is unable to ?nd such a time slot, it rejects the lease.

3.5 SLA Enforcement

SLA enforcement, especially in the area of cloud computing is an hot topic. It is important to note that different methods can be used to maximize the SLA compliance and at the same time serve as many SLAs or customers in parallel. For a cloud provider this is a very important point as its income and sales directly depend on it. Various techniques can be used to achieve this goal. Boniface et al. [137] discuss dynamic service provisioning using GRIA SLAs. Here the provisioning of services is based on pre agreed SLAs. The approach is based on Grid environments an they do not state how the low-level metrics are monitored and mapped to high-level SLAs. Comuzzi et al. [138] define the process for SLA establishment adopted within the EU project SLA@SOI framework. Their architecture for monitoring SLAs considers only two requirements, the availability of historical data for evaluating SLA offers and the assessment of the capability to monitor the terms in an SLA offer. The SLAs-LoM2HiS framework [139]uses a host monitor and run-time monitor. The first one is monitoring low-level resource metrics, whereas the latter is responsible for metric mapping and SLA violation monitoring. The framework tries to predict the SLA violations by predefining thresholds. Then alerts are notified into the internal knowledge component for possible adjustment in provider resource however there were not any other reaction strategies. It is part of the FOSII [90] infrastructure, where they mainly tried to map the low level metrics into high level SLA by their monitoring method. [140].In this framework, the SLAs consist of attributes, metrics and formulas have to be located in a central repository. The QoSMONaaS approach [141] focused to the QoS monitoring as a service. Although this flexible approach is a third party SLA monitoring system, it still needs the external service for SLA validating. Another related work is DeSVi architecture by Emeakaroha et al. [142], which is designed based on LoM2HiS [143] framework which limited for only one data centre as an IaaS. They applied the monitoring, analysis, planning and execution loop (MAPE) model to enforce the self-healing cloud computing. This architecture has beneficial for cloud management but not for sudden violation reaction. Overall the violation reaction process is rarely discussed in the mentioned SLA monitoring frameworks. The most of monitoring systems just tried to report the detected violations [84] [138] or assess the penalty cost of service providers [144] [145]. However, an effective reaction process is needed to react against SLA violations. Some studies such as [146], [147] and [148] utilized the SLAs to manage the provider resources without any reaction against violations. The SPECS project [149] aims towards the secure provisioning of cloud services based on SLA management. Therefore security SLAs are negotiated between the provider and customer. However, if the security policies contained therein are not adhered to, the service is simply considered unsafe and then terminated.

3.6 Autonomic Computing

Autonomic computing refers to the ability of a distributed system to manage its resources with little or no human intervention. It was launched in 2001 by IBM to reduce the complexity of managing large distributed systems [150]. Autonomic computing aims to facilitate self-management of complex systems consisting of various components. Especially in a highly dynamic and rapidly frequented area such as cloud computing, there is a need for autonomous systems that require no or only minimal user intervention. Autonomic computing systems involve service-oriented technologies, agent technologies, adaptive control theory, machine learning, optimization theory, and many more [151] [152]. In cloud computing it helps to address the challenges of cloud management in an fast paced environment. Autonomic resource allocation [153], [154] has been examined to meet the on demand thought of cloud computing. There are now many commercial approaches that can provide resources on the fly in a cloud system. Other areas of research include autonomic self-testing [155], as well as autonomic life-cycle management [156] in the cloud.

3.7 Conclusion

In this chapter, a detailed review of the state of the art for different areas required for the implementation of Cloud SLA autonomic management and performance monitoring is presented. The SLA specification can be considered as a basis for automating the whole Cloud SLA life cycle. The SLA creation process requires a common definition of the SLA parameters among the participating parties. Performance prediction can be seen as a perquisite to successfully manage SLA in the Cloud. Likewise, the SLA monitoring becomes of high importance due to performance and QoS requirements. Finally, scheduling mechanisms and SLA enforcement are required to react in case of unwanted outcomes.

CHAPTER 4

AUTONOMIC CLOUD COMPUTING SLA MANAGEMENT APPROACH

4.1 The Autonomic Computing Approach

The ever increasing complexity, multi dimensionality and abstraction of modern computer systems make it difficult for users to utilize, maintain and fully understand them. With this issue in mind, IBM [150] developed a concept for autonomous systems and coined the phrase Autonomic Computing. According to the IBMs vision autonomic behaviour can be characterised by key capabilities such as self-diagnosis, self-configuration, self-adaption and self-healing, which are all focusing on the continuous executability of a computer system without the need of user interaction. There are several definitions for autonomic computing, following the most common ones are stated:

- 'The vision of autonomic computing is to create software through self-* properties' [157].
- 'Autonomic computing is the ability to manage your computing enterprise through hardware and software that automatically and dynamically responds to the requirements of your business' [158].
- 'Autonomic computing is the ability to manage your computing enterprise through hardware and software that automatically and dynamically responds to the requirements of your business' [158].

The phrase autonomic is derived from human biology, where the autonomic nervous system keeps track of the bodies vital functions such as heartbeat, temperature and breathing while working in the background. However, there are main differences between the human body and autonomic systems. Firstly most of the tasks of the autonomic nervous system can't be controlled, where as the tasks of an autonomic systems are based on human written policies. To make these policies work the system has the need for sensing the state of the system and its environment. According to the IBM vision, a autonomic system consists many interacting autonomic elements. These elements control the autonomic computing environment, consisting of other autonomic elements or the managed system, based on defined policies. This was modelled into the autonomic control loop (MAPE-K paradigm).

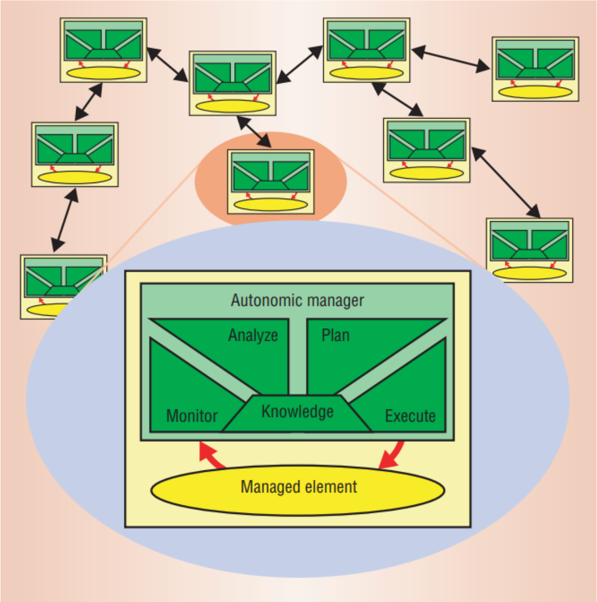


Figure 4.1: Autonomic Control Loop (MAPE-K) [6]

The four phases, monitoring, analysis, planning and execution are extended by a common knowledge base and form the so called autonomic manager, which directly interacts with the managed element. Such managed elements can consists of, for example CPU, network nodes, virtual machine instances, databases or applications.

- **Monitoring** is collecting informations and metadata about the status, process and trend of the managed element. Here various methods of monitoring can be used, such as push or pull systems or agent-based systems to periodically retrieve the needed information.

- **Analysis** uses these collected informations to decide whether or not the indication of an event to start an adaptation action has been reached. If such an event occurs an action strategy should be created to adapt and adjust the system to the targeted state.

- **Planning** uses this action strategy to create an adaption plan based on the targeted state and the current state of the system. This plan consists of instructions, which in detail describe the actual changes and actions.

- **Execution** runs these adaptations and adjustments. Therefore the execution of the system is altered and the adjustments are made. Afterwards the loop starts again.

The Dynaco adaptation model proposes a similar approach to the MAPE for distributed application by leveraging component-based design [159] [160]. Autonomic computing aims to facilitate self-management of complex systems consisting of various components. Autonomic computing systems involve service-oriented technologies, agent technologies, adaptive control theory, machine learning, optimization theory, and many more [151] [152].

The aim of introducing this paradigm in this thesis, is to achieve autonomic behaviour in our proposed Cloud management infrastructure, whereby appropriate actions are taken towards the enforcement of SLAs and performance metrics.

This report has identified the requirement for cloud computing infrastructures to guarantee service qualities and the need for new mechanisms for the integration and management of service level agreements.

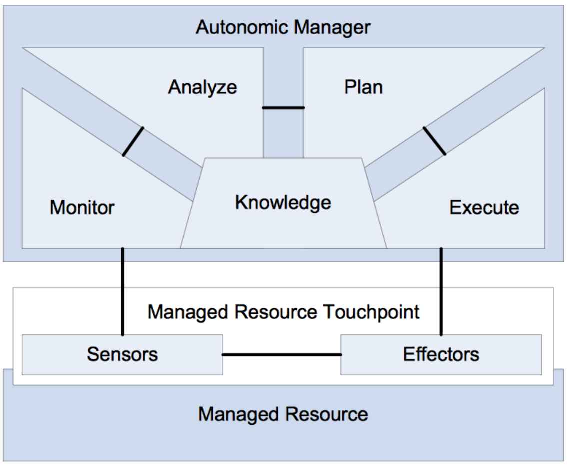


Figure 4.2: Autonomic Computing Control Loop from [7]

However, to address the identified problems within the current cloud computing landscape a system is needed which supports dynamic and fine-grained quality assurance while being flexible enough to cope with the frequently changing infrastructure. And automation of the underlying SLA management process, which includes creation, monitoring, reporting, control and adaption. Especially for this more research in the area of behaviour analysis, prediction and anomaly detection is necessary.

4.2 Autonomic SLA Management Architecture

To empower cloud users with the possibility of managed cloud service quality and simultaneously increase the trust, reliability and appear of cloud computing for business use, this research proposes the concept of autonomic continuous dynamic SLA management for clouds, which will be realized by the Autonomic SLA Management as a Service (ASLAMaaS) architecture. It is built upon the IBM developed autonomic manager concept MAPE-K [7] (Monitor-Analysis-Plan-Execute-Knowledge).

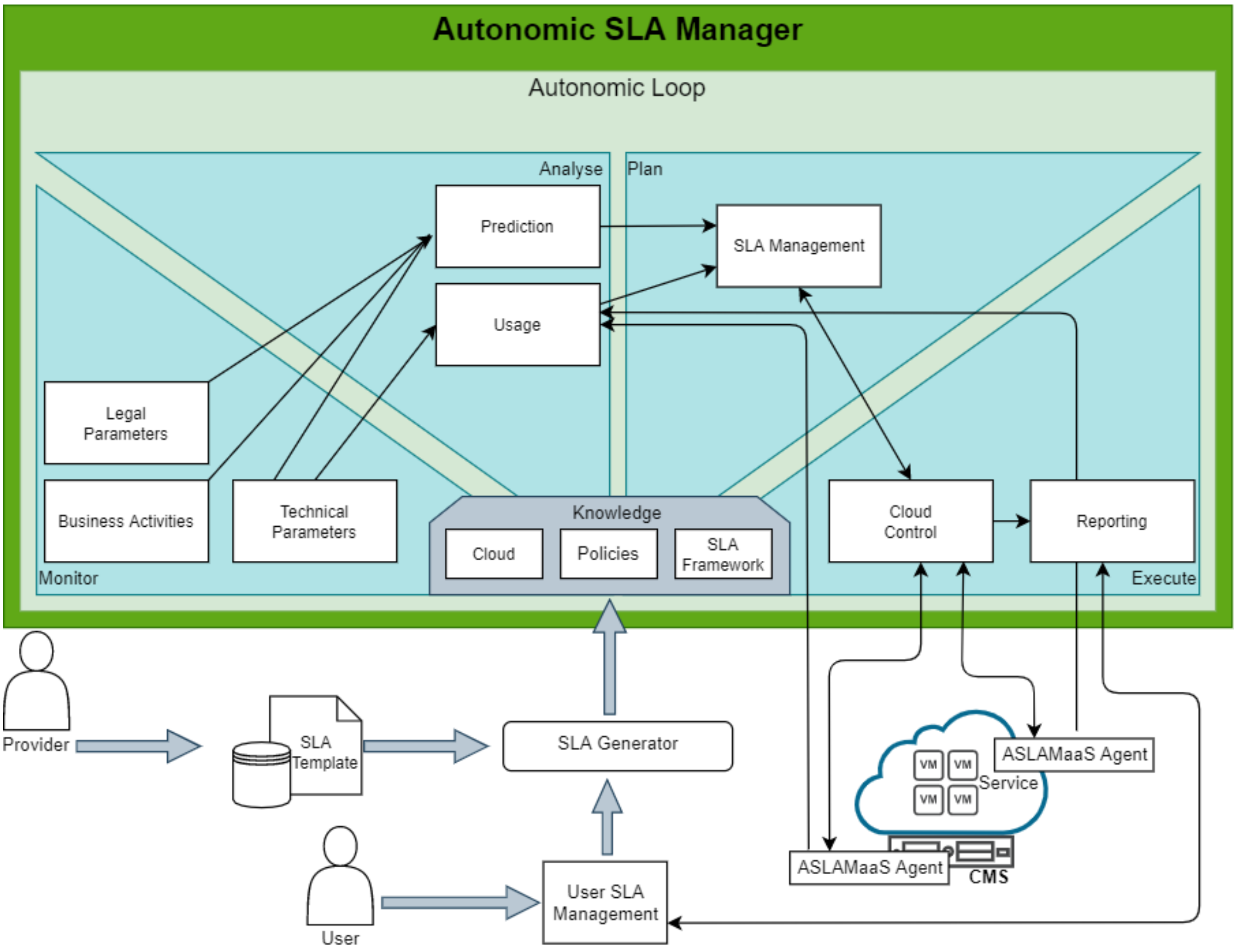


Figure 4.3: Early Iteration of the Autonomic SLA Management Approach

During the first phase of the research, cloud-specific issues with regard to the quality of cloud services and the need for dynamic management of service qualities were identified and analysed. The requirements for the architecture were derived from this. An early version of this architecture can be seen in Figure 4.3. The aim of the ASLAMaaS architecture presented was to integrate dynamic, autonomous SLA management into existing cloud infrastructures so that cloud users can independently conclude SLAs for cloud services on a self-service basis. The general service quality and reliability should also be improved in order to strengthen the trust and economic use of cloud computing in general. For this purpose, the architecture should enable continuous and flexible monitoring of the cloud resources with regard to the corresponding service levels. At the same time, advanced reports and logs are built into each module of the architecture to enhance transparency between the service provider and consumer by enabling comprehensibility. Due to the strong dynamic and rapidly changing character of cloud computing, it is necessary that the system continuously adapts. Autonomic computing systems are capable of continuous self-monitoring and adjustment. This early approach therefore was created by combining the information and entities identified for SLA management in the cloud with the autonomous principle. After reviewing existing frameworks, methods and SLA management research, the relevant components were arranged according to the APE-K concept in order to follow the autonomous concept. The overview of the components is shown in Figure 4.2.

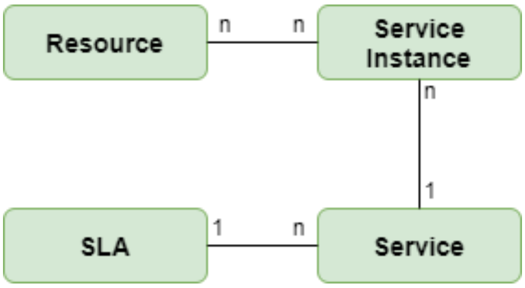


Figure 4.4: Architectural Entity Relationships

With this approach, shown in Figure 4.3, legally relevant and technical parameters as well as planned or executed business operations are monitored and the data is used in SLA management utilizing usage behaviour analysis and forecasting load in order to comply with SLA by controlling the CMS and the services running on it. The connection to the cloud system was established with the help of ASLAMaaS agents, which take over control and monitoring on the CMS or on the service level. For the monitoring, a sensor is connected to the CMS and the managed resource touchpoint, there the monitoring unit collects the data from the sensors and hands it over to the analysis unit. Here usage prediction and analysis is done, and with the help of an external knowledge base (cloud environmental data and policies) creates an action inside the SLA Management component. These actions are carried out by the Cloud Control unit (Execute), which adjusts the managed resource through effectors in the ASLAMaas agents. In the case of cloud infrastructure, this could be upscaling the number of CPUs of a VM instance. All operations are documented and stakeholders are informed by means of reporting. It should be noted here that the creation of the SLAs was viewed as an external process in which the provider creates SLA templates and the user can select the respective service level. During this research, a collaboration with a cloud provider and a medium-sized company showed that this approach is not sufficient. This becomes clear when the most important entities within the framework, SLA, service and resource as well as their relationships are shown, which are shown in Figure 4.4.

By reviewing the requirements and reiterating the approach, the need for an easy and self-serviceable SLA creation and thus a Frontend which is directed towards the user was born.

Therefore the ASLAMaaS framework was targeted towards the following cloud challenges:

- The architecture should meet the dynamic character of cloud computing.
- The architecture shall enable management of multiple SLAs and QoS parameters in parallel, in order to process the multi-tenancy of clouds.
- The architecture shall enable easy and self-serviceable creation of SLAs.
- The architecture must provide continuous and flexible monitoring of cloud resources in regard to the corresponding service levels.
- The architecture shall aim to minimize SLA violations.
- The architecture should require a minimal amount of user interaction.
- The architecture shall include business factors in the management of SLAs.
- The architecture must be reliable and scalable.

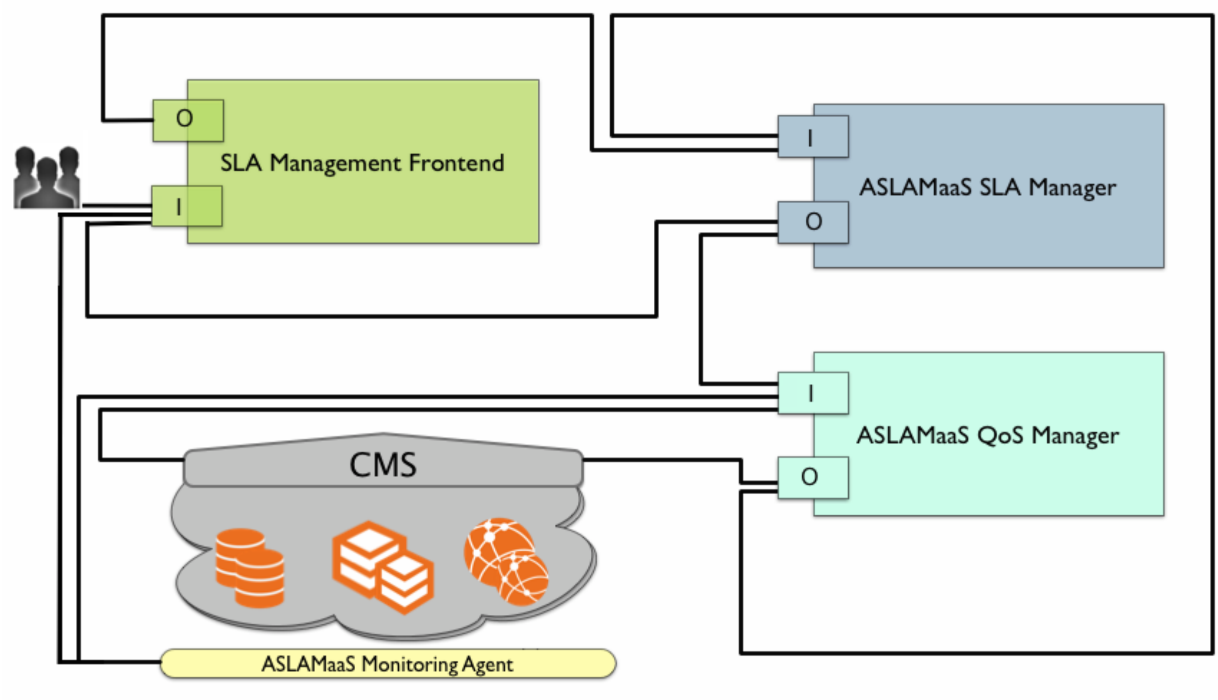


Figure 4.5: Information Flow between ASLAMaaS Modules

The final presented architecture consists of three main modules, the SLA Management Frontend, SLA Manager and the QoS Manager. The information flow between the individual modules can be seen in Figure 4.5. The SLA Management Frontend thereby marks the input layer of the

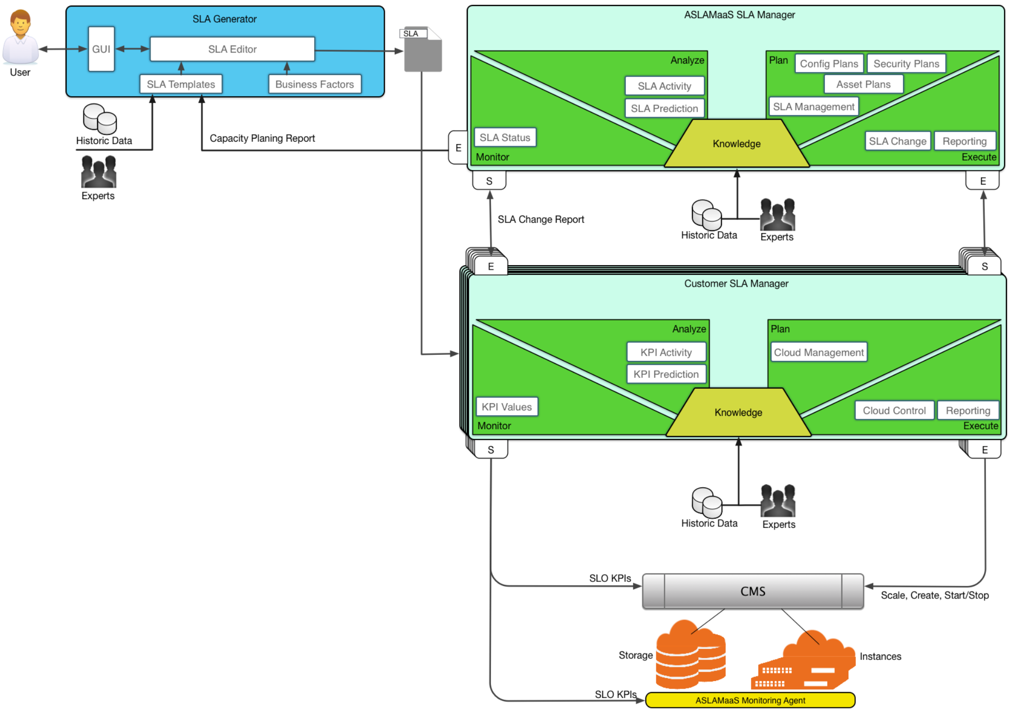


Figure 4.6: ASLAMaaS Architecture Overview

system, where users can create, monitor and adjust SLAs for their services.

To facilitate the process of creating new SLAs there will be a repository of pre-made SLA templates available within a graphical SLA editor. Thus users can easier and faster create their own specific SLAs by selecting a fitting template, which shall reflect best-practice and experience data for generalized types of services such as web services, databases systems, ERP systems and so on. The used SLA templates will use different key performance indicators (KPIs) grouped into categories based on their usage type. For each KPIs, a specific QoS parameter and the associated metric has to be stored within the repository. A more detailed outlook on the SLA creation process and its components can be found in subsequent Chapter 4.3.

After the creation and activation of an SLA for a cloud services the SLA Manager stores, it in a repository and starts the autonomous control loop. An overview of the ASLAMaaS SLA Manager and its components is shown in Figure 4.7. The SLA Manager is used as an abstraction level, in which the individual SLAs and their dependencies between each other are processed. Within the SLA Manager, the general functionality regarding SLA compliance is processed.

For this measuring and keeping track of the status of each SLA and the containing service, levels is needed. Therefore, the cloud management system, virtual resources through intelligent agents and archived data will be used as input sources. Within the Analyse part, the data is used

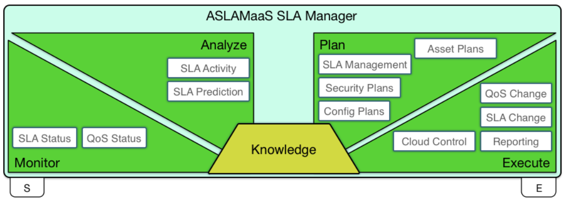


Figure 4.7: ASLAMaaS SLA Manager MAPE-K Loop

to perceive the current state and predict the future behaviour of the managed SLAs. Here the mutual influencing SLAs factors and the strategic business planning of the provider shall be weighed against each other. This could mean, for example, that in a state of emergency some contracts could be abandoned in favour of others so that the resulting financial damage will be reduced. Likewise here the beginning, suspension and stop of the dynamic SLAs and the total infrastructure overview data shall be used for strategic planning and pricing.

Table 4.1: ASLAMaaS SLA Manager Components

Monitor	
SLA Status	Monitoring of the overall SLA targets for compliance.
QoS Status	Monitoring of the individual QoS targets for compliance.
Analyze	
SLA Activity	Analysis of the behaviour, contract changes, history and volume of all SLAs.
SLA Prediction	Prediction of the to be expected QoS values and possible SLA violations.
Plan	
SLA Management	Contains all changes regarding execution of a SLA e.g. cancellation of SLA.
Security Plans	Contains all changes related to security topics e.g increase antivirus update interval.
Config Plans	Contains all changes related to configuration e.g change network packet size.
Asset Plans	Contains all changes related to assets e.g add additional memory to a system.
Execute	
Cloud Control	Implements the changes on the cloud platform or infrastructure.
QoS Change	Implements the changes on specific QoS related components.
SLA Change	Implements the changes on the SLAs.
Reporting	Documents and informs the stakeholders about the adjustments, status and changes.

Depending on the specific service, cloud SLAs may consist of several independent or conditional KPIs In order to ensure the service quality and operate within the agreed on limitations every KPI has to be controlled and monitored individually. For this every managed service level respectively KPI within an SLA will get its independent autonomous management loop. The corresponding graphical overview is shown in Figure 4.9 below.

The basic functionality will be nearly the same as the administration by the SLA Manager, only that instead of SLAs the individual QoS characteristics are directly influenced by the ad-

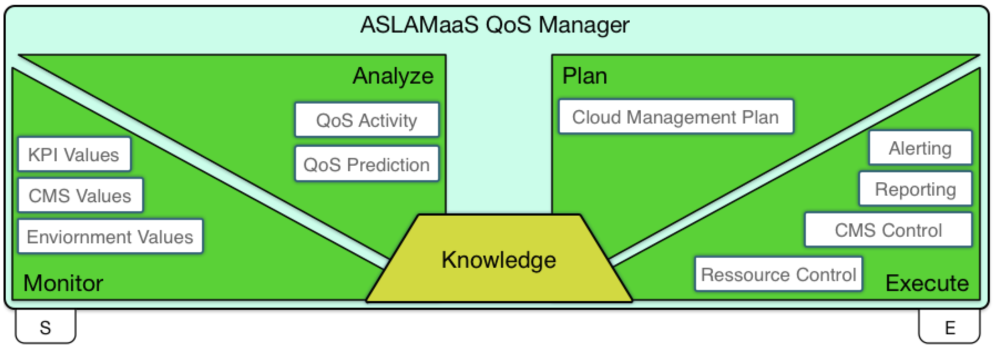


Figure 4.8: ASLAMaaS QoS Manager MAPE-K Loop

justment of the cloud infrastructure and resources. So every SLA Manager instance will have multiple instances of the QoS Manager which will take control of the corresponding parameters, monitoring and reporting. Starting by the Monitor, KPI relevant technical parameters, environmental data and dependencies are collected. This will be done by using the stored metrics for each KPI, deposited within the Knowledge base. Such metrics will be measured either directly within the Cloud Management System (CMS) or directly on the resources by using intelligent monitoring agents. Within the Analyse part prediction algorithms are used to estimate future problems and estimate long term developments. Based on these calculations, respectively the current state of the monitored parameters, action Plans will be conceived. Such plans will consist of actions like scaling up or down, allocation or deallocation of resources, altering the infrastructure, or alerting the corresponding service provider if an adaption is not automatically possible.

Table 4.2: ASLAMaaS QoS Manager Components

Monitor	
KPI Values	Monitoring of the individual KPI regarding the QoS parameter.
CMS Values	Monitoring of the overall parameters of the cloud management system.
Environment Values	Monitoring additional external and internal parameters e.g. planned events.
Analyze	
QoS Activity	Analysis of the behaviour, changes, history and volume of the parameter.
QoS Prediction	Prediction of the to be expected QoS values.
Plan	
Cloud Management Plan	Contains all changes regarding influencing components.
Execute	
Alerting	Alerts based on current QoS level or prediction.
Resource Control	Implements the changes on the resources e.g. change VM type to medium.
CMS Control	Implements the changes in the CMS e.g. add 4 new VMs of type small.
Reporting	Documents and informs the stakeholders about changes and status.

The proposed infrastructure shall intervene directly with the CMS. As reporting is an important core component of SLA management, for example as evidence in case of SLA violations or as a statement of the delivered service level, the continuous reporting and logging must be ensured. The system has to exhibit to the users that the in an SLA agreed service levels have been consistently met or remained within the defined deviations. Therefore the reporting is integrated into the Execute part of the QoS Manager as well as in the SLA Manager of the ASLAMaaS architecture.

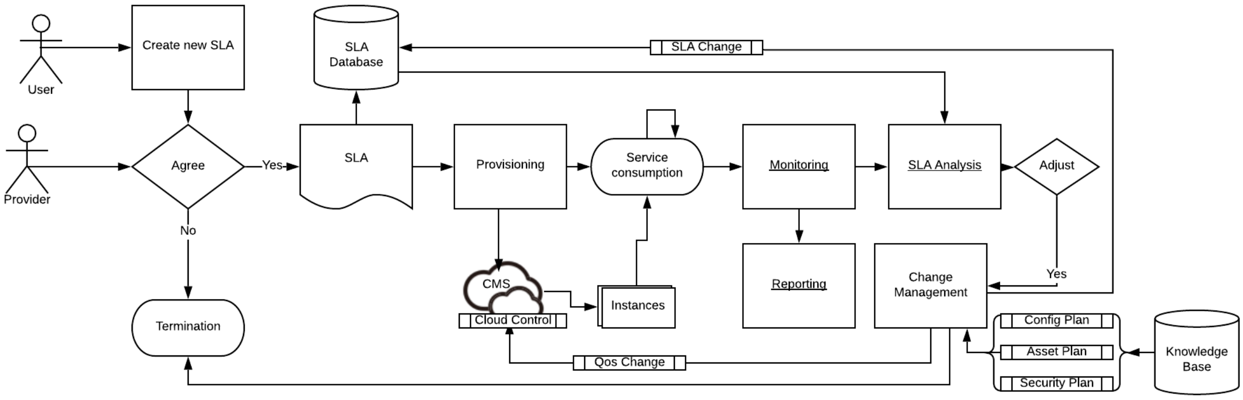


Figure 4.9: Flowchart of the ASLAMaaS SLA Manager

In figure 4.9 it can be seen how the informational flowchart of the SLA Manager module works. The entire process is started by a user creating a cloud SLA. A detailed description of the components, process and GUI interfaces used for this is presented in the next section. Yet it is sufficient to know that the process begins with the mutual consent of both parties, the user and the provider. The is no use made of negotiations with several iterations of contracts, such as are otherwise usual in the area of creating SLAs, because this would contradict the fast-moving nature of the cloud and the whole self-service idea of this approach. After the SLA has been accepted, the contract and the conditions agreed therein are saved in a database. Then the negotiated resources are provisioned via the CMS and made available to the user for use. At the same time, the usage, reporting and monitoring cycle begins. The data from the monitoring are analysed and compared with the stored SLA terms. This results in the need to adapt to the system or not. Based on this, the configuration of the CMS is adjusted with the help of stored configuration, asset, security and control plans and policies in order to meet the SLA. If this is not possible or an adjustment does not deliver the desired results, it is possible to terminate the SLA.

4.2.1 Fault tolerance and resilience

Although the focus of the presented architecture was on the dynamic and autonomous SLA integration into the cloud, the point of resilience and fault tolerance must not be forgotten. Reliability and thus the ability to deal with errors is one of the most important foundations of stable, as well as autonomous systems [161]. The two main components of the architecture are the SLA manager and the QoS manager with their modules. Both are essentially the core of the architecture presented here. The CMS thereby is seen as an external system which is controlled either via an interface or a module and thus will not be discussed further in this consideration. Starting with the SLA Management Frontend, the consideration regarding fault tolerance and resilience was to apply the principles of replication and redundancy. It should be noted with this component that a possible failure would only have a minor impact on the functioning of the system. In this case, it would no longer be possible to conclude new SLAs, to change existing ones or to submit current reports. Which is uncomfortable but not absolutely necessary for operation. However, it must be ensured that no incorrect or incomplete changes or data records are transferred to the system. This could be done by storing SLAs on two identical "mirror" servers and using protocols that ensure transactional safety in order to enhance data resilience. In contrast, the two ASLAMaaS managers are continuously required for operation. Failure of these components would have serious consequences. Therefore, in the architecture, it was considered to run several instances of the ASLAMaaS QoS Manager and SLA Manager at all times. Here, too, redundancy and replication principles ensure that the system is operated constantly. In addition, the individual modules are designed so that they operate statelessly and the number of available instances is continuously adapted. In the event of a temporary failure of a data source such as the SLA or KPI databases, the individual instances use data stores and dynamic caching so that they can continue to operate. The health status of each individual component is checked regularly and replaced if necessary. An even load distribution is accomplished with the help of load balancers.

4.3 SLA Creation Process

To keep up with the dynamic ephemeral character of cloud computing SLA management needs to be adaptable any time. For this classical process of SLA negotiation must be adapted towards the more dynamic self-service. In the proposed architecture this task should be solved by SLA Management Frontend. A structural overview is shown in Figure 4.10.

As described in Chapter 3, SLAs comprise the expected quality of service the provider has to deliver, which are recorded as the agreed service levels. The service levels include one or multiple KPIs, which describe in each case a specific QoS parameter and the associated metric to monitor. In order to allow users to choose the KPIs for their SLA contract, self-reliant certain preconditions must be met. Since cloud architectures are constantly changing the available KPIs

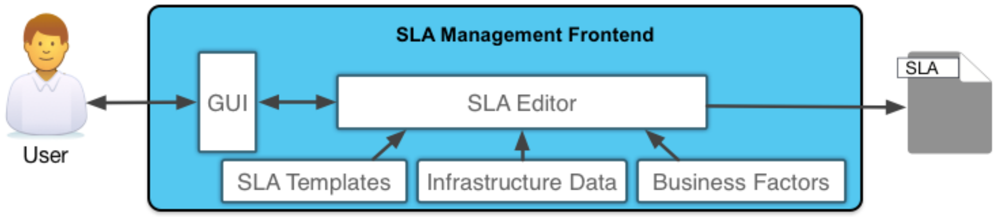


Figure 4.10: ASLAMaaS SLA Management Frontend Overview

is provider dependent and must be pre-accepted beforehand. Therefore possible cloud relevant KPIs have been identified in Chapter 2.3.2 and must be selected initially by the provider.

Based on the provider accepted KPIs meaningful margins have to be calculated as a basis for the SLA creation process. This is necessary because in this self-service approach the provider shall not have to engage in the process respectively negotiate with the consumer. The margins mark the boundaries in which a user can choose the service level. In order to provide only feasible conditions of which the user can choose pre-calculations have to be made and pre-conditions have to be checked. For this purpose, various mathematical functions shall be examined for their suitability and expressiveness in order to integrate a suitable model.

Additionally, the utilization, expected usage and performance data of the infrastructure shall be matched together with business concerns and the strategic focus of the provider to create a dynamic pricing model for the offered services and the corresponding SLAs. If, for example, an infrastructure constantly delivers a response-time of below 170ms and only a few fluctuations of users are expected the reasoning model may set the lower margin for this KPI to 200ms or slightly above. Since it is common practice to make prices not directly on each possible service level, but to create pricing categories, such as low, medium, high or silver, gold platinum the delivered price model shall then create these pricing categories. This should enable simpler management of the managed service levels and makes the gradation of customers easier for the system.

In order to autonomously conduct this process the SLA templates and the finished filled out SLAs used must be represented in a machine-processable form. For this, a machine-readable agreement description language shall be used. The Adaptable Service Level Objective Agreement (A-SLO-A) model [162], which is based on the SLA* model, enables the use of dynamic and constant agreement alteration and therefore it will be used as the basis of the description of SLAs in ASLAMaaS.

4.4 Prediction, Detection and Management of Service Levels

In order to guarantee the QoS of a provided service, it is necessary for particular things to be monitored. Firstly, the performance of the infrastructure must be monitored since dependencies arise in a multi-tenancy environment with shared resources. Secondly, the use of cloud services usually varies greatly and thus the need for resources fluctuates likewise. To cope with this changing needs and to achieve SLA compliance the provided resources must be adjusted. If one could predict the usage of a service, looking ahead further than the infrastructure provisioning delay time, one could guarantee the QoS for that specific service. Therefore prediction and detection techniques should be

To detect unforeseen events, cloud components behaviour is constantly monitored and analysed. Therefore events from each of the CMS and monitoring agents are processed to enable anomalous behaviour detection. To determine, which anomaly detection algorithms are suited best, an evaluation has to be done. The algorithms considered include machine learning approaches like neural networks or Markov models, statistical methods, outlier detection and some specialized algorithms [163].

The Plan part of the MAPE-K concept in this module creates the action plans for the infrastructure, the CMS, the SLA Manager itself and additionally delivers adjustments directly back into the SLA Editor, where for example the margins could be readjusted to cope with the changes. Thus this the SLA templates and the corresponding KPI margins are always compliant with the actual performance of the cloud system. The action plans result in adaptation of the cloud infrastructure, like for example if the stated response-time inside a certain SLA tends to be broken the SLA manager can instruct the virtual network agent to re-route the traffic if the problem is network dependent. In case of this problem related to overloaded CPU or virtual instances, the SLA Manager could start new instances or allocate more CPU cores.

The various methods to manage the QoS parameters have to be model individually and tested about their suitability. A sample for the regulation of the KPI response-time can be found at Frey et al. [164]. Their scalable cloud services have been started and stopped based on a fuzzy control set. This and other similar control mechanisms enable the Execute part to adapt the infrastructure and the services so that SLA violations can be avoided and the QoS is guaranteed. Within both MAPE-K loops, the Knowledge consists mainly of the historic data about the services, the SLAs and the cloud environment, which was measured continuously and expert knowledge in the form of best practices and empirical values as well as strategic business plans.

4.5 Autonomic SLA Management Evaluation

Since the developed Autonomic Cloud Computing SLA Management approach and architecture is a major novel contribution of this research by providing a cloud computing SLA management system, it gets evaluated in the following. SLA management and enforcement in cloud environments is a non-trivial task due to a cloud's characteristics. Using the autonomic computing principle and the MAPE-K paradigm the presented architecture achieves to meet the dynamic and self-service character of the cloud. Therefore the presented approach enables the on-the-fly generation of

SLAs within predefined or infrastructure-derived template frames. This minimizes the need for human intervention and operation. At the beginning of the research, no or only rudimentary SLAs for Cloud Computing had been established. These were generated either generally as a framework contract, like for example all customers have the availability of 98%, or with a great individual effort in a special SLA creation process. This has not yet changed. The author is not aware of any industrially used procedure that allows Cloud SLAs to be created dynamically using the cloud environment data. The second major point the presented architecture and approach aims at is the autonomous management, enforcement, optimization and compliance of already created Cloud SLAs. The presented approach here aims to improve the control and operation of a cloud system and the adaptation of such a system to its varying usage by prediction. Such approaches are not new. However, this was not yet applied in the area of cloud computing at the time of writing. Particularly in cloud computing with its dynamically usable resources and the associated possibilities for control, there are particular synergies here. However, there are many approaches in the field of grid computing which are based on the same principle. Finally, the implementation of the machine-usable SLA language A-SLO-A, which is described in detail in chapter 5, represents an important step towards dynamic Cloud SLAs. Because only with such a language, legally binding and on thy fly negotiable cloud SLAs can be generated. At the time of the research, there were several initiatives and European funded research projects that wanted to create such a standard. So far, however, this has not yet been achieved.

4.6 Conclusion

The proposed SLA management framework for cloud computing is a novel and robust specification, for supporting QoS and SLAs management in cloud infrastructures. It is designed to address the growing demand for QoS and performance guarantees arising from the increasing use of cloud computing by small and medium-sized industries. Overall it aims to increase the reliability and transparency of cloud infrastructures, giving users the opportunity to manage their quality needs. At worst it is providing the same level of quality management currently found in cloud computing environments. At best, all cloud customers can benefit from it, by defining and changing on-demand the service levels for their cloud instances. SLA reports enable users to get a comprehensible knowledge about the status of their cloud instances at any time and enable continuous verifiable proof about the delivered services. Cloud providers will be able to estimate the potential performance of their infrastructures and based on that give customers service guarantees. Performance evaluation, prediction and behaviour analysis is used to detect possible risks and intervene avert SLA violations. The ASLAMaaS framework is based on concurrent autonomous management modules, which evaluate the performance levels of cloud resources such as, cloud instances, storage, network and the cloud infrastructure as well as specific cloud services. This will improve the overall reliability and performance of cloud infrastructures.

CHAPTER 5

IMPLEMENTATION AND EVALUATION

This chapter discusses the individual components and the resulting test evaluation.

5.1 SLA Frontend

As it could already be seen in the previous chapters, the process of concluding SLAs for CLoud services is only rudimentary, if at all, established in the current state. Therefore, this chapter deals with the prototype implementation and its evaluation of a system that should make this possible. For this purpose, system components were built, tested and evaluated on the basis of the technologies and architectures described before. Starting with an SLA front end, the possible implementation of such a system is shown. The aim of the presented implementation is, on the one hand, to show a simple yet powerful way for cloud users to generate, manage and observer Cloud SLAs (Frontend), and on the other hand, to build a system that manages, checks and reports these SLAs and the KPIs they contain and uses various methods to adapt the cloud system to minimize SLA violations (Backend).

5.1.1 GUI

In order to interact with the implementation a GUI was designed to facilitate the use and the feel of the prototypical system. During the duration of the research work, the presented GUI was developed in cooperation with a small and a medium-sized company that specializes in the operation and provision of cloud services. Particular care was taken to reflect the long time experience of both partners in negotiating SLAs with customers. In figure 5.1 there can bee see the first version of the planned GUI as a mock-up. With the help of the subdivision of the SLA groups into tabs and the setting of the individual KPIs with the help of sliders, a simple but efficient option could be created to configure SLAs for the cloud system.

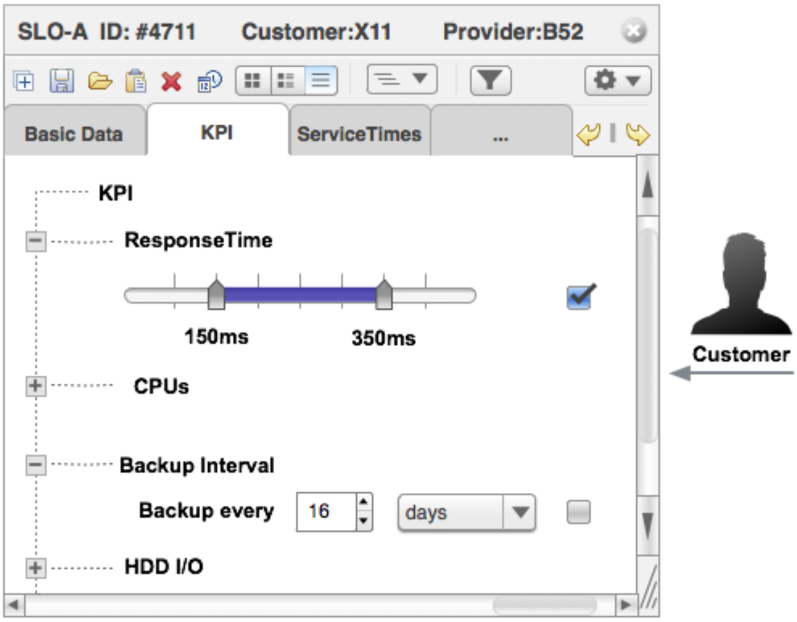
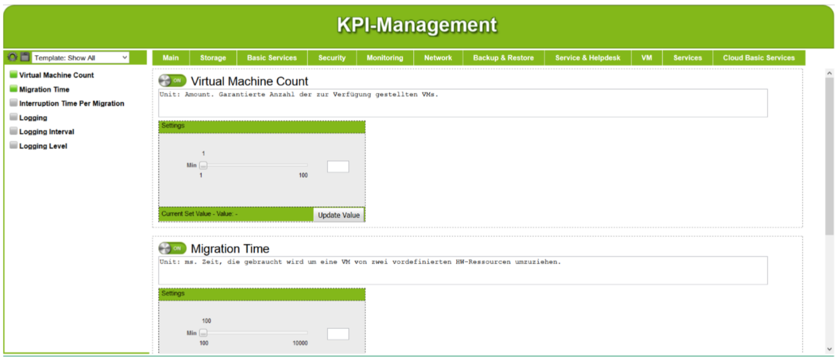


Figure 5.1: First GUI Mock-up

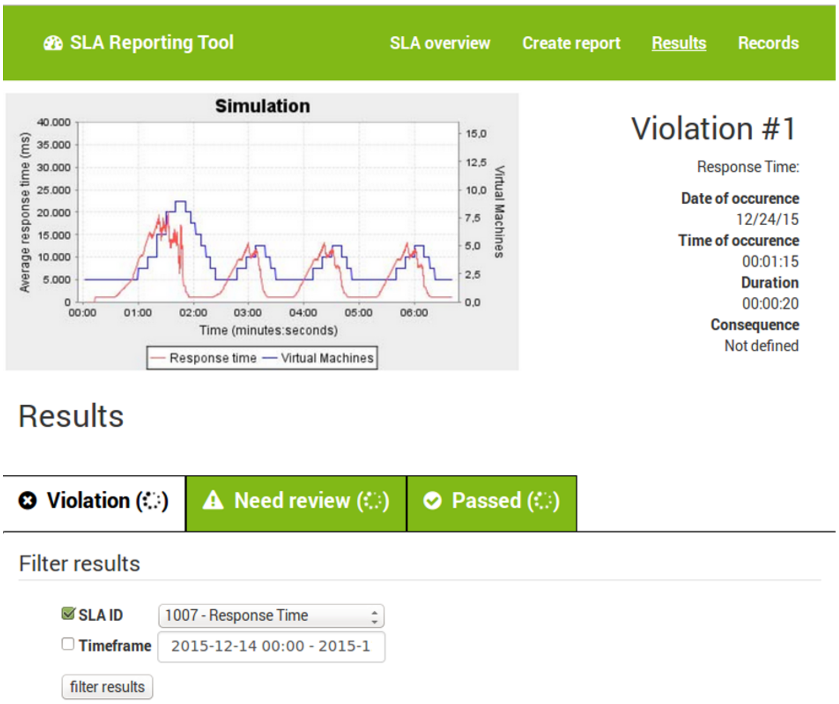
Figure 5.2 shows the finished GUI in our test system, which has been adapted in its colour scheme. However, the division into tabs and sliders has been retained. In addition, the SLA function for switching components middle switch on and off has been added. This enabled a greater modularization of the activated KPIs inside the SLAs.



Unit: Amount. Garantierte Anzahl der zur Verfügung gestellten VMs.

Unit: ms. Zeit, die gebraucht wird um eine VM von zwei vordefinierten HW-Ressourcen umzuziehen.

Figure 5.2: Final GUI



Violation #1

Response Time:

Date of occurence
12/24/15

Time of occurence
00:01:15

Duration
00:00:20

Consequence
Not defined

Results

Filter results

☑ **SLA ID** 1007 - Response Time

☐ **Timeframe** 2015-12-14 00:00 - 2015-1

Figure 5.3: SLA Violation Report

5.1.2 KPIs and Infrastructure Data

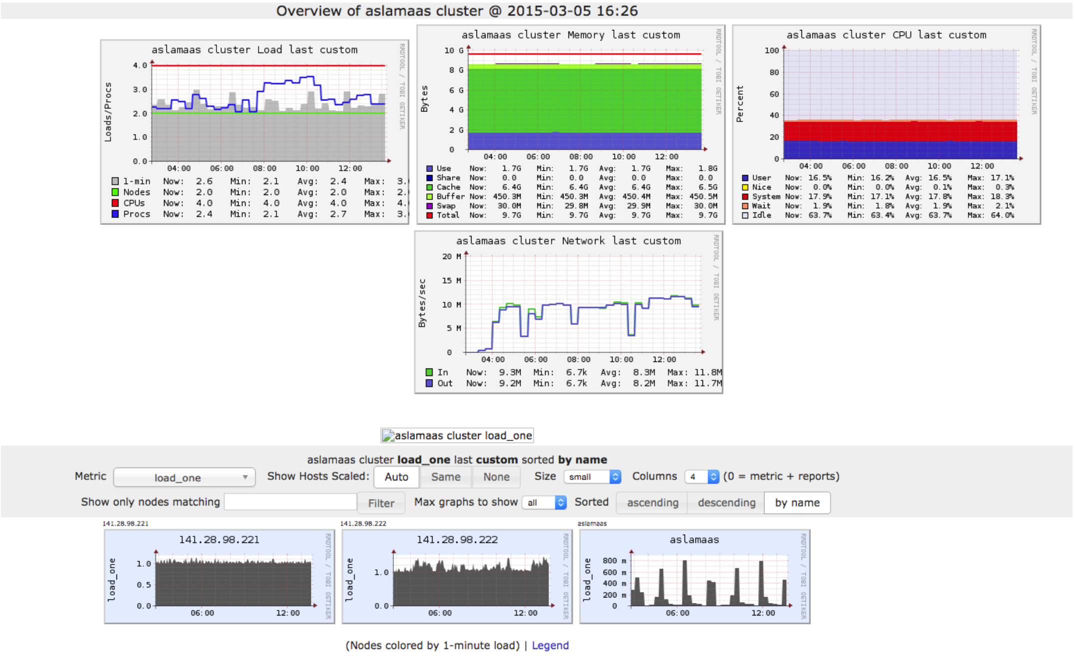
As introduced in chapter 2.3.2, KPIs are widely used to measure and overview computing environments. In this implementation and evaluation environment data has been collected via the Ganglia [165] distributed monitoring system, which is based on a hierarchical design and targeted at federations of clusters. Figure 5.4 below shows the Ganglia interface with collected infrastructure data from our OpenStack Cloud environment.

Here it can be seen how core KPIs, such as memory usage, cpu load, process load and network traffic is monitored. The test environment consisted of two worker nodes. Here different types of load scenarios were used by running a request generator, which accessed the services. This data was then collected and archived during a 6 months period.

Figure 5.5 above shows the aggregated load graph for the cluster during the testing period. All data was as well collected in a time-series data store. Additional in the scenarios described in later sections, simulated workloads have been used to generate a repeatable and comparable environment. Various typical workload scenarios were depicted, such as a burst load scenario, which could also be observed on the live system due to real-life usage

5.1.3 A-SLO-A

The reference implementation of the SLO-A format is based on the abstract SLA* model, developed within the SLA@SOI project [57]. The SLA* model is an abstraction layer and follows


Overview of aslamaas cluster @ 2015-03-05 16:26
aslamaas cluster Load last custom
Loads/Procs
4.0
3.0
2.0
1.0
0.0
04:00 06:00 08:00 10:00 12:00
1-min Now: 2.6 Min: 2.1 Avg: 2.4 Max: 3.
Nodes Now: 2.0 Min: 2.0 Avg: 2.0 Max: 2.
CPUs Now: 4.0 Min: 4.0 Avg: 4.0 Max: 4.
Procs Now: 2.4 Min: 2.1 Avg: 2.7 Max: 3.
aslamaas cluster Memory last custom
Bytes
10 G
8 G
6 G
4 G
2 G
0
04:00 06:00 08:00 10:00 12:00
Use Now: 1.7G Min: 1.7G Avg: 1.7G Max: 1.8G
Share Now: 0.0 Min: 0.0 Avg: 0.0 Max: 0.0
Cache Now: 6.4G Min: 6.4G Avg: 6.4G Max: 6.5G
Buffer Now: 450.3M Min: 450.3M Avg: 450.4M Max: 450.5M
Swap Now: 30.0M Min: 29.8M Avg: 29.9M Max: 30.0M
Total Now: 9.7G Min: 9.7G Avg: 9.7G Max: 9.7G
aslamaas cluster CPU last custom
Percent
100
80
60
40
20
0
04:00 06:00 08:00 10:00 12:00
User Now: 16.5% Min: 16.2% Avg: 16.5% Max: 17.1%
Nice Now: 0.0% Min: 0.0% Avg: 0.1% Max: 0.3%
System Now: 17.9% Min: 17.1% Avg: 17.8% Max: 18.3%
Wait Now: 1.9% Min: 1.8% Avg: 1.9% Max: 2.1%
Idle Now: 63.7% Min: 63.4% Avg: 63.7% Max: 64.0%
aslamaas cluster Network last custom
Bytes/sec
20 M
15 M
10 M
5 M
0
04:00 06:00 08:00 10:00 12:00
In Now: 9.3M Min: 6.7k Avg: 8.3M Max: 11.8M
Out Now: 9.2M Min: 6.7k Avg: 8.2M Max: 11.7M
RRDTOOL / TOBI OETIKER
aslamaas cluster load_one
141.28.98.221
141.28.98.222
aslamaas
load_one
1.0
0.5
0.0
800 m
600 m
400 m
200 m
0
06:00 12:00

aslamaas cluster **load_one** last **custom** sorted **by name**

Metric load_one Show Hosts Scaled: Auto Same None Size small Columns 4 (0 = metric + reports)

Show only nodes matching Filter Max graphs to show all Sorted ascending descending by name

(Nodes colored by 1-minute load) | Legend

Figure 5.4: Ganglia Cluster Overview

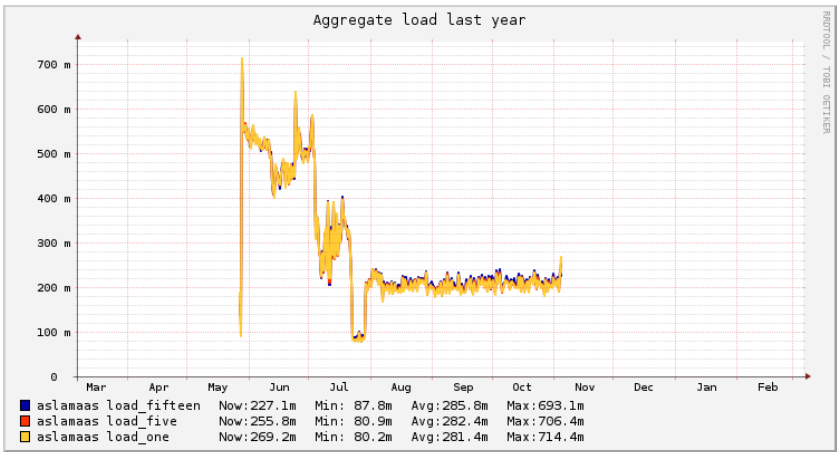


Figure 5.5: Cluster Load Aggregation

the description of the Meta Object Facility (MOF), specified by the Object Management Group (OMG). MOF describes a special metadata architecture and itself uses the highest abstraction layer M3. The SLA* model is on layer M2, this corresponds to the Platform Specific Model (PIM). The SLO-A Format is one abstraction layer below (M2). This describes the Platform Specific Model (PSM). So the SLO-A Format is on the same level like SLA(T) description of the SLA@SOI project, as shown in Figure 5.6. For the reference model of SLO-A, it has not been used with the primitive data types of the SLA* model, instead the EMF data types of the Eclipse module Ecore are used. This was necessary to get a functional prototype.

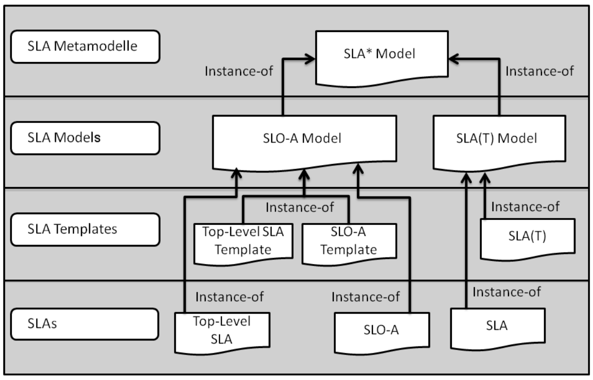


Figure 5.6: Model Based Development of SLO-A

The main goal of the format is to develop an adaptable, adequate machine-readable agreement which is legally binding. To get customer-oriented, runtime-adaptable SLAs it is first divided into a static and a dynamic part. The static part, like contract partner IDs, addresses, etc. is important, but more interesting is the part changeable during runtime, focused on SLOs like max. scaling limit, backup period, etc. This dynamic SLA part must be monitor-able and controlled by the customer.

Within the SLO-A format, it is possible to define Top-Level SLAs and SLO-Agreements (SLO-As) which are single and dedicated for each SLO.

5.1.3.1 Top-Level-SLA to a SLO

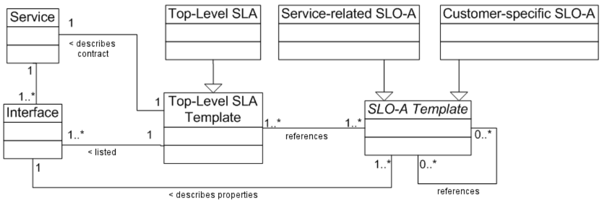


Figure 5.7: Overview of SLO-A Format

All Top-Level SLAs and SLO-Agreements are based on an appropriate and pre-approved template (see Fig. 5.7. Also, a Top-Level SLA always is displayed by a service which can have one or more interfaces, which are the base of the agreement. On the contrary, A-SLO-A is not based directly on a service only with the interfaces. Here the actual technical or contractual responsibilities can be modelled.

Resulting characteristics of the model:

- Technical services are characterized by a one to one assignment from a Top-Level-SLA to an SLO template, it's a so-called service-oriented SLO template. Also, SLO-A templates can be referenced to another SLO-A template and build a hierarchy.

- SLO-As can be grouped by a higher SLO-A or a Top-Level SLA. This builds a hierarchical tree where each entity is referenced to the next higher SOL-A with a UUID and a naming convention. So an entity directly references the low SLO-A and monitoring can be done overall.

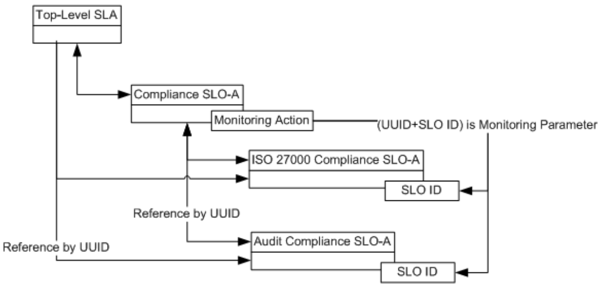


Figure 5.8: Example of References within SLO-A

5.1.3.2 Top-Level SLAs and Top-Level SLA Templates

Basically, the Top-Level SLA comprehend static contractual information and only a few dynamic pieces of information. Also, there can be made references to outsourced documents like general terms and conditions. A Top-Level SLA necessarily must have information about the accounting of services, IT continuity plan, service development plan, terminology, escalation plans, guidelines for priorities, responsibilities of both customer and company, data of both parties, service times, accomplishment penalties, signing, an ID, how reports have to been done and how they were displayed in which frequency, the interfaces with ID and description, references of SLO-as with UUID and naming convention, and the termination reason.

For a legally binding SLO-A, it has to contain an SLO-A identifier, contact data, service time. Also, the SLO itself with an ID, a value and data type which has exactly one interface, which priority, if needed the allocation to other SLO with their interfaces, the boundary a marginal values. The SLO-A additionally should contain the SLO-A reference, the accounting, accomplishment penalties, monitoring, how and with the interface it is done, how the reporting is done and the termination clause.

5.1.3.3 Workflow for a Top-Level Sla or SLO-A

To generate a Top-Level SLA or an SLO-A, the customer first has to fill in the customer data, service times, how it will be paid and how it should be reported with which interfaces. In comparison to an SLO-A, there also has to be filled in how it should be monitored. Afterwards, the decision is made if its a standard template or a customer-oriented template. In case it is a standard all information will be checked and if necessary it is asked for a correction. Otherwise, if it is a customer-oriented template, it is asked for a special template. If the value verification is accepted at the Top-Level SLA, all SLO-A are filled in and the inferior references are built. At the SLO-A template, the reference is to the higher entity is build directly. After that, the templates got signed and the contract is ready.

The incident change management is mapped as an inferior grouped SLO-A, so they can be used as KPI. An extension of conditions and modality can involve more actions, which always have a condition, guidelines and postcondition. They describe how the action will be triggered.

5.1.3.4 Reference Implementation

The SLA Format implementation is done as part of master thesis by Ralf Teckelmann at the HS Furtwangen University [166]. There have been made many extensions to the SLA Templates proposed by the SLA@SOI project. One important part of a SLA Template (see Figure 5.9) is the extension of the attribute *Type*. It differs from the Type of the SLA Template by the possibility to have 3 values: *top-level*, *service* and *customer*. This causes an 'IF' clause to load a concrete structure into the SLA template or SLA. So that means, when *Type* is either *service* or *customer* it is an SLO-A Template or SLO-A. For the unique identification, the *UUID* attribute is used. Also, the model version can be found in the attribute *modelVersion*. The next subsections discuss the main features of the SLO-A Model in detail.

5.1.3.5 SLAs and SLA Templates

As described in the last chapter the documentation is as following: There are two segments, that contains the documentation part of the SLA/SLO-A Templates. The first segment *descriptions* keeps the general terms of the agreement. The second segment *fuDocs* (shortcut for further Documents) contains the interface description. Both structures will be described later. Normally it is possible to use SLA and SLA Templates without agreement term segment, but for compatibility reasons to SLA* respectively SLA(T) this segment is included.

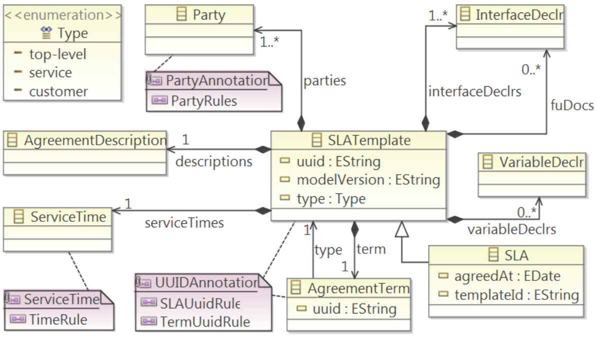


Figure 5.9: Overview of SLA Template, a modified and extended SLA(T) model [8]

5.1.3.6 Description of Agreements

The class *Agreement Description* as segment *description* of SLA templates and SLA has also its own structure. This is a different to the SLA* model. In SLA(T) there exists only a *ServiceDescription* segment without further information. In SLO-A Format the segment *descriptions* contains exactly one element of the class *Property* with its segment *entries*. These *entries* use the class *Entry* with its two attributes *key* and *value*. The attribute *key* contains the type of the following document in the attribute *value*. The type can be either a *CoverageDescr*, a *AgrementDescr* or a *Disclaimer*. The information itself is in the attribute *value*.

5.1.3.7 Time Frames

The next segment of the class *SLATemplate* is the *serviceTimes*. It describes the time the service must be available. The segment uses the class *ServiceTime* one time. All times are save there. This is also a different to the implementation in the SLA* model. In the SLA* model only a start and a stop time defined. This is not enough. So the SLA-O Format creates the class *ServiceTime* which holds in its segment *serviceTimePairs* a list of different time intervals, where an interval contains a pair of a class *entry*, where the time definition is stored.

5.1.3.8 Parties

The contracting parties will be placed in the segment *parties* of the class *SLATemplate*. Here are used the original class *party* of the SLA* model. It contains a contact point with all required information. Also the class *STND* is included which holds information about the *agreementRole*. This is normally either the *provider* or the *consumer*. Also it is possible to store *Operatives*. These are contact persons, which can be directly assosiated to *Actions*. The annotation *PartyRules* describes conditions when information be available in a SLA or SLA template, this is a deviation to the SLA* model.

5.1.3.9 Signatures in SLAs and SLA Templates

A complete new attribute is in the class *SLATemplate* the segment *ProviderSignature* and in the class *SLATemplate* the segment *customerSignature*. This contains the signature of the corresponding party. This feature allows it, to sign a SLA online. Is a SLA accepted by the customer and/or the provider, the corresponding signature will be added to the *SLA* (segment *customerSig*) and *SLATemplate* (segment *providerSig*).

5.1.3.10 Interface Declaration

The interface declaration is part of the service description and is used to connect the interfaces of a service and the SLOs. Also for the reference of further documents it is needed. There exists two ways to implement that, but both uses the abstract class *interface*. First is done by the resource oriented interface *ResVersion*. This is used to keep further external documentation in the segment *endpoints*. The class *Endpoints* contains the three attributes *location*, *id* and *protocol*, to describe how to access the document. The *location* represents the location of the file, for example an URL, the *protocol* gives the required protocol to access the file, for an URL it could be *HTTP* or *HTTPS*. The *id* contains an unique identifier for the document. The attribute *refProvider* describes, who deposit the document. Also the segment *interface* repesented by the class with the same name contains through the class *ResourceType* the type of the document. This is in upper letter the type of the document. This is for PDF document the shurtcut 'PDF'. The other possibility to describe an interface is the class *SpecVersion*. Here a Web Service can be defined by WSDL (Web Service Description Language). This is also part of the SAL* interface specification. Therefore the classes *InterfaceDeclr*, *InterfaceSpecification*, *InterfaceOperation* and *InterfaceProperties* are needed. The class *InterfaceSpecification* contains the *id* of the specification and in the segment *operations* are deposit all the operations of the WSDL part.

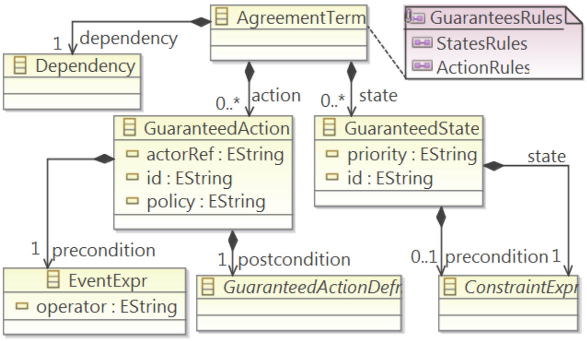


Figure 5.10: Overview of Agreement Terms

5.1.3.11 Aggreement Terms

The *AgreementTerm* segment is the core part of the SLA* Agreement and is a segment of *SLATemplate*, because it contains the states *GuaranteedStates* and the resulting actions *GuaranteedAction*. A state describes the relation between a SLO and a measure point based on a interface. In general states contains an attribute *priority*. Both (states and actions) have conditions. The actions must have a *precondition* and a *postcondition*, a state can have many *preconditions*. The representation of conditions happens by the ground expressions of the SLA* model. The actions have an attribute *actorRef*. This field contains the unit, who is responsible to work on it. The class *AgreementTerm* is contained in the segment *term* of the top level SLAs or SLO-As. This is a difference between the SLA* model and the SLO-A Format. Important is, that the class *AgreementTerm* can have only a segment *GuaranteedStates*, when the *type* is not *top-level*. Therefore are exists three state rules:

- A Top-Level SLA (Template) does not hold any states

- An SLO-A (Template) can hold one or more states
- An SLO-A (Template) without a state is a grouping KPI

For *ActionRules* it is essential that a Top-Level SLA (Template) or a SLO-A (Template) can hold any number of actions.

5.1.3.12 Dependencies

The Dependencies of the documents itself is solved simple with the classes *Dependency* and *Property*. Also there are no existing solution in the SLA* model or SLA(T). The idea behind this is, to set the documents itself in relation. All Agreement Term can have many superordinate (segment *depending*) and subordinate (segment *anteceding*) Terms over the class *Dependency* as segment *dependency*. To store the information below the class *Dependency* the class *Property* is used. The attribute *key* contains the name and the attribute *value* references the document either as URI or as UUID. Also rules can be applied. Important is the*CustomerSLO-ARule*. This allows for non customer SLO-A (see attribute *type*) only exactly one entry for *anteceding*. This structure is quiet simpler than the mechanism in the SLA* model.

5.1.3.13 Service Level Objectives

The description of Service Level Objectives contains two components. The first component is the declaration of objectives, its value and the threshold. The threshold will be defined by the variable declaration. This is a mechanism to define a *name*, a *value* and a *data type*. Also the threshold values will be added, that all required information are available. The state definition creates a connection between the in the SLO defined value and at the runtime estimated value (measure point). The second component of the SLAs is the state definition. This is done by the class *GuaranteedState*. It has a *Priority*, this can be optional set. This class connects the values defined in the SLO and the measured real time values. These information are called measure point. Important is that the name of a defined interface and a UUID of a top level template are required. This relation is known by the SAL* model but was extended. An association to an interface of a SLO-A is possible by storing an UUID of the top level SLA.

5.1.3.14 Action Definitions

The next part are the actions of the SLO-As. In general there are two different types. *Business Level Guaranteed Actions* and *Pricing Business Terms*. The first one is from the point of system functionality view and is used for the service provision. In this terms and templates are stored information that describes the execution of the system functionality. When an event is happened this information describes all further steps, which must be done. For example this could be an agreement between provider and customer about the start time of the monitoring. Also an example is after an incident, the automatically sending of messages.

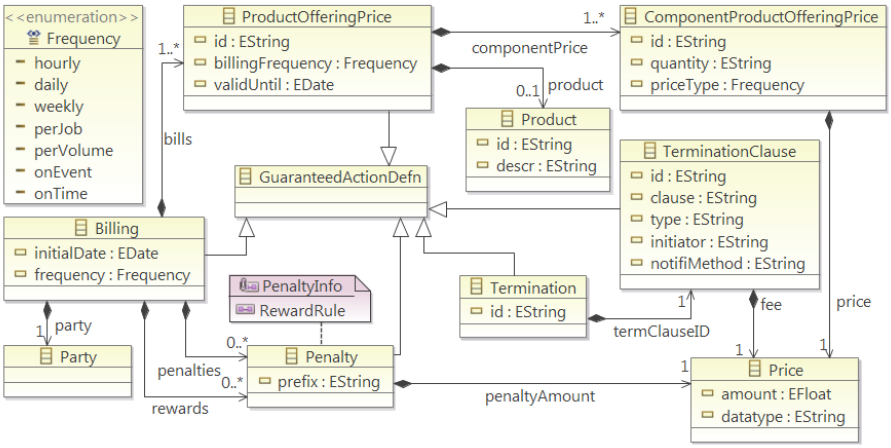


Figure 5.11: Pricing Business Term Classes

5.1.3.15 Pricing Business Terms

The *Pricing Business Terms* describes actions for the agreements and their conditions. Also actions triggers system functions, but they have influence to the service itself. Here are happens actions to costs, penalties, bonuses and termination clauses of the SLAs. A bonus and a penalty are both of the type *Penalty*, but the relation class *Billing* is either *rewards* or *penalties*. The class *Penalty* has an attribute *prefix*, which describes if it is a bonus or a penalty. Also the class *Price* was added. This contains an absolute amount. Also it depends on the relation, if it represents a Price or a free.

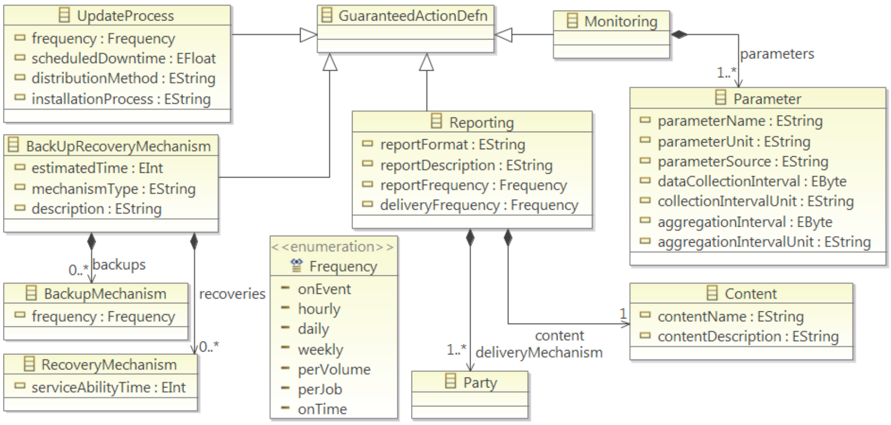


Figure 5.12: Business Level Guaranteed Action Classes

5.1.3.16 Business Level Guaranteed Actions

The *Business Level Guaranteed Actions* has currently monitoring and reporting action implemented. The monitoring classes are expanded for the SLO-As. Also the attribute *ParameterSource* was added. This allows it, to recognize properties through their names, data types and its source (UUID). Also frequencies for the definition of report, aggregation and acquisition intervals. the following frequency types are added:

- *onEvent* instead of an interval an event triggers the action
- *onTime* a constant time stamp triggers the action
- *perJob* the interval is defined by the runtime of a job
- *perVolume* a concrete value trigger the action. For example 30GB disk space is full, or 1000 connections are open

A major change is the relation to the class *Parameter*. This was changed to composition with *1:n* relation, where *n* must be at least 1. This allows the declaration of multiple parameters and together with the source of the parameter (*SourceParameter*) it is possible to assign values of the depended SLO-As.

5.1.3.17 Use Cases

To illustrate possible applications this section presents use cases and gives the corresponding sample implementations. Cloud computing hugely benefits of the self-serve principle, which underlies it. Therefore it is particular important to give customers the possibility to fill in the SLO templates with only a few steps and in an self-explanatory way. Three use cases will show the modelling possibilities of SLA-A:

5.1.3.18 Availability Use Case

For the first use case we assume that a customer runs a cloud application on an specific regular basis. Therefore he wants to ensure that the resources are available for this specific times and can be used. We assume that the customer already has completed the initial process of creating a valid SLA with an provider for his service. The contract thereby covers the specific service times, in this case every Friday from 14:00 to 18:00. This time-frame is therefore stored in the class *ServiceTime* as a *serviceTimePair*, where the key *startTime* has the value 14:00:00 and the key *endTime* has the value 18:00:00. Additionally another field is filled in where the *interval* is set to *weekly* and *Friday* is the value for the day so that all the defined agreements, for example the availability or response time are legaly binding for this specific period.

Due to the changing business environment the customer now wants to change the SLA, because he needs the same service to be available on every Wednesday from 09:00 to 14:00.

Therefore the customer loads the existing SLA and adapts the *ServiceTime* so that the new requirements are met. The respecting changes to the XML files can be seen in figure 5.13. In the real life the customer would change these settings by loading the already defined SLA into a tool with an graphical user interface like presented before and would simply change the the *ServiceTime* and generate the new SLA. This however requires that the newly chosen parameters are inside the limitations given by the provider. This validation is done automatic within the adjustment tool. If the validation succeeds the SLA gets signed online by both parties and is legally binding. In the case of mismatching customer demands and provider offerings the SLA can not be signed and a manual negotiation process has to be performed.

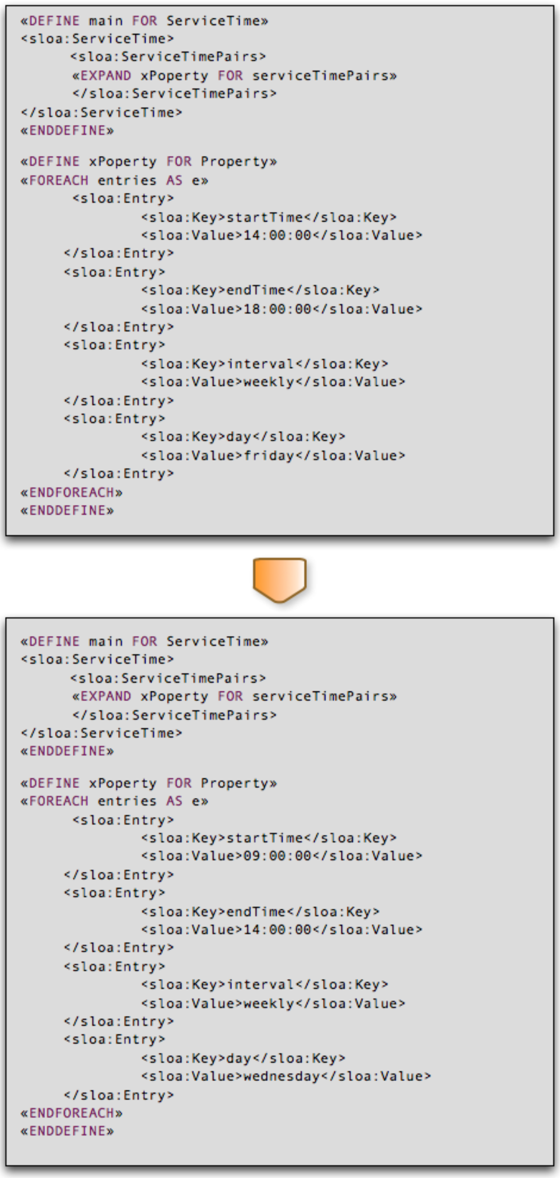
5.1.3.19 Additional KPI Use Case

As exemplary second use case a customer wants to extend an existing SLA in order to match the advanced or varied requirements of his services, so the service quality can be better monitored and ensured. In this case the customer wants to add another KPI, which represents his needs the best, to the existing SLA. For the SLA this means that another it AgreementTerm has to be added. Inside this it AgreementTerm the it GuaranteedState, itGuaranteedAction, etc. have to be specified. Additionally the it ServiceLevelObjective has to be created in order to measure and monitor the newly created KPI. If we assume that the customer wants to add the guarantee for the minimum bandwidth his service can use, this KPI and the corresponding it interfaceDeclr can be easily added by using the graphical interface like shown in figure reffig:gui. Therefore the customer chooses a representing KPI from an list of available KPIs, fills in the appropriate values and so adds the new contract clause to the existing SLA. Again it has to be checked if newly added KPI is within the providers range of offerings or not and based on that can be signed online.

5.1.3.20 Additional Resource Use Case

Another use case for using dynamic SLAs is to extend resources. For example, a customer wants to expand his computing power because of large and short term calculations. In this case the customer can order the new CPU power within the graphical interface, and he also is able to do that with time limitations. But it is necessary for the cloud provider to check the resources before he gives a guarantee to the customer for availability and pricing.

For the pricing, it is possible for the provider to give certain offers in regard to predefined bundles. In this way, the provider is able to achieve better resource allocation and usage prediction. A sample offering regarding the CPUs can be seen in figure reffig:offering. There a special bundle price for dual-core and quad-core CPUs is given. The price is calculated by accumulating bundles. In special cases, the customer can negotiate new terms with the provider. For the resource allocation, the requested demand of the customer is checked against the real-time resources of the provider. This is done automatically by inside the SLA creation tool. So after automatically



```
«DEFINE main FOR ServiceTime»
<sloa:ServiceTime>
      <sloa:ServiceTimePairs>
      «EXPAND xPoperty FOR serviceTimePairs»
      </sloa:ServiceTimePairs>
</sloa:ServiceTime>
«ENDDEFINE»

«DEFINE xPoperty FOR Property»
«FOREACH entries AS e»
      <sloa:Entry>
              <sloa:Key>startTime</sloa:Key>
              <sloa:Value>14:00:00</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>endTime</sloa:Key>
              <sloa:Value>18:00:00</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>interval</sloa:Key>
              <sloa:Value>weekly</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>day</sloa:Key>
              <sloa:Value>friday</sloa:Value>
     </sloa:Entry>
«ENDFOREACH»
«ENDDEFINE»
```

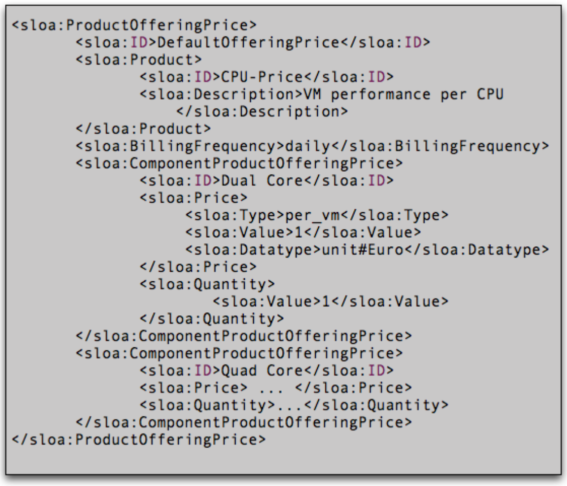
```
«DEFINE main FOR ServiceTime»
<sloa:ServiceTime>
      <sloa:ServiceTimePairs>
      «EXPAND xPoperty FOR serviceTimePairs»
      </sloa:ServiceTimePairs>
</sloa:ServiceTime>
«ENDDEFINE»

«DEFINE xPoperty FOR Property»
«FOREACH entries AS e»
      <sloa:Entry>
              <sloa:Key>startTime</sloa:Key>
              <sloa:Value>09:00:00</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>endTime</sloa:Key>
              <sloa:Value>14:00:00</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>interval</sloa:Key>
              <sloa:Value>weekly</sloa:Value>
     </sloa:Entry>
     <sloa:Entry>
              <sloa:Key>day</sloa:Key>
              <sloa:Value>wednesday</sloa:Value>
     </sloa:Entry>
«ENDFOREACH»
«ENDDEFINE»
```

Figure 5.13: ServiceTime adjustment by the customer

checking the requested resources and calculating the new price the adjusted SLA is available for the customer and can be signed.

5.2 QoS Management Modules

This section presents the different approaches (threshold values, fuzzy, neural networks, linear regression) to manage the KPIs of a Cloud SLA. These methods have been choose because on the one hand, with the threshold value system, they reflect the current state of technology in most



```
<sloa:ProductOfferingPrice>
       <sloa:ID>DefaultOfferingPrice</sloa:ID>
       <sloa:Product>
              <sloa:ID>CPU-Price</sloa:ID>
              <sloa:Description>VM performance per CPU
                  </sloa:Description>
       </sloa:Product>
       <sloa:BillingFrequency>daily</sloa:BillingFrequency>
       <sloa:ComponentProductOfferingPrice>
              <sloa:ID>Dual Core</sloa:ID>
              <sloa:Price>
                   <sloa:Type>per_vm</sloa:Type>
                   <sloa:Value>1</sloa:Value>
                   <sloa:Datatype>unit#Euro</sloa:Datatype>
              </sloa:Price>
              <sloa:Quantity>
                      <sloa:Value>1</sloa:Value>
              </sloa:Quantity>
       </sloa:ComponentProductOfferingPrice>
       <sloa:ComponentProductOfferingPrice>
              <sloa:ID>Quad Core</sloa:ID>
              <sloa:Price> ... </sloa:Price>
              <sloa:Quantity>...</sloa:Quantity>
       </sloa:ComponentProductOfferingPrice>
</sloa:ProductOfferingPrice>
```

Figure 5.14: Service Offering for CPUs

cloud management systems. And on the other hand, fuzzy logic presents the classic approach to improving such controls [167] [168] [169]. Linear regression is a statistical method of modelling a target value based on independent predictors. It is commonly used for forecasting and finding cause and effect relationships between variables [170]. Different regression techniques vary based on the number of independent variables and the type of relationship between the independent and dependent variables. Linear regression is one of the most versatile statistical methods and a useful method for forecasting [171]. While NNs powerful approximation capabilities and self-adaptive data-driven modelling approach allow them great flexibility in modelling time series data, it also complicates substantially model specification and the estimation of their parameters. Therefore the comparison with support vector machines and linear regression becomes so interesting. Thus makes all the chosen approaches relevant and interesting for this field.

5.2.1 Threshold Value System

Threshold value systems are one of the simplest forms of process management tools. In Simple, a threshold is a boundary between two states, where when the actual value of the threshold is reached the system starts acting. An example of this would be a system where as soon as a defined temperature is reached it shuts down.

In the case of SLAs and QoS thresholds often are seen as the boundary between achieving the guaranteed purchased service offering or not. For example, the response time of a system is the desired QoS aspect, and the SLA guarantees the system response in less than 500 milliseconds. So the threshold value would be 500ms if the response of the system takes longer, the SLA is broken and the QoS target is failed. In reality, however, such a system would not only be set to

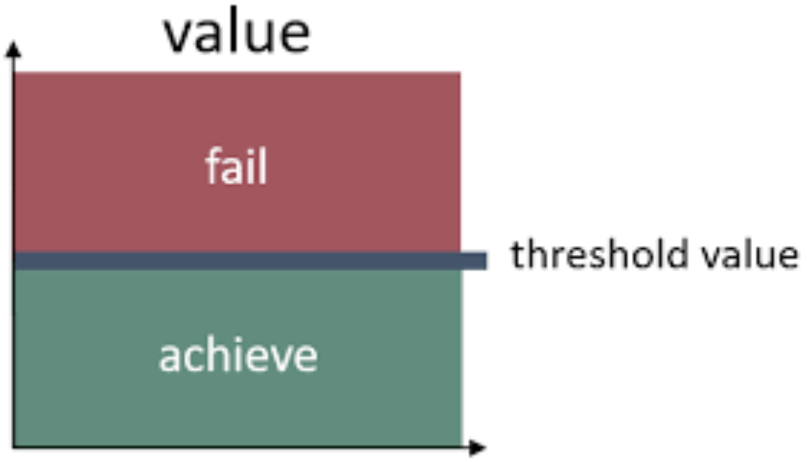


Figure 5.15: Threshold Value System

the actual value of the SLA but rather a buffer which would allow the system to react to avoid an SLA violation, would be implemented. In our example, this could be that with a response of over 300ms, free up more network resources to keep the response time from increasing.

Fuzzy Inference

Most approaches consider infrastructure sensor data like bandwidth, request/response time, CPU usage, memory usage, etc. to control the scaling infrastructure as depicted in Fig. 5.16 (dashed box). The approach of this thesis is to use additional, often imprecise information (e.g. weather forecasts) to improve the management to meet the QoS requirements stated in SLAs. These imprecise factors (e.g. the user wants to scale aggressive/moderate, etc.), political factors (legal changes, political summits, etc.), economic/market factors (product advertising, product launch, etc.), other factors influencing the service usage (e.g. weather, gossip, etc.) can not be modelled precisely.

Fuzzy logic allows the modelling of imprecise information by the user in the form of non-numeric linguistic variables (e.g. weather: bad/good/excellent). These fuzzy inputs are used in the fuzzy control system, that uses expert knowledge to inference a fuzzy output. After defuzzifying this output to a crisp value, then this states how big the up and downscale factor should be. For example, if a customer wants to have an aggressive scaling control the infrastructure will scale up with e.g. 3 VMs otherwise with only one VM at a time. The scaling domain expertise is modelled in a knowledge base with fuzzy IF-THEN rules.

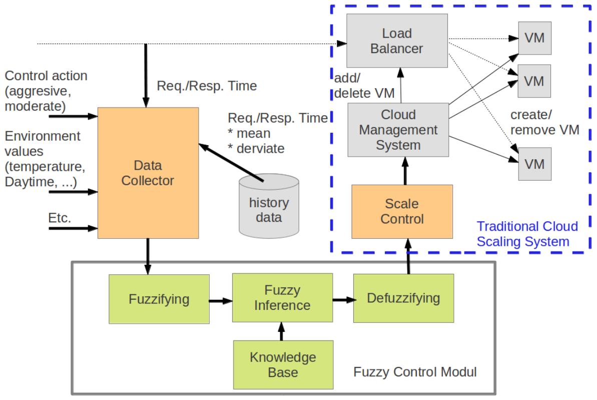


Figure 5.16: Fuzzy Controlled Scaling Architecture

5.2.1.1 Fuzzy Controled Scaling Architecture

Figure 5.16 shows the architecture for a load balanced service by automatically scaling up/down the infrastructure by provisioning/deprovisioning VMs. It consists of two new modules compared to the traditional scaling infrastructure (blue box), the *Data Collector* and the *Fuzzy Control Module*.

The *Data Collector* collects all information data, crisp (e.g. cpu usage) and imprecise data (e.g. weather). The data is categorized in infrastructure data (e.g. req./resp. time), history data (e.g. req./resp. time 5 minutes ago), control action (e.g. aggressive up/down scale), environment data (e.g. daytime), and other information that might influence the load of the service. Some of the data is automatically monitored, like the infrastructure data, some is computed like the history data, and some is determined by users/experts like control action or environment data. All collected data is input data to the *Fuzzy Control Module*, where the data is fuzzified, results are propagated by the fuzzy inference engine and quantified by defuzzification. The defuzzified value (Number of VM to be started or stopped) is put into the *Scale Control* module, which generates XML-RPC calls to the *Cloud Management System*.

5.2.1.2 Collected Information Factors for the Fuzzy Control

The relevant information to improve elasticity can be categorized into the following monitoring data: infrastructure, infrastructure history, time-dependent, and service-dependent sensor data. Infrastructure sensor data includes factors that can mostly be monitored using sensors placed in various locations and layers of the cloud infrastructure. KPIs, like request response time, which can easily be measured at the load balancer (LB). Cloud specific parameters, like start time of VMs, can be acquired at the cloud management system. The quality of the cloud infrastructure or service implementation can be taken into account as well. The load balancing control might be influenced by the basic robustness of the overall infrastructure. The infrastructure robustness can be modelled by an imprecise parameter e.g. strong, weak. Infrastructure history sensor data are parameters that have been previously collected into a history database. The purpose is to calculate values like mean value, derivation value, etc. These statistical data can be good indicators to improve the LB management. Imprecise history parameters can be of interest as well. Suppose a service depends on the weather condition (e.g. online shop for winter tires), then a sudden change of the weather condition from dry to snowy condition makes it more likely, that the load of such a service is higher. Time-dependent sensor data contain parameters that can influence the infrastructure management at a predefined time. The knowledge of the typical weekly usage for the service (see Fig. 5.17) can be modelled and therefore the decision to scale up or down strongly or weakly depending whether the change is high or not.

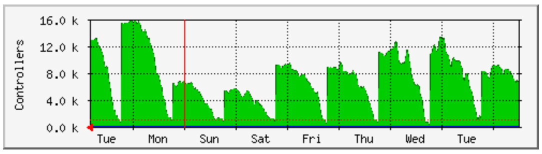


Figure 5.17: Example: Weekly Load of the HFU Learning Management Platform

Service-dependent sensor data involve parameters that influence the control infrastructure depending on the related service. Political parameters, like new legal issues enforcing more logging at the service side. Market events, like product launches, marketing events, new prices, etc. can influence the usage of services. Gossip, modelled as good news or bad news is influencing service usages. Importance of service might need a more aggressive management to make sure, that the SLA violations can be minimized.

5.2.1.3 Fuzzy Control Module

The *Fuzzy Control Module* consists of four main fuzzy control processes represented by the four sub-modules respectively (see Fig. 5.16). The crisp and imprecise input data is converted into fuzzy values for each input fuzzy set with the *Fuzzifying* module. The decision making logic of the *Fuzzy Inference* module determines how the fuzzy logic operations are performed (SUP-MIN inference), and together with the *Knowledge Base* module determine the outputs of each fuzzy IF-THEN rules. Those are combined and converted to crisp values with the *Defuzzification* module. The output crisp value can be calculated by the centre of gravity or the weighted average and are converted to the number of VM to started or stopped.

Fuzzification Fuzzification is the process of decomposing the input data into fuzzy sets, with trapezoidal shaped membership functions. Figure 5.18 shows a system of fuzzy sets for an input with trapezoidal membership functions. Any particular input is interpreted from this fuzzy set

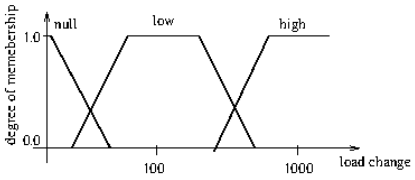


Figure 5.18: Input Fuzzy Set for Load Deviation

and a degree of membership is interpreted. If the request-response-time, for example, it set to about 100 request-per-seconds, the fuzzy value *loaddeviation* is set to *low*.

Fuzzy Inference The fuzzy values gathered from the input data are processed by the inference engine using the expert domain knowledge modelled as fuzzy IF-THEN rules. The following fuzzy rules are examples how to state the domain knowledge in the area of up and down scale control.

```
IF ReqRespTime_rising=high AND
   expected_ReqRespTime_rising=high AND
   product_launch=now AND
   ....
THEN
   up_scale=very high
...
```

Defuzzification After the fuzzy reasoning the resulting linguistic output variable (e.g. scale up = high) needs to be translated into a crisp value (e.g. number of VMs to be started or stopped at time). Defuzzification maps the output from the fuzzy domain back into the crisp domain. The most common defuzzification methods is the Center-of-Area (C-o-A) often referred to as Center-of-Gravity and is defined as follows:

$$x^* = \frac{\int \mu_i(x) x dx}{\int \mu_i(x) dx} \tag{5.1}$$

where x^* is the defuzzified output, $\mu_i(x)$ is the aggregated membership function and x is the output variable. The C-o-A method calculates the area under the scaled membership functions and within the range of the output variable and afterwards calculates the geometric center of this area.

5.2.1.4 Simulation Environment

In order to test the feasibility of our approach, we created a simulation environment to be capable of validating the general fuzzy controlled scaling architecture proposed in this paper. The simulator, therefore, consists out of four major modules (see Fig. 5.19). Firstly, the *request* *generator module*, which simulates real-life usage by generating requests to the cloud service. In our scenario, we choose a system where an HTTP request is sent to a service, which then starts a calculation based on the request and sends an answer back to the requester. It is important to note, that in our simulation requests are generated with a static predefined workload of the same length. Based on a load pattern the number of simultaneously send requests per second is adjusted to simulate various fluctuations in demand. The system was locally built and consisted on an OpenStack [172] cloud system with two hardware nodes, each could run up to 20 virtual machines, which included the cloud service responder. The second component is the *load balancer module*, which distributes the generated requests equally to the pooled Virtual Machines (VMs) by round-robin. Here also the request-response time is determined, therefore the time from generating the request until the response arrives at load balancer is measured. Within the *logic modules* this determined request response times then are checked and based on the stored fuzzy or conventional rules will be decided to scale the service up, down or do nothing. And finally, the *scaler* , which connects to the actual cloud management system, which is responsible for the provision and adding or removal of resources. The used conventional rules are represented by a simple boundary system based with a low and high threshold. When the measured average request-response hits the upper boundary a VM gets started or when the lower boundary is hit a VM is stopped. These boundary systems are widely used and provide a common way of QoS management. The here described system can be seen as a common client-server system as it is used everywhere these days. By using HTTP requests, a load balancer and simple cloud management system components, this system can be compared to most common internet services deployed nowadays. It was very important to choose a general environment to derive high informative value.

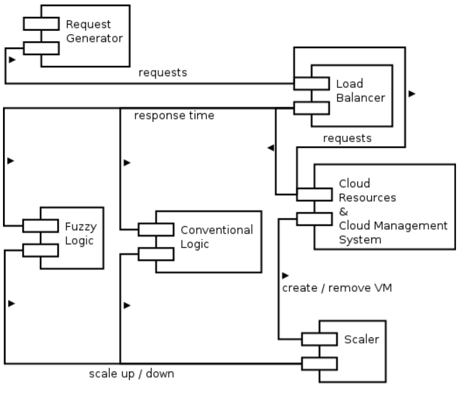


Figure 5.19: Simulator Module Diagram

The fuzzy set basically uses the same boundaries as the conventional rules, but as additional decision factors, a prediction based on expert knowledge or outlooks is used. This additional information allows an earlier response to the changing loads and thus a better reaction to upcoming demands. Based on historical data the approximate usage and thus the load can be determined for a service. Figure 5.17 shows the historic data of the load for a service during a week, based on this a simplified load prediction during daytime can be derived. Based on such knowledge an expert with domain knowledge specifies whether the load will expectedly be increasing at a high, regular or low rate. A predicted regular load will result in a rule set that equals the conventional boundaries, therefore resulting in essentially the same behaviour. In case of a high prediction, the scaler generally scales up faster, which means VMs are started on a lower load and additionally up to two VMs can be started at the same time based on the load. Simultaneously the scale down boundary is set to a lower load to keep a higher pool of available VMs. The low prediction is in principle a reversed high prediction, which will change the behaviour into generally scale up later and scale down faster and up to two machines at once. As an additional basis for the decision, the slope of the last measured points is used. The basis for this assumption is that the strong growth of response time indicates an upcoming peak load. The slope factor is therefore calculated of two response time values with the following formula:

$$slope_a = \frac{\Delta Y}{\Delta X} = \frac{Y_2 - Y_1}{X_2 - X_1} \tag{5.2}$$

Where Y and X are the time and value of the measured response time and $slope_a$ is the currently determined slope. Since the course response time is unlikely linear, the slope must be determined more than once in order to reconstruct the actual system behaviour. Problematic are load bursts, which the determined slope suddenly rises or falls down extremely. In the worst case this could cause a VM to be started up and immediately get shut down again or vice versa. Therefore, the average of of the last n slopes are used for the assessment.

The simulator is based on a model in which a generated request includes a static processing time of 100ms. The KPI, is measured as the request/response time, based on the average of the last 10 processed requests. Thereby the time is counted form the generation of the request, till arrival of the response after the processing at the load balancer. The QoS limit has been set to 2000ms in this model and the conventional rule set regulates at an average response time of 1500ms by up-scaling and at 1000ms by down scaling one VM at a time. To eliminate the influences of the test environment, like processor fluctuations the factor of 10 was used to all above described values.

To determine the suitability of the procedure presented, different scenarios have been created and tested with and without the fuzzy control mechanism.

5.2.1.5 Fuzzy Evaluation

In the presented scenario an complex course of load is sent to the service. In the first three minutes there is an peak demand, followed by reoccurring burst loads. This scenario simulates an peak load that doesn't flatten quickly. This could be the case for services that depend heavily

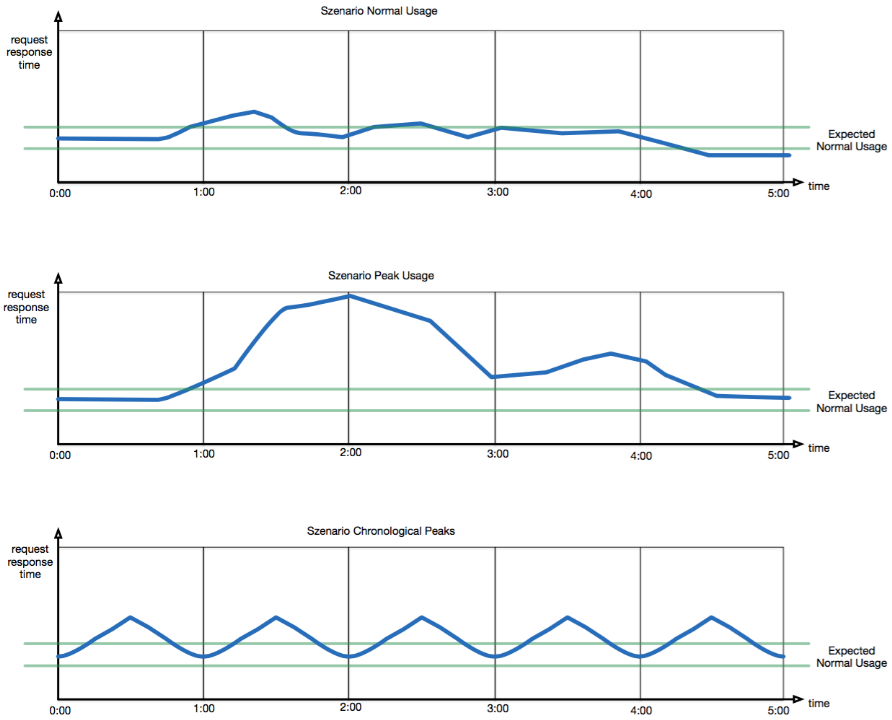


Figure 5.20: Different Usage Load Graphs

on news (e.g. launch of intensive product advertising). If the related service shows up in the news a initial large peak is produced and is recreated as smaller recurring peaks by spreading the word are following. Figure 5.21 shows load graph for this scenario.

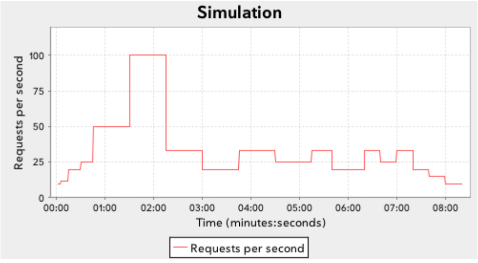


Figure 5.21: Load Graph

Result with conventional rule set: The conventional rule set result graph shows that in this test the QoS limit of 20,000ms is exceeded for the first initial peak load and is nearly hit for the following rises. The scaling lags behind compared to the load increase. So it can not cope with the increasing load fast enough by starting new VMs. This is clearly proven by the fact that the maximum number of VMs simultaneously used is only reached immediately before the flattening of the load.

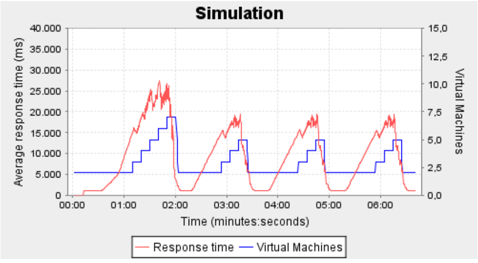


Figure 5.22: Conventional Rule Set Results

Result with fuzzy rule set (high prediction): Compared to output of the fuzzy rules with *high* prediction, as shown in figure 5.23, it becomes clear that here the VMs are much earlier for disposal allowing to reduce the response time and be inside the SLA margin. Although for the first peak load two VMs additional VMs are used, this resembles a relatively small expense in compare to breaking the SLA. It is shown later on in the simulation that better results are delivered in the smaller peaks with the same number of VMs.

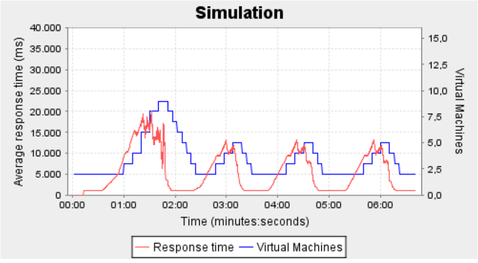


Figure 5.23: Fuzzy & "High" Prediction Results

Result with fuzzy rule set (high prediction & slope): Comparing these results with those of the fuzzy rules with *high* prediction and *slope* (see Figure 5.24), the response time has been reduced again, in spite of the very rapid scale up of VMs. The response time is in the range of all other peaks and is in this case is maintained very spacious. However, in this case an additional VM is switched on compared to the fuzzy rules with *high* prediction. The large differences in the results graphs are however not dependent on this additional resource. Instead the up-scaling of several VMs simultaneously achieves the improvement. In this case it is up to 3 VMs are allocated at the same time.

It can be concluded that with the presented approach it is clear that more resources are used which can lead to higher costs for the customer. But the response time and the SLA thresholds are maintained better and less SLA violations occur. Whether this is economically in each case must be decided by experts contemplating the SLA and contract data.

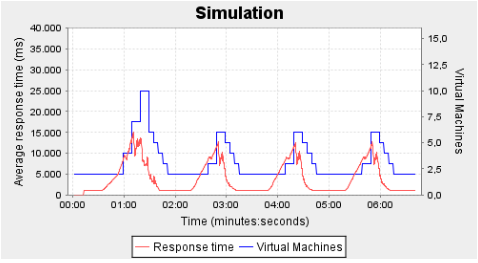


Figure 5.24: Fuzzy & "High" Prediction & Slope Results

5.2.2 Machine Learning Algorithms

To improve the output of the QoS Management Module, different machine learning algorithms were used to test the feasibility of the approach. Firstly a general Artificial Neural Network was used to predict different environmental KPIs. Afterwards an investigation and comparison of Support Vector Machines, Artificial Neural Network, and general Linear Regression was conducted.

5.2.2.1 Artificial Neural Network

The aim of this first approach was to create a prototype application which enables efficient provisioning of cloud storage resources with the use of Artificial Neural Networks to achieve better compliance with SLAs. The most common type of ANNs used for forecasting is the feed-forward multilayer perceptron (ffMLP), as seen in Figure 5.25. Artificial neural networks, in short: ANN, are networks of artificial neurons which aims to simulate the process inside the human brain and represent a branch of artificial intelligence. Artificial neural networks have been successfully used in the past with all sorts of optimization problems, which makes them a suitabel approach for our solution. The aim of this first approach was to create a prototype application which enables efficient provisioning of cloud storage resources with the use of Artificial Neural Networks to achieve better compliance with SLAs. The most common type of ANNs used for forecasting is the feedforward multilayer perceptron (ffMLP), as seen in Figure 5.25.

These are Neural Networks, which consist of one input layer, n-hidden processing layers and one output layer. Feed-forward networks are classified by each neuron in one layer having only direct connections to the neurons of the next layer, which means they have no feedback. In feed-forward multilayer perceptrons, a neuron is often connected to all neurons of the next layer, which is called completely linked. So, there is no direct or indirect connection path from neuron N_x which leads back to a neuron N_{x-z}. To compute a one-step-ahead forecast, these NNs are using lagged observations inputs of time series or other explanatory variables.

For the creation of the Neural Network model we used the graphical editor and simulator MemBrain [173] [174] . The presented Neural Network consists of 119 neurons, which are aligned

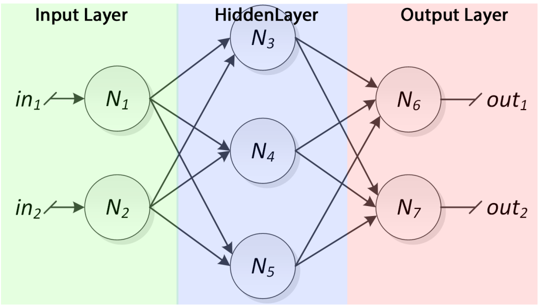


Figure 5.25: Simple 3-tier Feedforward Multilayer Perceptron.

into 5 layers, and corresponds to a ffMLP where not all neurons are completely linked. An architectural overview of the presented model is shown in Figure 5.26 below.

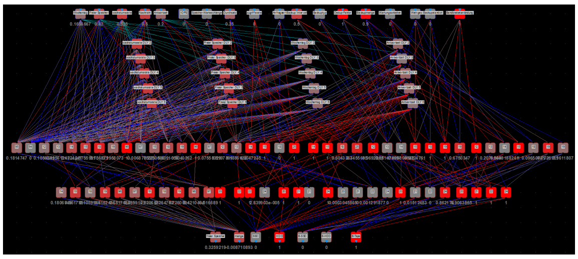


Figure 5.26: Feedforward Multilayer Perceptron Architecture.

Training of ANNs can be seen as a complex non-linear optimization problem, and sometimes the network can get trapped into a local minimum. ANNs can theoretically learn by developing new or deleting existing connections, changing the connection weights or threshold values, altering one or more of the three neuron functions (activation, propagation and output) and developing new or deleting existing neurons. In order to improve outputs, the input neurons should get normalized variables. This can simply be done by the equation below.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5.3}$$

In order to avoid local minima and bad results, the training should be initialized several times with different starting weights and alignments. For the training of the proposed model, data sets were created in the form of Comma Separated Value (CSV) files. Each file contains storage usage patters with input and output vectors. Here, 60% of the samples were used for training and the remaining 40% were used for the validation of the network. The output behaviour was modelled by depending on a input vector, where the desired output values where manually entered into the input vector. Thus, situations in which the responsible output neuron shall increase the amount of allocated memory were mapped.

To teach the network the prediction capability of future memory usage, the input vector was extended. The entire course of the used memory amount was added for the period of t_0 to t_n. The desired output for this input vector at the given time t_i shall be the predicted amount of memory used at time t_{i+x}. To achieve this, the value of the output vector at any point t_i in the time period t_0 to t_n was set to the input vector of the point t_{i+x}, by which x determines the length of the forecast period. Through this shift in values the network can be trained for a prognosis. During each training session the network error was checked with validation data. MemBrain calculates this using the following formula:

$$NetError = \frac{\sum_{i=1}^{n} (Target_i - Output_i)^2}{n} \tag{5.4}$$

The desired activation of the output neurons is here referred to as $Target$ and the actual calculated activation is the $Output$. The squared deviations are summed and divided by the number of training data sets. To determine whether the Neural Network shows good results of the output behaviour, it has been trained and validated with 10 different training data sets. The result for the network error after each learning processes is shown in Table 5.1 below.

Table 5.1: Infrastructure Service Parameter

TrainingNr.	NetError(Training)	NetError(Validation)
1	0,0000573	0,046
2	0,0000586	0,040
3	0,0000713	0,040
4	0,0000702	0,112
5	0,0000611	0,040
6	0,0000783	0,083
7	0,0000703	0,046
8	0,0000627	0,038
9	0,0000645	0,061
10	0,0000630	0,046

Here, it can be seen that the NetError reaches overall good values close to zero and not only for a particular dataset. The average total error for all training runs from Table 5.1 is 0.0000657 for trained and 0.0573 for untrained (unknown) input data.

Evaluation The aim of this prototype was to investigate, whether or not the use of a Artificial Neural Network for the provisioning of a cloud storage resources has a positive effect on SLAs compliance, and whether this can lead to a better resource utilization compared to a classic threshold value system. For this purpose we created a simulation environment where storage requests (read, write, and delete) form a generator were sent trough a QoS monitor. Inside the QoS control module, the Artificial Neural Network and the threshold value system were used to regulate the amount of allocated storage capacity. Figure 5.27 shows the architectural overview of the simulation environment.

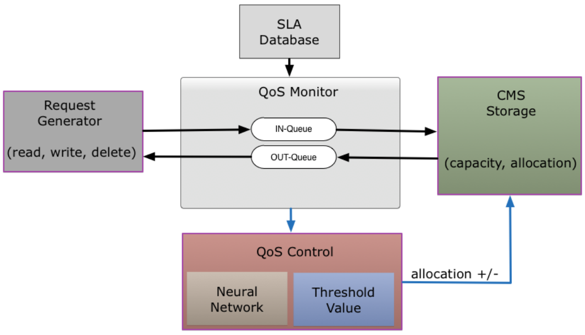


Figure 5.27: Simulation Architecture.

In the simulation, the impact of regulatory mechanisms on the following key performance indicators was considered:

- Free memory amount: providing an optimal amount of memory by the control logic.
- Response time: compliance with the KPI response time by adjusting the storage medium.
- Backup Media: proposal of a suitable backup medium.

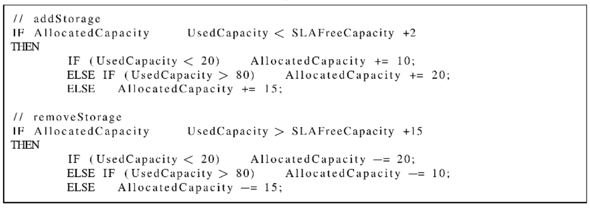
For this, the used Neural Network consisted of 11 different input neurons. Table 5.2 lists the used input neurons and describes the used input factors. As output neurons, there is one neuron that gives the expected used memory amount for the next simulation step, a neuron that determines the amount of memory to be added or removed, as well as other neurons that recommend the optimal backup medium.

In order to compare the results of the Neural Network with a common, in practice widely used method, a threshold value based scaling was implemented. This regulation system is controlled by predefined thresholds for the monitored KPI values. The implementation for the threshold rules for adding and removing allocated storage can be seen below in Figure 5.28, as simple pseudo-code if then rules.

Here, it can be seen that, by falling below a 2% buffer of the storage value defined in the SLA, the allocation will be increased and by exceeding 15% over the amount of storage defined in the SLA, the allocation will be lowered. The amount of which the allocated storage will be changed is dependent on how much the overall storage usage is. In case of an usage of over 80 %) increase will be 20%, with an usage of below 20% the increase will be 10% and in between the increase is 15 % of the overall volume. These settings are reversed for the deallocation of the storage.

Table 5.2: Simulation Input Neurons

Neuron	Description
Time	Point t_i in $t_0 ... t_n$
Weekday	Day of week for point t_i
Free Storage Capacity	Free storage capacity at point t_i
Growth Rate	Change of capacity from t_{i-1} to t_i
Response time	Mean response of last 5 inputs $\frac{\sum_{i=i-5}^{i} t_i}{n}$
Queue Length	Still open request at point t_i
Troughput	Troughput at point t_i
Access Rate	Amount of requests per time slot
Request Type	Distinction between large and small requests
Backup Amount	Size of backup data
Bandwidth	Usable bandwidth at point t_i



```
// addStorage
IF AllocatedCapacity    UsedCapacity < SLAFreeCapacity +2
THEN
        IF (UsedCapacity < 20)    AllocatedCapacity += 10;
        ELSE IF (UsedCapacity > 80)    AllocatedCapacity += 20;
        ELSE   AllocatedCapacity += 15;

// removeStorage
IF AllocatedCapacity    UsedCapacity > SLAFreeCapacity +15
THEN
        IF (UsedCapacity < 20)    AllocatedCapacity −= 20;
        ELSE IF (UsedCapacity > 80)    AllocatedCapacity −= 10;
        ELSE   AllocatedCapacity −= 15;
```

Figure 5.28: IF THEN rules for threshold system.

For the scenario in this simulation, a dynamic storage SLA, in which a customer gets granted 10GB of free space and up to 100GB of overall usage, was assumed. With such a dynamic limit described in the SLA, it is particularly important for the provider to find a solution that is as close as possible to the guaranteed amount of storage, since this will ensure a high economic efficiency. In practice, however, this usually is not possible. For this reason and because a violation of the SLAs can have monetary consequences, bigger buffer zones are installed. Figure 5.29 shows the resulting graph of the simulation with the conventional threshold value rules.

The red graph in Figure 5 and Figure 6 shows the course of the memory usage in GB, by the user during the simulation. The usage has been pre-generated for the simulation purpose and shall resemble a system, where a user regularly creates and deletes files with up to 15GB size, as well as generate larger files with up to 50GB. This type of usage may occur while working with different media files, like in the post-processing of movie projects. The green line marks the guaranteed amount of storage available to the user, granted by the SLA. It proceeds synchronous to the red graph, since the user gets guaranteed 10GB more than they currently use. The blue graph shows the pre-allocated amount of storage, which is directly usable by the user.

If we compare these results with those obtained by the Neural Network controlled storage allocation, shown in Figure 5.30, it becomes clear that the efficiency is marginally improved.

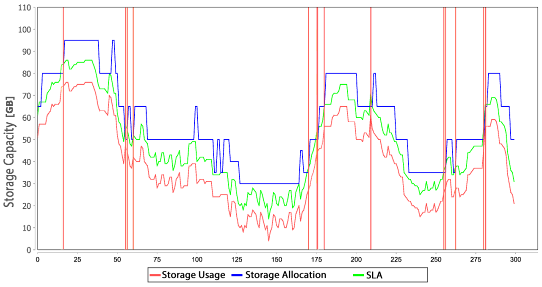


Figure 5.29: Storage allocation results for threshold rules.

With an average of 18.68% of memory over provisioned the threshold value system is almost as effective as the neural network, with a 18.22% overhead. The slight difference arises from the fact that the allocation offered by constantly adopting, fits to the SLA limits with a relatively constant overhead. In contrast, the threshold value system initially provides too much memory, and then only adopts the amount of allocated storage shortly before a violation of the SLA it to occur.

While comparing the two graphs, we see that the threshold system due to the fixed thresholds less often adjusts the amount of memory (blue curve). Since the added / removed amount of memory operates with a fixed predefined value, often too much memory is provided and then immediately gets removed again. This happens likewise when reducing the amount of memory allocated, which often leads to falling below the specified minimum amount in the SLA. However, the Neural Network determines constantly, based on the learned training data, a variable amount of memory that is to be added or removed, which leads to adequate reactions and a slightly better economic result.

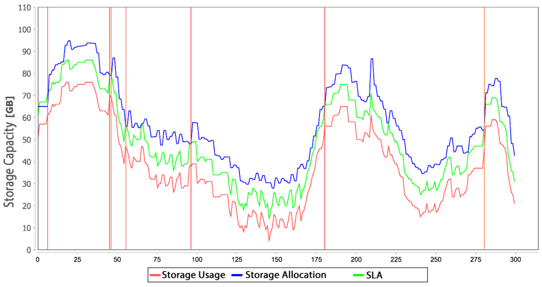


Figure 5.30: Storage allocation results for NN.

However, when comparing the number of SLA violations, it becomes clear that the Neural Network approach delivers a significantly better solution. This is also evident in the resulted graph seen in Figures 5.30 and 5.29, where the SLA violations are indicated by vertical red lines.

These exemplary results of the simulation show that the Neural Network produces 7 and the threshold value system 13 SLA violations. These results were also confirmed within the other test runs, where the Neural Network generates an average of 7.45 violations per run and the threshold values system of 15.03 SLA violations per run. Overall, the Neural Network generated solution for the provisioning of storage is better suited, since the number of SLA violations is significantly lower. Together with the slightly lower overhead makes this a reasonably good solution.

5.2.2.2 Linear Regression and Support Vector Machines

Regression is a statistical method of modelling a target value based on independent predictors. It is commonly used for forecasting and finding cause and effect relationships between variables [170]. Different regression techniques vary based on the number of independent variables and the type of relationship between the independent and dependent variables. Linear regression is one of the most versatile statistical methods and a useful method for forecasting [171]. Here the number of independent variables is one and there is a linear relationship between the independent and dependent variable. The variable we are predicting is called the criterion variable and is referred to as Y. The variable we are basing our predictions on is called the predictor variable and is referred to as X. In simple linear regression, the predictions of Y when plotted as a function of X form a straight line. The accuracy of the prediction depends on the correlation or the strength of the linear connection between the criterion and the predictor variables. The higher the correlation, the more accurate the prognosis can get [175].

Support Vector Machines (SVM), is a universal learning procedure based on the statistical learning theory. SVM can be used for both regression and classification tasks, but it is mostly used to in classification objectives. Thereby SVM have similarities to neural networks classification. SVMs however, use simpler and thus more comprehensible mathematical methods. The SVM algorithm was develeoped by V. Vapnik on the classification of data [176] [177]. A SVM constructs a hyperplane or a set of hyperplanes in an N-dimensional space, which can be used for classification, regression or other tasks. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the width between classes reinforcement so that future data points can be classified with more confidence. Hyperplanes thereby are the decision boundaries with which the data points are classified. Data points that lie on either side of the hyperplane can be assigned to different classes. The dimension of the hyperplane depends on the number of features. If the input is 2, the hyperplane is only one line. If the input is 3, the hyperplane becomes a two-dimensional plane and so on. Here support vectors are data points that are closer to the hyperplane and affect the position and orientation of the hyperplane. With these support vectors, we maximize the edge of the classifier. Deleting the support vectors changes the position of the hyperplane. These are the points that help us build our SVM.

5.2.2.3 Machine Learning Evaluation

In order to apply and evaluate different Machine Learning algorithms, the open source software RapidMiner [178] was used. The log files created by the CloudSim application were utilized as training and test sets. Furthermore, the available Series extension provided by RapidMiner was used. This extension enables an efficient way to quickly replace different Machine Learning algorithms during the process of creating and evaluating a model in regard to ordered time series.

With the help of a horizon of h=20 (2 seconds), it was defined that the learning algorithms gets to learn the next h time steps in order to be able to predict the value of the average response time of h+1. After the prediction, the time window is incremented by 1 and the next value gets predicted.

- **Neural Network:** Feed forward NN, training back propagation | Hidden layers: 8 | Training cycles: 1000 | Learning rate: 0.3 | Momentum: 0.2 | Decay: true | Normalize: true | Error epsilon: 0.00001
- **Support Vector Machines:** Kernel Type: radial | Kernel Gamma: 1.0 | Kernel cache: 200 | C: 0 | Convergence epsilon: 001 | Max iterations: 10000 | Scale: true | L pos: 1.0 | L neg: 1.0 3) Linear Regression: | Feature selection: Iterative T-Test | Max iterations: 1.0 | Forward alpha: 0.05 | backward alpha: 0.05 | eliminate co-linear features: true | min tolerance: 0.05 | use bias: true
- **Linear Regression:** Feature selection: Iterative T-Test | Max iterations: 1.0 | Forward alpha: 0.05 | backward alpha: 0.05 | eliminate co-linear features: true | min tolerance: 0.05 | use bias: true

Scenario 1 The first testing scenario resembles a classical burst load. Here the server is hit with short bursts of high demand. In out test the requests at the server are highly increased during 10 seconds followed by a phase of low requests. This pattern ist repeated four times during the course of this test. This type of load can occur, for example, on servers that are attacked by botnet (denial of service attack). Alternatively, this pattern can also occur with automated access to APIs or synchronized access to data. In reality, this kind of load is particularly difficult to process since you can adjust it badly and depending on the scale used the system simply can be too slow.

- *Neural Network:* In figure 5.31 it can be seen, that the NN overestimates the peak of the burst loads in every case. Also it can be seen that the difference between predicted peak and real peak is the biggest during the first burst and that there is an improvement when predicting the later peaks of the bursts. The briefly following decline and rise after each

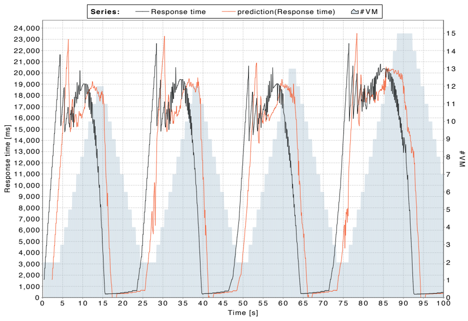


Figure 5.31: NN Scenario 1: 0s-100s

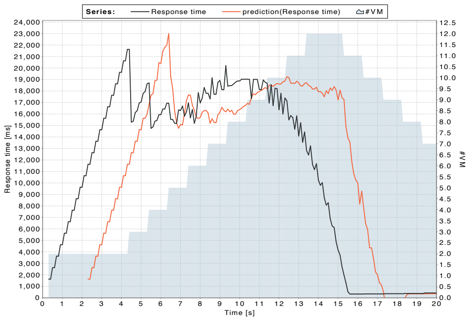


Figure 5.32: NN Scenario 1: 0s-20ss

peak, e.g., during sec 8-15 is respectively underestimated and overestimated but it can clearly be seen that there is an improvement in the last iteration. Figure 5.32 shows the delay characteristics and that the algorithm in general can adapt well to the problem.

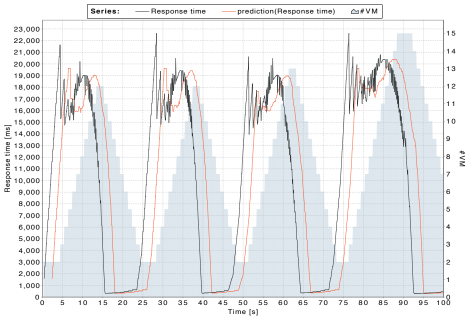


Figure 5.33: SVM Scenario 1: 0s-100s

- *Support Vector Machines:* Figure 5.33 shows a contrast to the NN algorithm. In the case of SVMs the first peak of each burst is underestimated. The following cooldown phase before the second peak of each burst is overestimated but an improvement over time can

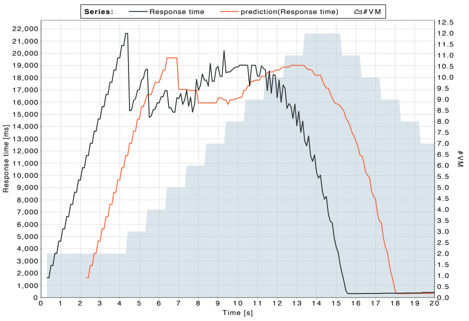


Figure 5.34: SVM Scenario 1: 0s-20ss

be seen, especially on the last burst. Figure 5.34 looks specifically at the first burst and a comparison to the NN 5.32 makes it clear that SVMs predict a more smooth curve. It should be kept in mind that it is realistic to assume that in real life there are scenarios with different requirements regarding the reaction to those predictions where this specific differences, smooth or rough, could be seen as either an advantage or disadvantage.

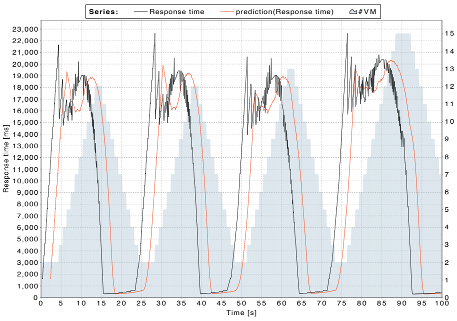


Figure 5.35: Linear Regression Scenario 1: 0s-100s

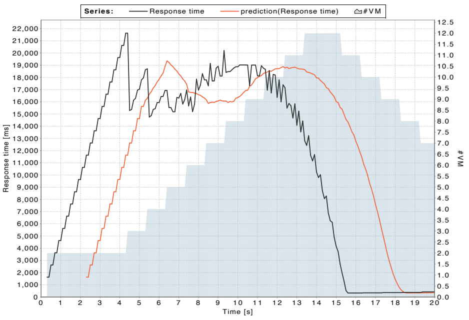


Figure 5.36: Linear Regression Scenario 1: 0s-20ss

- *Linear Regression:* The predictions made with the help of a Linear Regression learner shown in figure 5.35 seem to be very similar to those predictions made by the SVM in figure 5.33. This insight is further substantiated by taking a closer look at the bursts in figure 5.36 and figure 5.34 where a similar prediction pattern can be seen. Worth mentioning is that the predictions made by Linear Regression lead to an even smoother curve compared to the curve predicted by the other 2 algorithms.

Scenario 2 The second scenario shows a system with a normal usage load. The load slowly builds up and then slowly decays again. This scenario in this form or alternatively with strongly decaying load at the is a normal use of the system such as a file transfer or the curl of an larger API. The scenario duration of twenty seconds is in the normal real life range for such actions. This load is easier to scale because the duration and therefore the speed needed to process this is lower.

- *Neural Network:* When looking at the overview in Figure5.37 it is demonstrated that moderate changes in response times are learned rather well.

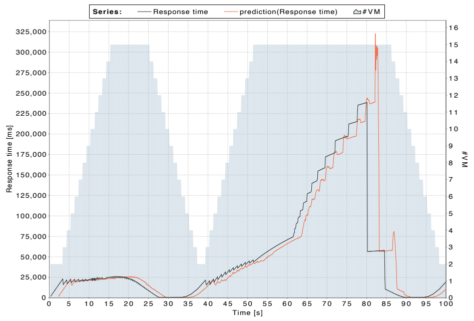


Figure 5.37: NN Scenario 2: 0s-100s

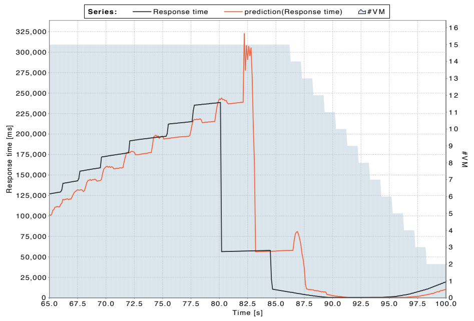


Figure 5.38: NN Scenario 2: 65s-100ss

The interesting part, shown in more detail in Figure 5.38, showcases the nature of overestimation. Again, the peak is overestimated, but the predicted curve recovers very fast and yet this issue occurs again after the second plateau. It should be noted that with a different configuration of the NN algorithm a very different curve can be predicted. For this paper we looked at a specific configuration of the algorithm because this characteristic can be utilized and will be explained during the comparison of the algorithms.

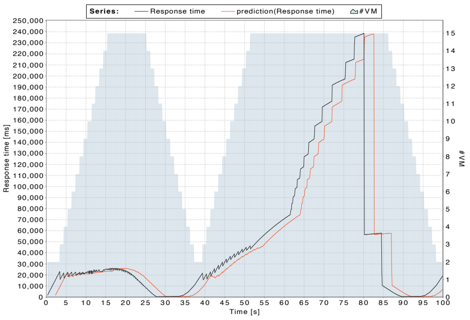


Figure 5.39: SVM Scenario 2: 0s-100s

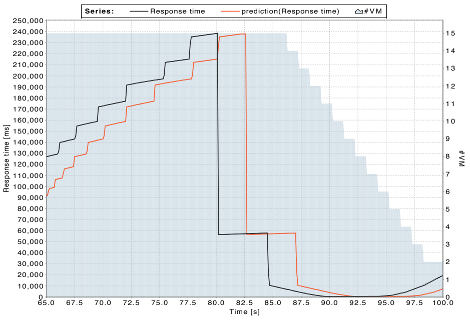


Figure 5.40: SVM Scenario 2: 65s-100ss

- *Support Vector Machines:* The overview shown in Figure 5.39 displays the capability of the algorithm to be able to adapt to a singular, steadily climbing response time. The prediction during the first phase (0-60s) is handled well by the algorithm. Figure 5.40 illustrates that the spontaneous and large decline in response time is learned very well. This is an important characteristic as predictions based on those quick changes could be the focus during the application in real-time scenarios. None of the other algorithms is able to predict scenario 2 this precisely.

- *Linear Regression:* Figure 5.41 shows again great similarity between predictions made with the help of Linear Regression and SVMs. Again the difference is that the ascent of the curve is predicted in a smoother way. Additionally, the spontaneous and large decline seen in Figure 5.42 is not predicted very well. The same characteristic applies on the following

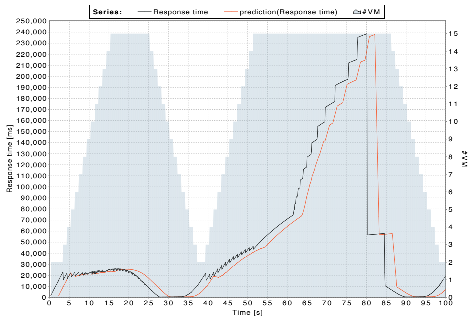


Figure 5.41: Linear Regression Scenario 2: 0s-100s

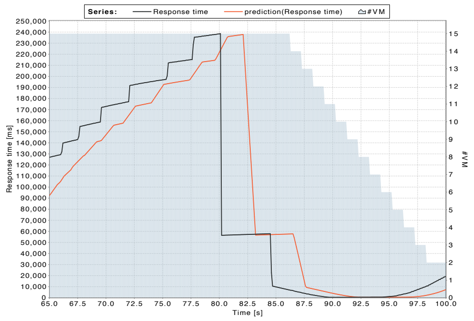


Figure 5.42: Linear Regression Scenario 2: 65s-100ss

smaller decline. As a conclusion it can be said that in this specific scenario the model trained by Linear Regression is the weakest.

Scenario 3 The third and final scenario is a mixture of the two previous ones. First, a single burst is triggered, followed by a slower but higher load, which decreases in the end. Such mixed load profiles are very often found in reality because different users and systems access resources at the same time. These loads are not synchronized and therefore can lead to chaotic scenarios.

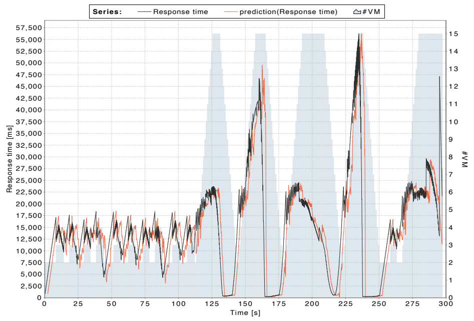


Figure 5.43: NN Scenario 3: 0s-300s

- *Neural Network:* Figure 5.43 shows that during the first phase (0-100s) the model trained by a NN has minor problems in predicting the response time. Although the peaks are generally predicted well, sometimes they are underestimated and sometimes overestimated. But in contrast to the other algorithms the difference in error margin is very small in most cases. During the recovery times after each slope, the local minima are overestimated almost in every case. While the first big peak of a response time over 42000ms is overestimated as well, the second one is predicted almost perfectly. The relative smooth slopes before, during and after the larger peaks are predicted very well with no prominent deficit.

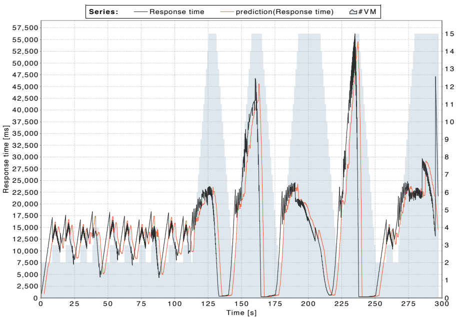


Figure 5.44: SVM Scenario 3: 0s-300s

- *Support Vector Machines:* Figure 5.44 shows that a model trained by SVMs can predict the response time for a varying scenario rather well. The occurring peaks during the first phase (0-100s) are underestimated in every case, but not to a large degree. This leads likewise to the underestimation of the recovery times after each peak, which are the consequence of adding and deleting Virtual Machines. The two larger peaks with a response time of over 42000ms are underestimated again by a small margin while the relative smooth slopes before, in between and after are learned well with non prominent deficit in their prediction.

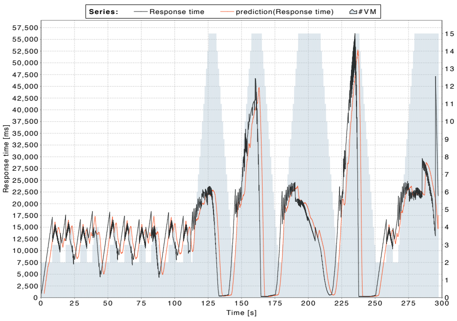


Figure 5.45: Linear Regression Scenario 3: 0s-300s

- *Linear Regression:* Figure 5.45hows that a model trained by linear regression can cope well in a varying scenario. Similar to the SVM model it slightly underestimates the response time in the first phase (0-100s). In general, it can be said that those models are very similar and have only minor, negligible differences. The main difference is that the use of Linear Regression leads to smoother slopes.

While it was shown that all 3 algorithms can be effectively used for predicting the response time in different scenarios it can be said that the NN has a minor advantage over the other algorithms. The main reason is that the NN, in general, slightly overestimates and almost never underestimates the response time. The practical application of this knowledge, e.g., using those predictions in combination with a scaler who manages the quantity of VMs leads to a more assuring state that requirements like defined SLAs can be covered more carefully than with other algorithms. In less critical business cases, where the defined SLAs and response times are not that sensitive, the other 2 algorithms, SVMs and linear regression, can be used despite their tendency to slightly underestimate response times. Especially the Linear Regression with its fast training and deployment times could be considered in near real-time scenarios.

5.3 Results

Given all the evaluation results of the different methods in this section it can be stated that every proactive approach of controlling the QoS of an cloud instance is beneficial in terms of SLA compliance. Especially the approaches which are supposed to minimize SLA injuries through the use of machine learning algorithms have proven to be effective. Similar research with respectively different focus has been conducted in the past for Neural Networks [179] [180], Support Vector Machines and deviations [181] [182] as well as for Linear Regression [183]. In detail it was shown that the machine learning based approaches come up with better management regarding the avoidance of SLA infractions. Nevertheless, the use of the simple fuzzy thresholds also offers a beneficial aspect, towards SLA compliance and QoS management and can be implemented much more easily. Additionally the effort for the prediction and the training of the algorithms should not be taken into consideration. Therefore the linear regression module has proven to be an good choice.

5.4 Holistic SLA Management

As described in the previous sections, the quality, respectively the KPIs of service can be improved by the described methods. However, this may not be enough to concentrate merely on the quality of the individual services, since in modern cloud landscapes, disruptions, failures or performance issues can occur. This requires a holistic approach, where the management of the individual service quality and the overall management of the cloud in brought together. For example, in a general cloud landscape, it is quite possible that the individual services are scaled and managed almost optimally by the algorithms described above, but the overall performance of the landscape is the limiting factor.

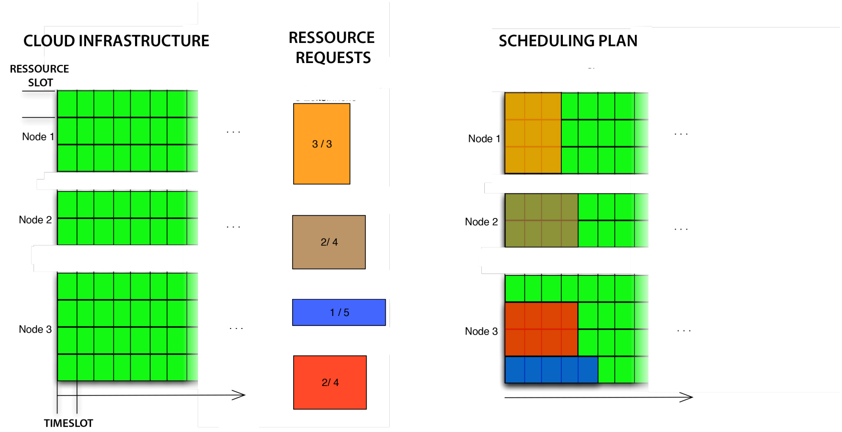
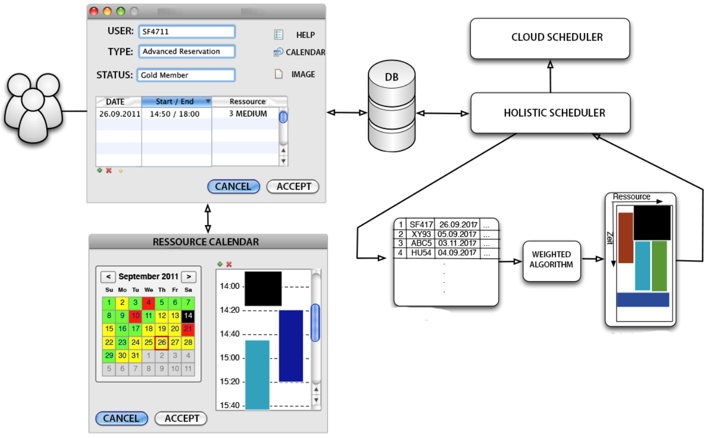


Figure 5.46: Cloud Instances Abstraction

This is where holistic SLA management comes to bear. Here, not only on each service as a single is respected but also the overall situation of the landscape is taken care of. For example, in a cloud environment where the performance is adversely affected by an infrastructure problem, this may mean that certain services are sacrificed in favour of other services. In particular, the contractual penalties and categorization of the customer regulated by the SLAs play an overriding role. In order to manage the usage of the underlying infrastructure as best as possible, either in a monetary or customer satisfactory way, an approach for scheduling the provided services is needed. To formalize this problem a customers service or instance can be viewed as a two or more-dimensional plane. For example, in the simplest way, the first dimension is the time and the second dimension the computational resources that a cloud instance needs. Scheduling these can be abstracted as seen in figure 5.46.

For this type of problem, there is already an enormous publication on solution approaches as this is known as bin packing or knapsack optimization problem [184]. Here the 2-dimensional planes are ordered and distributed as efficiently as possible in regards to packing height [185]. In particular, the algorithms for weighted packing [186] [187] [188] represents an approach that is interesting for holistic scheduling. For here, individual weights are additionally weighted and the algorithm optimizes packing to achieve optimal stacking with the highest weighting value. When deploying Cloud Instance schedules for SLAs, this could be used to map this weighting, for example, to the amount of a contractual penalty. The status of the customer could also be included in such a weighting.

As shown in figure 5.47 the model of the holistic SLA scheduling process uses a weighted algorithm to decide which customer instances and therefore SLA shall be satisfied and which ones to drop. However, this approach is only for execution in outages and emergency situations.



USER: SF4711

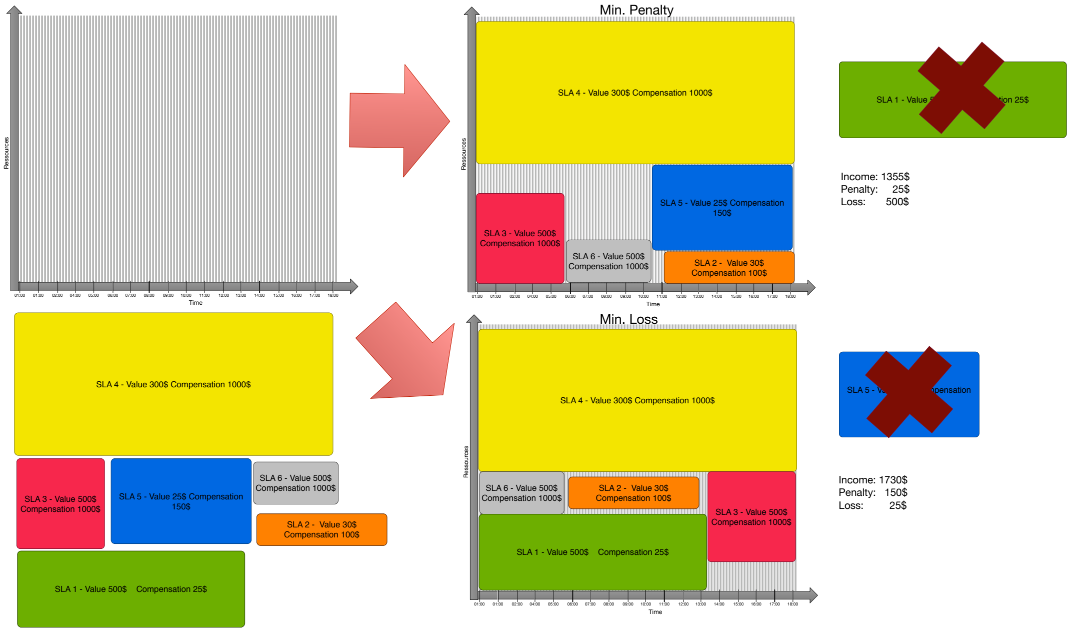
TYPE: Advanced Reservation

STATUS: Gold Member

DATE	Start / End	Ressource
26.09.2011	14:50 / 18:00	3 MEDIUM

1	SF417	26.09.2017	...
2	XY93	05.09.2017	...
3	ABC5	03.11.2017	...
4	HU54	04.09.2017	...

Figure 5.47: Holistic SLA Scheduling Model



Ressources
Time
Min. Penalty
SLA 4 - Value 300$ Compensation 1000$
SLA 3 - Value 500$ Compensation 1000$
SLA 6 - Value 500$ Compensation 1000$
SLA 5 - Value 25$ Compensation 150$
SLA 2 - Value 30$ Compensation 100$
SLA 1 - Value 5 ...tion 25$
Income: 1355$
Penalty: 25$
Loss: 500$
Min. Loss
SLA 4 - Value 300$ Compensation 1000$
SLA 6 - Value 500$ Compensation 1000$
SLA 2 - Value 30$ Compensation 100$
SLA 3 - Value 500$ Compensation 1000$
SLA 1 - Value 500$ Compensation 25$
SLA 5 - Va... ...pensation
Income: 1730$
Penalty: 150$
Loss: 25$
SLA 4 - Value 300$ Compensation 1000$
SLA 3 - Value 500$ Compensation 1000$
SLA 5 - Value 25$ Compensation 150$
SLA 6 - Value 500$ Compensation 1000$
SLA 2 - Value 30$ Compensation 100$
SLA 1 - Value 500$ Compensation 25$

Figure 5.48: Holistic SLA Scheduling Algorithms

With the approach presented here, damage can be minimized and costs for contractual penalties reduced in the event of an emergency. As shown in Figure **??**, there are various approaches to be taken in the event of dropping SLAs by the system. In this example, it is shown how on the one hand, optimization can be done after the minimum penalty fine due to SLA violation or after the minimum loss in terms of revenue or income. In this case, the provider would have to determine which criteria should be used for optimization, since this can have both monetary and legal consequences

5.5 Conclusion

In this Chapter the components and modules of the prototypical implementation have been described. A reference implementation of the individual components and the results obtained were presented and discussed. Starting with the description of the front end and the underlying components, it was possible to show how a GUI and the underlying A-SLO-A format for SLA in the cloud can be easily and clearly created, managed and operated. The environmental variables, threshold values and KPIs, which are relevant for this were also examined. Afterwards, various QoS management modules were presented based on the most important algorithms and their performance was evaluated using tests. Finally, the various modules were evaluated on the basis of the obtained results, their suitability classified and a holistic approach to the management of SLAs was also presented.

Chapter 6

Conclusion and Future Work

6.1 Achievements of the Research

Cloud Computing provides high potential in flexible on-demand usage of computing resources, including rapid deployment, rapid elasticity. This thesis addresses the problem of managing service execution in cloud infrastructures while meeting SLA constraints. Chapter 2 introduced the evolution of cloud computing during recent years and presented cloud deployment and service models, typical roles and related technologies. Thereby SLAs in general and cloud computing related SLAs, their life cycle, content and objectives have been introduced. An overview of the current cloud SLA landscape was given and advantages, as well as disadvantages, were discussed. To layout the basis of this research, Chapter 3 critical analysed the current situation of cloud computing SLA issues, as well as components needed to improve such systems. Chapter 4 introduced the novel autonomic cloud computing SLA management system "ASLAMaaS", which autonomously enforces SLA within specific constraints in a cloud infrastructure. During this research over 90 cloud-specific service KPIs have been documented and publicised. To the best knowledge of the author, no similar catalogue existed by the time of publication. SLA management functionalities are proposed which handle service instantiation, provisioning and termination in an autonomous fashion. In addition, the thesis employs further management actions and prediction algorithms in order to increase profits by reducing infrastructure costs and preventing SLA violations. The general SLA management process and its adoption to the cloud computing model have been presented. For this thesis, a special SLA creation process has been implemented and its SLO-A format has been presented, thus making it possible to write legally binding contracts in a machine-readable way. Allowing customers of cloud services to specify exactly which service level they booked for their cloud service and where, how and when to check it. Hereby the consumer is able to individually adjust the performance of his services and the resulting service levels using a predefined KPI catalogue, where the resulting SLA is directly monitored for compliance by autonomous management, and possible violations are prevented adaptively by resource control algorithms. This new kind of resource management and service management increases the trust and efficiency of the cloud environment and enables to build more trust into cloud environment services. Chapter 5 presents the developed prototype. Three development stages, representing the SLA frontend with its editor and A-SLO-A language, the QoS and SLA management modules with different prediction algorithms and the holistic SLA management system are described. The SLA compliance management algorithms presented and analysed here showed that the number of SLA violations could be reduced by already drastically by expanding the current methods of resource management in cloud environments. Furthermore, the SLA compliance became even better through the use of the latest machine learning approaches which could predict the environmental variables of the system and the usage by the user. These innovative approaches make it possible to make better use of existing services, to better integrate new services into the system and to reduce the reliability and therefore the costs for the provider. Furthermore, this thesis includes initial proposals for resource acquisition and allocation. Based on scheduling procedures, the required resources are used as efficiently as possible and additional resources are provided in the event of an SLA violation. In order to increase provider profit, this thesis has shown how packing algorithms can generate resource utilization close to potential optimum. Besides, it was considered that in a future version, contracts with lower priorities or penalties would be exchanged for contracts with high priorities/penalties as soon as the required resources can no longer be provided to keep the penalties as low as possible.

6.1.1 Key Contributions

In order to emphasize the achievements of this work, are here the main contributions listed below. Section 6.1 states the contributions in more detail and puts them in their context.

- Research and publication of the "Richtliniendokument", stating over 90 cloud-specific service KPIs, cloud SLA content and structure guidelines. [189] [1]

- The Autonomic Cloud Computing SLA Management architecture and its implementation.

- Graphical SLA frontend to facilitate ease of cloud SLA creation and monitoring.

- A-SLO-A an extended reference implementation of SLA* to create cloud dynamic and machine-readable SLAs.

- KPI QoS enforcement modules to improve SLA compliance based on prediction algorithms with the use of fuzzy logic, ANN and linear regression.

- The holistic SLA management approach with scheduling methods to minimize the financial impact of SLA violations.
- Conference papers and presentations. See Appendix A for a complete listing.

6.1.2 Limitations

Although the overall objectives of the research have been met, a number of decisions had to be made, which imposed limitations upon the work. The main limitations of the research are summarised below

- Service Level Agreements - In the here presented approach, the creation, operation and monitoring of cloud SLAs, was modelled after the common understanding and the requirements that kept the negotiation of SLAs unnecessary. The process of SLA creation was considered in such a way that there is no negotiation between provider and user. However, this is not necessarily true for all cloud providers. Depending on usage and general usability this part, for which there are already a lot of approaches in the literature, has to be integrated.
- A-SLO-A - Although A-SLO-A can be seen as a reference implementation of SLA* with minor domain-specific extensions, its application remains so far solely to this research. No other implementation, nor a vital user community exists. The agreements and corresponding QoS management configuration were prototypically implemented for the introduced use cases. However, for interchangeable cloud SLA an industry-wide adaptation would be needed. Furthermore, the domain-specific changes made must be accepted and applied by the respective cloud providers.
- QoS Management Modules- A number of practical limitations exist with the QoS management modules, the simulation environment and the cloud usage data. Since no real-life cloud provider data set was available the developed simulator creates a simplified cloud usage data set. Although these usage data sets were generated according to realistic scenarios, the question of the real-world applicability always remains. Additionally, due to insufficient time and resources, the integration of more input features for neural networks and additional machine learning algorithms could not be further assessed.

Despite these limitations, the research has made valid contributions to knowledge and provided sufficient proof of concept for the ideas proposed

6.2 Lessons Learned

While working on this thesis some interesting insights were gained via the experimental implementations. As shown by the fuzzy approach, even small improvements in resource allocation of cloud system can result in great improvements in service level compliance. Although the machine learning approaches can improve these results, it still remains questionable whether the extra effort justifies the result. Especially in the case of time-critical real-time systems, due to the fast reaction time of the system, the prediction must be viewed critically. During my extensive work with cloud systems, it has been shown that simpler, but easy-to-use solutions are particularly more often chosen in corporate environments, but this does not reduce the interest in cutting edge solutions. Especially in the cooperation during the research project and the industry, it was repeatedly shown how important an accurate and legally correct description of SLAs and KPIs is. Through the resulting catalogue of directives, we were able to return this knowledge back to practical application. The research on this work and the possibility of carrying out tests and measurements in a real environment and then working with the industry revealed several interesting insights. In fact, my work clearly showed that in contrast to developmental environments in the real world outages or SLA violations, things can happen much more frequently because of things that can not be steered directly through an automatism. Worse still, in today's complex multi-cloud environments, both internal and external factors play a big role on which one may not always be able to influence.

6.3 Future Work

Scalability The current implementation of the ASLAMaaS framework does not aim at scalability. Expanding the system with external cloud services such as Amazon EC2 or the Alibaba Cloud could enable the overall system to respond even more dynamically to the load of users. Both the acquisition of resources and the associated integration would have to be automated, as well as the dynamic management or precautionary planning of resources. For example, Amazon offers cheaper resources if they are booked in advance for a foreseeable period of time, which in turn can have a positive impact on costs. These extensions would extend the existing system by a further dimension in the planning and enable much greater dynamics. Simultaneously the move from one single resource provider and therefore only horizontal elasticity as means for improving performance, to a multi cloud environment with various elasticity in horizontal and vertical order as well as global distribution of resources. Exploiting this additional elasticity allows implementing further QoS assurance mechanisms. Regarding multi cloud environments, future work may consider checking prices and negotiating contracts with them to improve costs. In addition, this would allow overcoming resource shortages and the utilization of cheaper resources.

On-The-Fly Profiling The profiling mechanism implemented profiles the service provider before provisioning. However, profiling data during provisioning is useful for calibrating according to unpredicted environment changes, e.g., network throughput degradation and unusual customer demands. An improvement to this issue is enabling to update the provider profiling while treating customer requests.

Renegotiation The SLA contracts considered in this work can be more flexible by including renegotiation (alteration) of established contracts. Supporting renegotiation would allow adapting the service provisioning to environment changes without violating SLAs.

Improved machine learning algorithms The field of machine learning has grown in the last time very rapidly. This may result in new technologies that improve the use of machine learning algorithms and make them more efficient. The expansion of the ML approaches could lead to future work being able to generate on the fly predictions about the state and the future use of the system not only more accurately but also much faster. As a result, the possibility of countering possible SLA violations already arises and thus achieving even higher SLA compliance. Especially the area tensor flow could be interesting here.

Prove of Concept for Holistic SLA Management Since the approach presented here for the holistic management of SLAs was presented purely computationally and through simulations, a prototypical implementation in a real cloud landscape would be desirable. Unfortunately, in order to get a realistic experience, this is a challenge for both the infrastructure and the test user.

References

[1] HFU Cloud Research Lab, "Autonomic SLA Management as a Service Project Website," 2014.
http://wolke.hs-furtwangen.de/currentprojects/aslamaas [Accessed: 25. 07. 2014].

[2] F. Liu, J. Tong, J. Mao, R. B. Bohn, J. V. Messina, M. L. Badger, and D. M. Leaf, *NIST Cloud Computing Reference Architecture*.
National Institute of Standards and Technology, 2011.
http://www.nist.gov/manuscript-publication-search.cfm?=909505 [Accessed: 24. 06. 2014].

[3] Gartner, "Worldwide Public Cloud Revenue to Grow 17.3 Percent in 2019," 2018.
https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019 [Accessed: 02. 05. 2019].

[4] Gartner Research Group, "Hype Cycle," 2014.
http://www.gartner.com/technology/research/hype-cycles [Accessed: 25. 08. 2014].

[5] Gartner Research Group, "Hype Cycle for Cloud Computing," 2018.
https://www.gartner.com/en/documents/3884671/hype-cycle-for-cloud-computing-2018, [Accessed: 15. 05. 2019].

[6] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[7] E. Manoel, M. J. Nielsen, A. Salahshour, S. Sampath, and S. Sudarshanan, *Problem Determination Using Self-Managing Autonomic Technology*.
IBM Redbooks, 2005.
ISBN 978-0738-4911-10.

[8] J. Butler, J. Lambea, M. Nolan, W. Theilmann, F. Torelli, R. Yahyapour, A. Chiasera, and M. Pistore, "Slas empowering services in the future internet," in *The Future Internet Assembly*, pp. 327–338, Springer, 2011.

[9] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*.
O'Reilly Media, 2009.

ISBN-978-0-5968-0276-9.

[10] M. Peter and G. Timothy, *The NIST Definition of Cloud Computing,*.
National Institute of Standards and Technology, 2011.
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [Accessed: 24. 04. 2014].

[11] BITKOM and KPMG, "Cloud Monitor 2019," 2019.
https://www.bitkom.org/sites/default/files/2019-06/bitkom_kpmg_pk_charts_cloud_monitor_18_06_2019.
[Accessed: 18. 07. 2019].

[12] Dropbox Inc., "Dropbox Website," 2014.
https://www.dropbox.com [Accessed: 12. 09. 2014].

[13] Microsoft, " OneDrive Website," 2014.
https://onedrive.live.com [Accessed: 12. 09. 2014].

[14] Google, "Google Drive Website," 2014.
https://drive.google.com [Accessed: 12. 09. 2014].

[15] Amazon, "AWS Web Services Website," 2014.
http://aws.amazon.com/, [Accessed: 12. 09. 2014].

[16] Microsoft, "Azure Cloud Platform Website," 2014.
https://azure.microsoft.com [Accessed: 12. 09. 2014].

[17] G. Cattaneo, M. Kolding, D. Bradshaw, G. Folco, IDC for the European Commission, "Quantitative estimates of the demand for cloud computing in europe and the likely barriers to take-up."
http://cordis.europa.eu/fp7/ict/ssai/docs/study45-d2-interim-report.pdf [Accessed: 12. 08. 2014].

[18] Distributed Management Task Force, "Architecture for Managing Clouds," 2013.
http://dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf [Accessed: 13. 07. 2014].

[19] ISO/IEC SC 38 Study Group, "JTC SC 38 Study Group Report on Cloud Computing."
http://isotc.iso.org [Accessed: 17. 06. 2014].

[20] Amazon, "EC2 Service Level Agreement Website."
http://aws.amazon.com/de/ec2/sla, [Accessed: 02. 05. 2015].

[21] Dimension Data, "Comparing Public Cloud Service Level Agreements," 2013.
http://cloud.dimensiondata.com/sites/default/files/comparing_public_cloud_service_level_agreements_0_0.pdf [Accessed: 02. 05. 2015].

[22] S. A. Baset, “Cloud SLAs: Present and Future,” *SIGOPS Oper. Syst. Rev.*, vol. 46, pp. 57–66, July 2012.

[23] Gartner Research, “Cloud iaas slas can be meaningless,” 2012.
http://blogs.gartner.com/lydia_leong/2012/12/05/cloud-iaas-slas-can-be-meaningless [Accessed: 12. 09. 2014].

[24] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck, “Web Service Level Agreement (WSLA) Language Specification, v1.0,” Jan. 2003.
http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf [Accessed: 03. 05. 2010].

[25] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, “QoS Aware Clouds,” *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 321–328, July 2010.

[26] P. Wieder, J. M. Butler, W. Theilmann, and R. Yahyapour, *Service Level Agreements for Cloud Computing*.
Springer Science & Business Media, 2011.
ISBN 978-1-4614-1614-2.

[27] N. Leavitt, “Is Cloud Computing Really Ready for Prime Time?,” *Computer*, vol. 42, pp. 15–20, Jan 2009.

[28] BITKOM, “Cloud monitor 2014,” 2014.
http://www.bitkom.org/de/presse/30739_78524.aspx [Accessed: 18. 08. 2014].

[29] Juniper Research, “Cloud Computing - Consumer Markets - Strategies & Forecasts 2014-2018,” 2014.
http://www.juniperresearch.com/reports.php?id=739 [Accessed: 12. 08. 2014].

[30] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[31] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande, “Folding@ home and genome@ home: Using distributed computing to tackle previously intractable problems in computational biology,” *arXiv preprint arXiv:0901.0866*, 2009.

[32] C. Adam-Bourdarios, D. Cameron, A. Filipčič, E. Lancon, W. Wu, *et al.*, “Atlas@ home: harnessing volunteer computing for hep,” in *Journal of Physics: Conference Series*, vol. 664, p. 022009, IOP Publishing, 2015.

[33] B. P. Abbott, R. Abbott, R. Adhikari, P. Ajith, B. Allen, G. Allen, R. Amin, S. Anderson, W. Anderson, M. Arain, *et al.*, “Einstein@ home search for periodic gravitational waves in early s5 ligo data,” *Physical review d*, vol. 80, no. 4, p. 042003, 2009.

[34] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared,” *Grid Computing Environments Workshop, 2008. GCE ’08*, pp. 1–10, Nov. 2008.

[35] N. Chawla and D. Kumar, “Desktop virtualization as a service and formulation of tco with return on investment,” in *Software Engineering*, pp. 599–608, Springer, 2019.

[36] S. Srinivasan, *Cloud Computing Basics*.
ISBN 978-1-4614-7698-6: Springer New York, 1st ed., 2014.

[37] H. M. Elmasry, A. E. Khedr, and M. M. Nasr, “An adaptive technique for cost reduction in cloud data centre environment,” *International Journal of Grid and Utility Computing*, vol. 10, no. 5, pp. 448–464, 2019.

[38] Y.-C. Lee, “Adoption intention of cloud computing at the firm level,” *Journal of Computer Information Systems*, vol. 59, no. 1, pp. 61–72, 2019.

[39] S. Mishra, N. Tripathy, B. K. Mishra, and C. Mahanty, “Analysis of security issues in cloud environment,” *Security Designs for the Cloud, Iot, and Social Networking*, pp. 19–41, 2019.

[40] Intel Corp., “What is holding back the cloud?,” 2012.
http://www.intel.com/content/www/us/en/cloud- computing/whats-holding-back-the-cloud-peer-research-report.html, [Accessed: 02. 08. 2014].

[41] A. Behl and K. Behl, “An analysis of cloud computing security issues,” *Information and Communication Technologies (WICT), 2012 World Congress on*, pp. 109–114, Oct. 2012.

[42] D. Zissis and D. Lekkas, “Addressing cloud computing security issues,” *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[43] A. Garg and R. Rathi, “A survey on cloud computing risks and remedies,” *International Journal of Computer Applications*, vol. 178, no. 29, pp. 35–37, 2019.

[44] Amazon, “AWS Simple Monthly Calculator Website.”
https://calculator.aws/, [Accessed: 14. 02. 2019].

[45] A2 Hosting, “Dedicated Server Hosting,” 2019.
https://www.a2hosting.com/dedicated-server-hosting/core/compare [Accessed: 12. 07. 2019.

[46] S. Krishnan and L. Chen, “Legal concerns and challenges in cloud computing,” *arXiv preprint arXiv:1905.10868*, 2019.

[47] ITIL Foundation, “Information Technology Infrastructure Library (ITIL).”
http://www.itil-officialsite.com [Accessed: 12. 09. 2014].

[48] H. Schmidt, *Entwurf von Service Level Agreements auf der Basis von Dienstprozessen*.
Herbert Utz München, 2001.
ISBN: 3-8316-0071-6.

[49] W. Sun, Y. Xu, and F. Liu, "The role of XML in service level agreements management," *2005 International Conference on Services Systems and Services Management, 2005. proceedings of ICSSSM '05.*, vol. 2, pp. 1118–1120, 2005.

[50] P. Hasselmeyer, B. Koller, I. Kotsiopoulos, D. Kuo, and M. Parkin, "Negotiating SLAs with dynamic pricing policies," *Proceedings of the SOC@ Inside'07*, 2007.

[51] G. R. Gangadharan, G. Frankova, and V. D'Andrea, "Service license life cycle," *Collaborative Technologies and Systems, 2007. CTS 2007. International Symposium on*, pp. 150–158, 2007.

[52] R. Scholderer, *Management von Service-Level-Agreements: methodische Grundlagen und Praxislösungen mit CobiT, ISO 20000 und ITIL*.
Dpunkt.Verlag GmbH, 2011.
ISBN 978-3-8986-4702-1.

[53] T. G. Berger, *Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen*.
Dissertation, TU Darmstadt, Germany, 2005.

[54] R. Sturm, W. Morris, and M. Jander, *Foundations of Service Level Management*.
Sams Professionals, Pearson Sams, 2000.
ISBN: 978-0-6723-1743-9.

[55] Vanson Bourne Technology Market Research, "Companies struggling with cloud performance."
http://servicemanagement.cbronline.com/news/cloud-performance-issues-costing-firms-600000-a-year-survey [Accessed: 12. 11. 2013].

[56] ISO/IEC JTC 1/SC 38 Study Group, "Study Group Report on Cloud Computing," tech. rep., International Organization for Standardization, 2011.
http://isotc.iso.org/livelink/livelink/fetch/-8913189/8913214/8913373/Study_Group_on_Cloud_Computing_final_report.pdf [Accessed: 08. 09. 2014].

[57] K. T. Kearney, F. Torelli, and C. Kotsokalis, "SLA*: An Abstract Syntax for Service Level Agreements," *11th IEEE/ACM International Conference on Grid Computing*, pp. 217–224, 2011.

[58] SLA@SOI, "SLA@SOI Project Website."
http://sla-at-soi.eu/ [Accessed: 17. 09. 2016].

[59] P. Chronz and P. Wieder, “Integrating ws-agreement with a framework for service-oriented infrastructures,” in *2010 11th IEEE/ACM International Conference on Grid Computing (GRID 2010)*, (Los Alamitos, CA, USA), pp. 225–232, IEEE Computer Society, oct 2010.

[60] Service Now, “KPI Library,” 2013.
http://mirror42.com [Accessed: 12. 07. 2014].

[61] International Organization for Standardization, “ISO/IEC NP 19086-2 Information technology – Cloud computing – Service level agreement (SLA) framework and Technology – Part 2: Metrics,” 2014.
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=67546 [Accessed: 22. 09. 2014].

[62] Experton Group, “Experton-analyse: Wer f/"ullt die cloud mit business services.”
http://www.computerwoche.de/management/cloud- computing/1934027/ [Accessed: 17. 03. 2014].

[63] A. Keller and H. Ludwig, “The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services,” *Journal of Network and Systems Management*, vol. 11, pp. 57–81, Mar. 2003.

[64] M. Perry and H. Kaminski, *SLA Negotiation System Design Based on Business Rules*.
08 2008.

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

[66] S. Venticinque, R. Aversa, B. Di Martino, M. Rak, and D. Petcu, “A cloud agency for sla negotiation and management,” in *Euro-Par 2010 Parallel Processing Workshops* (M. R. Guarracino, F. Vivien, J. L. Träff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knüpfer, B. Di Martino, and M. Alexander, eds.), (Berlin, Heidelberg), pp. 587–594, Springer Berlin Heidelberg, 2011.

[67] Rackspace Ltd - Compare Service Levels, 2015.
Website, http://www.rackspace.co.uk/managed-cloud/compare-service-levels, [Accessed: 25. 08. 2015].

[68] ASLAMaaS Project, “Autonomic SLA Management as a Service Project - SLA Richtliniendokument für Cloud Dienstleistungen,” 2014.
http://www.wolke.hs-furtwangen.de/assets/files/ASLAMaaS-SLA-Richtliniendokument.pdf [Accessed: 25. 07. 2014].

[69] D. Gunter, B. Tierney, B. Crowley, M. Holding, and J. Lee, "Netlogger: a toolkit for distributed system performance analysis," *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No.PR00728)*, pp. 267–273, Aug 2000.

[70] W. Fu and Q. Huang, "Grideye: A service-oriented grid monitoring system with improved forecasting algorithm," *proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops*, pp. 5–12, 2006.

[71] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Comput. Netw.*, vol. 53, pp. 2923–2938, Dec. 2009.

[72] J. S. Ward and A. Barker, "Observing the clouds: a survey and taxonomy of cloud monitoring," *Journal of Cloud Computing*, vol. 3, no. 1, p. 24, 2014.

[73] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of cloud monitoring tools: Taxonomy, capabilities and objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, 2014.

[74] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[75] M. Saqib, "Cloud monitoring: A survey.," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 2, 2017.

[76] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v3.0," 2011.
https://downloads.cloudsecurityalliance.org/assets/research/security-guidance/csaguide.v3.0.pdf [Accessed: 25. 07. 2016].

[77] J. Spring, "Monitoring cloud computing by layer, part 1," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 66–68, 2011.

[78] J. Spring, "Monitoring cloud computing by layer, part 2," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 52–55, 2011.

[79] D. A. Menascé, "Load testing, benchmarking, and application performance management for the web," *Int. CMG Conference*, pp. 271–282, 2002.

[80] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," *4th IEEE International Conference on Digital Ecosystems and Technologies*, pp. 606–610, Apr. 2010.

[81] J. Spillner, M. Winkler, S. Reichert, J. Cardoso, and A. Schill, “Distributed contracting and monitoring in the internet of services,” *proceedings of the 9th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, pp. 129–142, 2009.

[82] P. Patel, A. Ranabahu, and A. Sheth, *Service Level Agreement in Cloud Computing*. The Ohio Center of Excellence in Knowledge-Enabled Computing, KNO.E.SIS PUBLICATIONS, 2009.

[83] R. Kübert, G. Katsaros, and T. Wang, “A RESTful Implementation of the WS-agreement Specification,” *proceedings of the Second International Workshop on RESTful Design*, pp. 67–72, 2011.

[84] C. Müller, M. Oriol, M. Rodríguez, X. Franch, J. Marco, M. Resinas, and A. Ruiz-Cortés, “SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents,” *2012 4th International Workshop on Principles of Engineering Service-Oriented Systems (PESOS)*, pp. 43–49, June 2012.

[85] A. Kertesz, G. Kecskemeti, and I. Brandic, “Autonomic sla-aware service virtualization for distributed systems,” *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pp. 503–510, Feb. 2011.

[86] A. Undheim, A. Chilwan, and P. Heegaard, “Differentiated availability in cloud computing slas,” *2011 IEEE/ACM 12th International Conference on Grid Computing*, pp. 129–136, Sept. 2011.

[87] F. Liu, J. Tong, J. Mao, R. B. Bohn, J. V. Messina, M. L. Badger, and D. M. Leaf, “NIST Cloud Computing Reference Architecture,” 2011.
http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505 [Accessed: 03. 05. 2013].

[88] B. Khasnabish, J. Chu, S. Ma, Y. Meng, N. So, and P. Unbehagen, “Cloud reference framework,” tech. rep., Internet Engineering Task Force, 2010.

[89] Y. Kouki, F. A. De Oliveira, S. Dupont, and T. Ledoux, “A language support for cloud elasticity management,” *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 206–215, 2014.

[90] FOSII Team, “Foundations of self-governing ict infrastructures website.”
http://www.infosys.tuwien.ac.at/linksites/FOSII/index.html [Accessed: 10. 07. 2014].

[91] I. Brandic, D. Music, P. Leitner, and S. Dustdar, “Vieslaf framework: Enabling adaptive and versatile sla-management,” *proceedings of the 6th International Workshop on Grid Economics and Business Models*, pp. 60–73, 2009.

[92] Fraunhofer IAO, “Theseus Texo - Business Webs im Internet der Dienste,” 2011. https://www.e-business.iao.fraunhofer.de/de/projekte/beschreibung/theseus-texo.html [Accessed: 25. 07. 2015].

[93] A. Lawrence, K. Djemame, O. Wäldrich, W. Ziegler, and C. Zsigri, “Using service level agreements for optimising cloud infrastructure services,” *proceedings of the 2010 international conference on Towards a service-based internet*, pp. 38–49, 2011.

[94] SLA@SOI, “SLA Model Website.” http://sourceforge.net/apps/trac/sla-at-soi/wiki/SLA[Accessed: 17. 02. 2017].

[95] SLA@SOI, “Framework Alpha Website.” http://sourceforge.net/projects/sla-at-soi/files/ [Accessed: 17. 02. 2017].

[96] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, “Slac: A formal service-level-agreement language for cloud computing,” *proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 419–426, 2014.

[97] Project Website, “SLA-Ready,” 2017. https://www.sla-ready.eu/sla-negotiation [Accessed: 25. 08. 2017].

[98] A. Lapedes and R. Farber, “Nonlinear Signal Processing Using Neural Networks: Prediction and System Modelling,” Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.

[99] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[100] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[101] G. P. Zhang, “An investigation of neural networks for linear time-series forecasting,” *Computers and Operations Research*, vol. 28, no. 12, pp. 1183–1202, 2001.

[102] G. Zhang, B. E. Patuwo, and M. Y. Hu, “Forecasting with artificial neural networks: The state of the art,” *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[103] M. Adya and F. Collopy, “How effective are neural networks at forecasting and prediction? a review and evaluation,” *Journal of Forecasting - Special Issue: Neural Networks and Financial Economics*, vol. 17, pp. 481–495, Sept. 1998.

[104] G. Langella, A. Basile, A. Bonfante, and F. Terribile, “High-resolution space–time rainfall analysis using integrated {ANN} inference systems,” *Journal of Hydrology*, vol. 387, no. 3–4, pp. 328–342, 2010.

[105] J. Taylor and R. Buizza, “Neural network load forecasting with weather ensemble predictions,” *Power Systems, IEEE Transactions on*, vol. 17, pp. 626–632, Aug. 2002.

[106] P. J. Roebber, M. R. Butt, S. J. Reinke, and T. J. Grafenauer, “Real-time forecasting of snowfall using a neural network,” *Weather and Forecasting*, vol. 22, pp. 676–684, 2014/07/15 2007.

[107] D. C. Pattie and J. Snyder, “Using a neural network to forecast visitor behavior,” *Annals of Tourism Research*, vol. 23, no. 1, pp. 151–164, 1996.

[108] D. Park, M. El-Sharkawi, I. Marks, R.J., L. Atlas, and M. Damborg, “Electric load forecasting using an artificial neural network,” *Power Systems, IEEE Transactions on*, vol. 6, pp. 442–449, May 1991.

[109] H. Hippert, C. Pedreira, and R. Souza, “Neural networks for short-term load forecasting: a review and evaluation,” *Power Systems, IEEE Transactions on*, vol. 16, pp. 44–55, Feb. 2001.

[110] Y. Bodyanskiy and S. Popov, “Neural network approach to forecasting of quasiperiodic financial time series,” *European Journal of Operational Research*, vol. 175, no. 3, pp. 1357–1366, 2006.

[111] P. McAdam and P. McNelis, “Forecasting inflation with thick models and neural networks,” *Economic Modelling*, vol. 22, no. 5, pp. 848–867, 2005.

[112] I. Kaastra and M. Boyd, “Designing a neural network for forecasting financial and economic time series,” *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.
Financial Applications, Part II.

[113] E. Guresen, G. Kayakutlu, and T. U. Daim, “Using artificial neural network models in stock market index prediction,” *Expert Systems with Applications*, vol. 38, no. 8, pp. 10389–10397, 2011.

[114] A. M. Vukicevic, G. R. Jovicic, M. M. Stojadinovic, R. I. Prelevic, and N. D. Filipovic, “Evolutionary assembled neural networks for making medical decisions with minimal regret: Application for predicting advanced bladder cancer outcome,” *Expert Systems with Applications*, 2014.

[115] C. Arizmendi, D. A. Sierra, A. Vellido, and E. Romero, “Automated classification of brain tumours from short echo time in vivo mrs data using gaussian decomposition and bayesian neural networks,” *Expert Systems with Applications*, vol. 41, no. 11, pp. 5296–5307, 2014.

[116] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, “Prediction of cloud data center networks loads using stochastic and neural models,” *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pp. 276–281, 2011.

[117] C.-C. Li and K. Wang, “An sla-aware load balancing scheme for cloud datacenters,” *Information Networking (ICOIN), 2014 International Conference on*, pp. 58–63, 2014.

[118] R. Hu, J. Jiang, G. Liu, and L. Wang, “Kswsvr: A new load forecasting method for efficient resources provisioning in cloud,” *Services Computing (SCC), 2013 IEEE International Conference on*, pp. 120–127, 2013.

[119] A. Bankole and S. Ajila, “Predicting cloud resource provisioning using machine learning techniques,” *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*, pp. 1–4, 2013.

[120] M. T. Imam, S. F. Miskhat, R. M. Rahman, and M. A. Amin, “Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources,” *Computer and Information Technology (ICCIT), 2011 14th International Conference on*, pp. 333–338, 2011.

[121] M. Mao and M. Humphrey, “Auto-scaling to minimize cost and meet application deadlines in cloud workflows,” *SC ’11: proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, Nov. 2011.

[122] S. K. Panda, I. Gupta, and P. K. Jana, “Allocation-aware task scheduling for heterogeneous multi-cloud systems,” *Procedia Computer Science*, vol. 50, pp. 176–184, 2015.
Big Data, Cloud and Computing Challenges.

[123] Umale, J. and Mahajan, S., “Optimized grid scheduling using two level decision algorithm (tlda),” *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on , vol., no., pp.78-82, 28-30*, 2010.

[124] Kun-Ming Yu and Cheng-Kwan Chen, “An evolution-based dynamic scheduling algorithm in grid computing environment,” *Intelligent Systems Design and Applications, 2008. ISDA ’08. Eighth International Conference on , vol.1, no., pp.450-455, 26-28*, 2008.

[125] Song, S. and Kai Hwang and Yu-Kwong Kwok, “Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling,” *Computers, IEEE Transactions on , vol.55, no.6, pp.703-719*, 2006.

[126] Wang Meihong and Zeng Wenhua, “A comparison of four popular heuristics for task scheduling problem in computational grid,” *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on , vol., no., pp.1-4, 23-25*, 2010.

[127] Yujia Ge and Guiyi Wei, “Ga-based task scheduler for the cloud computing systems,” *Web Information Systems and Mining (WISM), 2010 International Conference on , vol.2, no., pp.181-186, 23-24*, 2010.

[128] Khalid, O.; Maljevic, I.; Anthony, R.; Petridis, M.; Parrott, K.; Schulz, M., “Deadline aware virtual machine scheduler for grid and cloud computing,” *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on , pp.85-90,*, 2010.

[129] Kailasam, S. and Gnanasambandam, N. and Dharanipragada, J. and Sharma, N., “Optimizing service level agreements for autonomic cloud bursting schedulers,” *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on , pp.285-294, 13-16*, 2010.

[130] Seoyoung Kim and Yoonhee Kim and Naeyoung Song andChongam Kim, “Adaptable scheduling schemes for scientific applications on science cloud,” *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on , pp.1-3, 20-24*, 2010.

[131] Haizea Project - University of Chicago, “Haizea - an open source vm-based lease manager.” http://haizea.cs.uchicago.edu/, 2011.
[Accessed: 25. 08. 2016].

[132] Jeyarani, R. and Ram, R. Vasanth and Nagaveni, N., “Design and implementation of an efficient two-level scheduler for cloud computing environment,” *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on , pp.585-586, 17-20*, 2010.

[133] Sotomayor, B., Keahey, K.: Haizea, “Haizea website,” 2011.
http://haizea.cs.uchicago.edu/ [Accessed: 25. 01. 2014].

[134] Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I., “Capacity leasing in cloudsystems using the opennebula engine,” *Cloud Computing and Applications, CCA2008*, 2009.

[135] Sotomayor, B., Foster, I., Keahey, K., “Overhead matters: A model for virtual re-source management,” *proceedings of the 2nd International Workshop on Virtu-alization Technology in Distributed Computing, IEEE Computer Society, Wash-ington, DC, USA*, 2006.

[136] Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I., “Resource leasing and the art of suspending virtual machines,” *In 11th IEEE International Conference on High Performance Computing and Communications, pp. 59–68. IEEE*, 2009.

[137] M. Boniface, S. C. Phillips, A. Sanchez-Macian, and M. Surridge, “Dynamic service provisioning using gria slas,” *International Conference on Service-Oriented Computing*, pp. 56–67, 2007.

[138] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, “Establishing and monitoring slas in complex service based systems,” *2009 IEEE International Conference on Web Services*, pp. 783–790, July 2009.

[139] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, “Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments,” *2010 International Conference on High Performance Computing & Simulation*, pp. 48–54, 2010.

[140] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, “Cloud resource provisioning and SLA enforcement via LoM2HiS framework,” *Concurrency and Computation: Practice and Experience*, vol. 25, 7 2013.

[141] G. Cicotti, L. Coppolino, S. D, N. Antonio, and L. Romano, “How to monitor qos in cloud infrastructures: The qosmonaas approach,” *International Journal of Computational Science and Engineering*, vol. 11, p. 29, 1 2015.

[142] V. Emeakaroha, M. Netto, R. Calheiros, I. Brandic, R. Buyya, and C. DeRose, “Towards autonomic detection of sla violations in cloud infrastructures,” *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1017–1029, 2012.
Special section: Quality of Service in Grid and Cloud Computing.

[143] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, “Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments,” *2010 International Conference on High Performance Computing Simulation*, pp. 48–54, June 2010.

[144] P. Varalakshmi, K. H. Priya, J. Pradeepa, and V. Perumal, *SLA with Dual Party Beneficiality in Distributed Cloud*.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
ISBN 978-3-642-22709-7.

[145] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, “A dependency-aware ontology-based approach for deploying service level agreement monitoring services in cloud,” *Softw. Pract. Exper.*, vol. 42, pp. 501–518, Apr. 2012.

[146] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, “Workflow scheduling for saas / paas cloud providers considering two sla levels,” *2012 IEEE Network Operations and Management Symposium*, pp. 906–912, Apr. 2012.

[147] H. Liu, D. Xu, and H. K. Miao, “Ant colony optimization based service flow scheduling with various qos requirements in cloud computing,” *2011 First ACIS International Symposium on Software and Network Engineering*, pp. 53–58, Dec. 2011.

[148] X. Wang, Z. Du, and Y. Chen, “An adaptive model-free resource and power management approach for multi-tier cloud environments,” *Journal of Systems and Software*, vol. 85, no. 5, pp. 1135–1146, 2012.

[149] V. Casola, A. De Benedictis, M. Eraşcu, J. Modic, and M. Rak, “Automatically enforcing security slas in the cloud,” *IEEE Transactions on Services Computing*, vol. 10, pp. 741–755, Sept. 2017.

[150] P. Horn, *Autonomic computing: IBMs Perspective on the State of Information Technology*.
IBM Press, 2001.

[151] W. Fei and L. Fan-Zhang, “The design of an autonomic computing model and the algorithm for decision-making,” *Granular Computing, 2005 IEEE International Conference on*, vol. 1, pp. 270–273, 2005.

[152] Z. Zhao, C. Gao, and F. Duan, “A survey on autonomic computing research,” *Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on*, vol. 2, pp. 288–291, 2009.

[153] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, “Resource provisioning for cloud computing,” *proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, CASCON ’09*, pp. 101–111, 1 2009.

[154] E. Casalicchio, D. A. Menascé, and A. Aldhalaan, “Autonomic resource provisioning in cloud systems with availability goals,” *proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 1:1–1:10, 2013.

[155] T. M. King and A. S. Ganti, “Migrating autonomic self-testing to the cloud,” *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, pp. 438–443, Apr. 2010.

[156] I. Brandic, “Towards self-manageable cloud services,” *2009 33rd Annual IEEE International Computer Software and Applications Conference*, vol. 2, pp. 128–133, 2009.

[157] R. Sterritt and M. Hinchey, “Autonomic computing-panacea or poppycock?,” *Engineering of Computer-Based Systems, 2005. ECBS’05. 12th IEEE International Conference and Workshops on the*, pp. 535–539, 2005.

[158] R. Murch, *Autonomic computing*.
IBM Press, 2004.

[159] J. Buisson, F. André, and J.-L. Pazat, “Dynamic adaptation for grid computing,” *Advances in Grid Computing-EGC 2005*, pp. 139–168, 2005.

[160] J. Buisson, *Adaptation dynamique de programmes et composants parallèles*.
PhD thesis, Thèse de doctorat, INSA de Rennes, 2006.

[161] F. J. Furrer, “Architecture principles for resilience,” pp. 309–336, 2019.

[162] S. Frey, C. Lüthje, R. Teckelmann, and C. Reich, “Adaptable Service Level Objective Agreement (A-SLO-A) for Cloud Services,” *CLOSER 2013, pp. 457-462, SciTePress*, 2013.
ISBN 978-989-8565-52-5.

[163] A. Patcha and J.-M. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer Networks, Volume 51, Number 21*, pp. 3448–3470, 2007.

[164] S. Frey, C. Lüthje, V. Huwwa, and C. Reich, “Fuzzy controled qos for scalable cloud computing services,” *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 150–155, 2013.

[165] Ganglia, “Ganglia Website.”
https://ganglia.info/ [Accessed: 25. 01. 2018].

[166] R. Teckelmann, “Service level objective agreements format zur adaption von dynamischen slas in cloudia,” Master’s thesis, Hochschule Furtwangen University, 2012.

[167] K. Tanaka, *An Introduction to Fuzzy Logic for Practical Applications*.
Springer, 1997.

[168] G. Chen, T. T. Pham, and N. Boustany, “Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems,” *Applied Mechanics Reviews*, vol. 54, pp. B102–B103, 11 2001.

[169] L. A. Zadeh, “Is there a need for fuzzy logic?,” *Information Sciences*, vol. 178, no. 13, pp. 2751 – 2779, 2008.

[170] B. Andrew and B. Peter, *Practical Statistics for Data Scientists - 50 Essential Concepts*.
Sebastopol: "O’Reilly Media, Inc.", 2017.
ISBN 978-1-491-95293-1.

[171] S. Weisberg, *Applied Linear Regression*.
Wiley Series in Probability and Statistics, Wiley, 2014.
ISBN 9781118789551.

[172] OpenStack, "Openstack official website."
https://www.openstack.org/ [Accessed: 17. 01. 2018].

[173] T. Jetter, "Membrain neuronale netze editor simulator website."
https://membrain-nn.de [Accessed: 17. 02. 2018].

[174] A. Popko, M. Jakubowski, and R. Wawer, "Membrain neural network for visual pattern recognition," *Advances in Science and Technology Research Journal*, vol. Vol. 7, nr 18, pp. 54–59, 2013.

[175] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*.
Wiley Series in Probability and Statistics, Wiley, 2013.
ISBN 9781118627365.

[176] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[177] V. N. Vapnik, *The Nature of Statistical Learning Theory*.
Berlin, Heidelberg: Springer-Verlag, 1995.
ISBN 0-387-94559-8.

[178] Rapidminer, "Rapidminer Website."
https://rapidminer.com/ [Accessed: 25. 01. 2016].

[179] C.-C. Li and K. Wang, "An sla-aware load balancing scheme for cloud datacenters," *The International Conference on Information Networking 2014 (ICOIN2014)*, pp. 58–63, Feb. 2014.

[180] J. Prevost, K. Nagothu, B. T. Kelley, and M. M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," *2011 6th International Conference on System of Systems Engineering*, pp. 276–281, 2011.

[181] R. Hu, J. Jiang, G. Liu, and L. Wang, "Kswsvr: A new load forecasting method for efficient resources provisioning in cloud," *2013 IEEE International Conference on Services Computing*, pp. 120–127, June 2013.

[182] A. Y. Nikravesh, S. A. Ajila, and C. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 35–45, May 2015.

[183] A. A. Bankole and S. A. Ajila, "Predicting cloud resource provisioning using machine learning techniques," *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, May 2013.

[184] B. Korte and J. Vygen, *Bin-Packing - Kombinatorische Optimierung: Theorie und Algorithmen*.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
pp 485–502, ISBN 978-3-540-76919-4.

[185] R. Sedgewick, "ALGORITHMS, Addison-Wesley Publishing Company, 1983. No. of pages: 552.," *Software: Practice and Experience*, vol. 14, pp. 501–502, May 1984.

[186] L. Epstein and A. Levin, "Minimum weighted sum bin packing," *Approximation and Online Algorithms*, pp. 218–231, 2008.
ISBN 978-3-540-77918-6.

[187] B. Patt-Shamir and D. Rawitz, *Vector Bin Packing with Multiple-Choice*.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
ISBN 978-3-642-13731-0.

[188] G. R. Raidl, "The multiple container packing problem: A genetic algorithm approach with weighted codings," *SIGAPP Appl. Comput. Rev.*, vol. 7, pp. 22–31, Mar. 1999.

[189] F. Stefan, L. Claudia, R. Christoph, Z. Michael, Maierand Tobias, and S. Markus, "Sla-richtliniendokument für cloud dienstleistungen."
http://www.wolke.hs-furtwangen.de/assets/files/ASLAMaaS-SLA-Richtliniendokument.pdf [Accessed: 17. 01. 2015].

[190] N. G. Carr, *The Big Switch: Der grosse Wandel - Die Vernetzung der Welt von Edison bis Google*.
ISBN 978-3-8266-5508-1: Heidelberg: Mitp, 1st ed., 2009.

[191] Amazon, "Amazon Elastic Compute Cloud (EC2)," 2006.
http://aws.amazon.com/de/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2—beta [Accessed: 22. 06. 2014].

[192] A. Regalado, *Who Coined 'Cloud Computing'?*
MIT Technology Review Business in the Cloud, 2011.
http://www.technologyreview.com/news/425970/who-coined-cloud-computing [Accessed: 24. 06. 2014].

[193] S. E. Gillett and M. Kapor, *Coordinating the Internet*.
The Self-Governing Internet: Coordination by Design, MIT Press Cambridge, USA, 1997.
pp. 3 – 38, ISBN 0-262-61136-8.

[194] R. K. Chellappa, *Intermediaries in cloud-computing: A new computing paradigm*.
INFORMS Dallas 1997, Cluster: Electronic Commerce, Dallas, Texas, 1997.

[195] M. Cooter and C. Pro, *The Ultimate Guide to Cloud Computing*.
Dennis Publishing, 2011.
ISBN 978-1781-0609-26.

[196] IBM, *The Virtualization Cookbook for IBM z/VM 6.3, RHEL 6.4, and SLES 11 SP3*.
IBM Redbooks, 2014.
ISBN 978-0-7384-3893-1.

[197] Rackspace, "Rackspace website," 2014.
http://www.rackspace.com [Accessed: 12. 09. 2014].

[198] Google, "Google Compute Website," 2014.
https://cloud.google.com/compute [Accessed: 12. 09. 2014].

[199] Google, "Google App Engine Website," 2014.
https://cloud.google.com/appengine/docs [Accessed: 12. 09. 2014].

[200] Google, "Google Apps Website," 2014.
http://www.google.com/work/apps/business [Accessed: 12. 09. 2014].

[201] Google, "Google Docs Website," 2014.
https://docs.google.com [Accessed: 12. 09. 2014].

[202] Microsoft, "Office 365 Website," 2014.
http://office.microsoft.com/de-de [Accessed: 12. 09. 2014].

[203] Salesforce, 2014.
http://www.salesforce.com [Accessed: 12. 09. 2014].

Appendix A - Published Papers & Outputs

Conference Papers

1. Stefan Frey, Claudia Lüthje, Christoph Reich, "**Key Performance Indicators for Cloud Computing SLAs**" EMERGING 2013 - The Fifth International Conference on Emerging Network Intelligence, October 2013, ISBN: 978-1-61208-292-9

2. Stefan Frey, Claudia Lüthje, Vitali Huwwa, Christoph Reich, "**Fuzzy Controlled QoS for Scalable Cloud Computing Services**" CLOUD COMPUTING 2013 : The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, June 2013, ISBN: 978-1-61208-271-4

3. Stefan Frey, Claudia Lüthje, Ralf Teckelmann, Christoph Reich "**Adaptable Service Level Objective Agreement (A-SLO-A) for Cloud Services**" International Conference on Cloud Computing and Services Science - CLOSER May 2013, page 457-462, SciTePress

4. Stefan Frey, Claudia Lüthje, Christoph Reich, Nathan Clarke, "**Cloud QoS Scaling by Fuzzy**" IC2E 2014 Proceedings of the 2014 IEEE International Conference on Cloud Engineering, Pages 343-348, ISBN: 978-1-4799-3766-0, DOI: 10.1109/IC2E.2014.30

5. Stefan Frey, Simon Disch, Christoph Reich, Martin Knahl, Nathan Clarke"**Cloud Storage Prediction with Neural Networks** CLOUD COMPUING 2015, The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization,Pages 52-56, ISBN: 978-1-61208-388-9

6. Matthias Lermer, Stefan Frey, Christoph Reich, "**Machine Learning in Cloud Environments Considering External Information** IMMM 2016: The Sixth International Conference on Advances in Information Mining and Management (includes DATASETS 2016), Pages 11-17 ISBN: 978-1-61208-477-0

Research Reports

1. Stefan Frey, Claudia Lüthje, Christoph Reich, Michael Maier, Tobias Zutavern, Markus Streif, "**SLA-Richtliniendokument für Cloud Dienstleistungen**"
Project: Autonomic Service Level Agreements as a Service, May 2013,
http://www.wolke.hs-furtwangen.de/assets/files/ASLAMaaS-SLA-Richtliniendokument.pdf

2. Stefan Frey "**Der Einsatz von genetischen Algorithmen für das Scheduling von Cloud-Instanzen**" HFU Informatik Journal 2011/12, November 2011; ISBN: 978-3-00-035367-3; p:11-17;

Appendix B - Further Background Information

History of Cloud Computing

Cloud computing is one of the most used buzzwords in information technology and it has become an extremely popular label, for all kinds of Internet and IT services. A simple Google search reveals its magnitude with billions of search results. Cloud computing thereby often is propagated as a new computing paradigm or as a information revolution [190]. The use of the term "cloud computing" really became popular in 2006 when big companies like Google and Amazon (Elastic Compute Cloud) [191] started using it. But the origins can be traced even further back in time into the late 1990s, where the MIT Technology Review states the first documented use of the term dates back to 1996 on a Compaq business plan [192]. Also academic papers as far back as in 1997 used the term [193] [194].

Aside from the terminology, the basic concepts of cloud computing date back even further. In the 1950s large-scale mainframe computers were installed in large companies, schools and government organizations. Because of the colossal hardware infrastructures and their immense costs of operating and maintaining, it was not feasible to grant each user sole access to his own mainframe. Therefore multiple users would have to share access to a mainframes computational power and storage via "dumb terminals". This is directly reflected in the cloud characteristics of shared resources and multi-tenancy [195].

Cloud Computing is largely based upon virtualization technologies and the Internet. Virtualization enables computer systems to share resources so that one physical resource can act as many virtual instances [196]. This is done by masking the real resources and granting access to them through a abstraction layer. IBM implemented the first virtual machines in the 70s with their operating system called VM. It allowed the division of huge mainframe computers into multiple distinct computers running in the same processing environment. At the same time developments in telecommunication played a crucial role for the cloud development. The whole concept of utilizing shared resources firstly became reasonably usable for everyone with progression towards the mass adoption of fast broadband connections.

Mather et al. [9] describe the emergence of cloud computing as a logical evolution of computing itself and view it as a development of the internet service providers (ISP). Figure 1 shows the evolution towards cloud computing. At first ISPs merely provided Internet access to their customers (ISP 1.0). After the access to the Internet became common, ISPs extended their

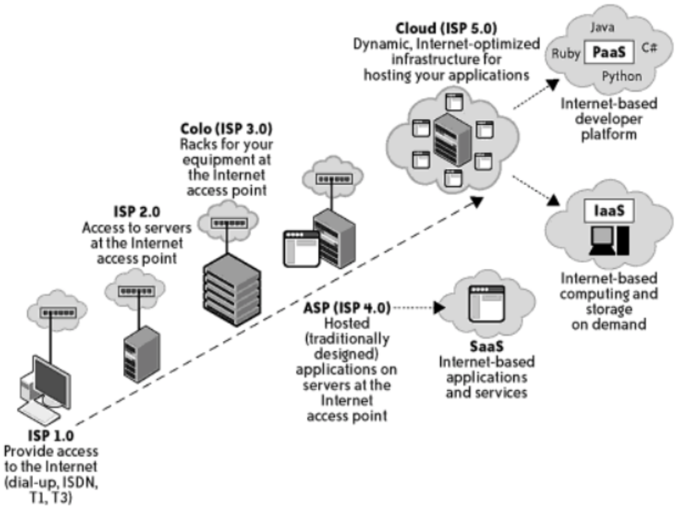

Cloud (ISP 5.0)
Dynamic, Internet-optimized infrastructure for hosting your applications
Java
Ruby
PaaS
C#
Python
Internet-based developer platform
Colo (ISP 3.0)
Racks for your equipment at the Internet access point
ISP 2.0
Access to servers at the Internet access point
IaaS
Internet-based computing and storage on demand
ASP (ISP 4.0)
Hosted (traditionally designed) applications on servers at the Internet access point
SaaS
Internet-based applications and services
ISP 1.0
Provide access to the Internet (dial-up, ISDN, T1, T3)

Figure 1: Evolution of Cloud Computing from [9, p.4]

offerings towards providing additional services like email and access to the servers at their facilities (ISP 2.0). This led to so-called collocation facilities (ISP 3.0), data centres where the ISP provided the infrastructure for the customer servers to be hosted. The next step in the evolution was the application service provider (ASPs), which added additional higher services like customized application for organizations (ISP 4.0). The service delivery model of ASPs may appear very similar to cloud computing, especially to the Software as a Service (SaaS) model, but there is a key difference in the underlying infrastructure. Within the ASP model every customer had his own dedicated infrastructure, so therefore every customer even had his own dedicated server and sole access to it. While cloud computing (ISP 5.0) offers access to shared resources.

Cloud Computing Definition

Even though cloud computing has become very popular in recent years, it is still difficult to get a generally accepted definition of it. What come closest to that is the definition of the US National Institute of Standards and Technology (NIST), which presents itself as a de facto standard for cloud computing:

"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. This cloud model promotes availability and is composed of five essential characteristics, three service models and four deployment models" NIST Definition of Cloud Computing [10]

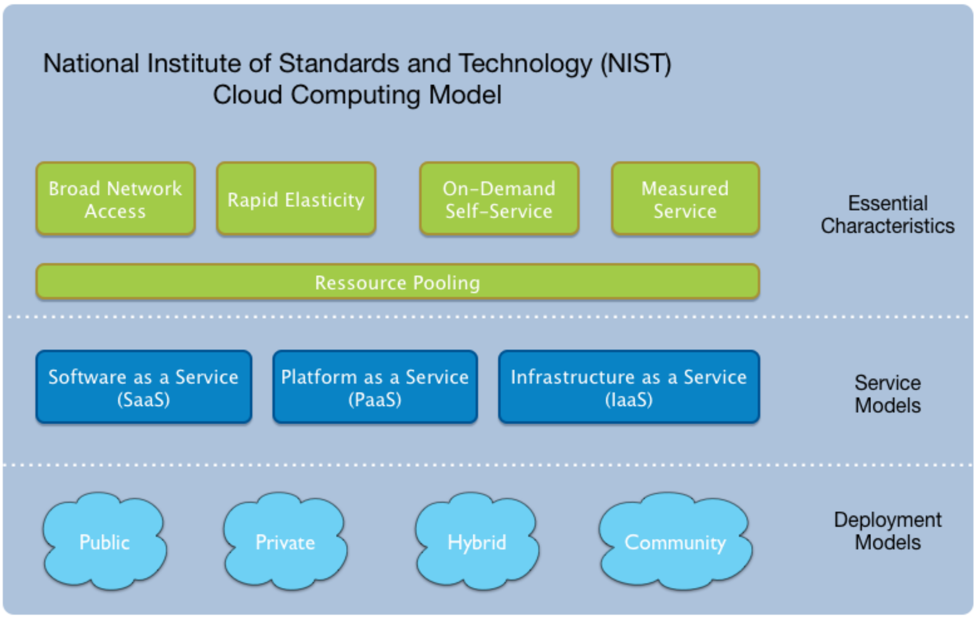


Figure 2: NIST Cloud Computing Model cf. [10]

A graphical representation of the cloud model based on the definition from NIST is shown in Figure 2. Here all the essential characteristics, service models and deployment models can be seen at once.

Characteristics

The NIST cloud computing definition identifies the following five essential cloud characteristics:

On-demand self service: Cloud customer can provision and manage cloud resources like computing power and network storage without requiring human interaction with a service provider.

Broad network access: Provided resources are accessed via networks (mostly the internet) using standardized protocols.

Resource pooling:The computing and storage resources of a provider are shared between multiple customers (multi-tenant model). A customer doesn't know the exact physical location of the resources he is using, but may specify a location on an higher abstraction level.

Rapid elasticity: Resources can be provided elastically and scaled rapidly to fulfill the customers current demand. This can also be done automatically.

Measured service:Consumption of resources by the customer gets metered. The cloud customer only pays for the resources and services he actually used. For transparency, the usage is controlled, monitored and reported to both provider and customer.

Cloud Models

As seen in Figure 2 another distinction is made between the three service models Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Figure 3 below shows the differences between the service models regarding flexibility and provider management.

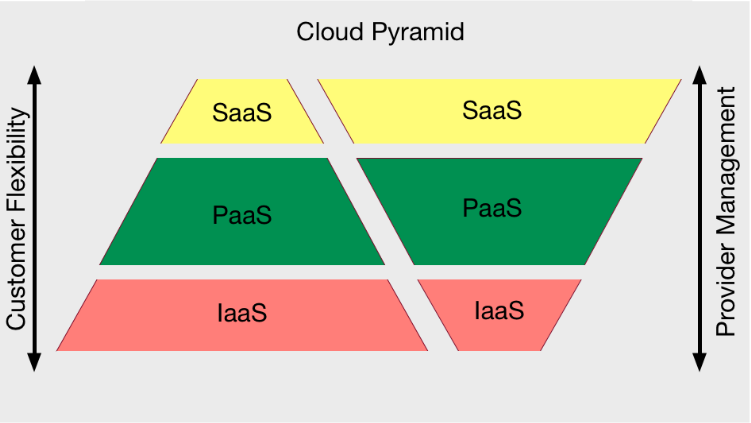


Figure 3: Cloud Pyramid of Flexibility and Management

It becomes evident that the models with increased flexibility in terms of usage come with less provider management and involvement, which means that the customer has to manage them by himself. This gets particularly clear with the Infrastructure as a Service model.

Infrastructure as a Service (IaaS)
Within the IaaS service model typical infrastructure components like virtual machines, CPU power, memory, storage and networking is provided. The user is able to run and deploy software on this resources at will. This may include operating systems. For this the physical infrastructure of the provider is virtualized by technologies like KVM, Xen or VMware and orchestrated into the virtual infrastructure for the user. The user does not control or manage the physical cloud infrastructure but can control and manage the virtual instance together with the operating system and the components that form the virtual instance, for example host-firewalls, load-balancer and so on. The main difference to traditional hosting servicer like server homing, is that the user only is provided with a virtual instance instead of the physical hardware. For cloud providers this means that they can better divide their resources between customers and so boost overall utilization. The provider charges the customer based on the amount and time of resources he has used on a on-demand model. The provider is responsible for the availability and usability of the infrastructure. Everything else like the reasonable usage, configuration and installation, as well as integration into the customers IT landscape and connection to other system remains with the customer. Often providers offer automatic installation services for operating systems based on virtual machine images. This may require based on the software additional licensing cost but mostly is offered as a free service. Storage as a Service is a IaaS model where only access to disk

space is granted. This model become increasingly popular in recent years, since it also enables users to easily share their data with third parties. Best known for this are providers like Dropbox [12], Microsofts OneDrive [13] or Google Drive [14]. Most popular IaaS providers at the moment are Amazon Web Services [15] and Rackspace [197]. Lately Google Compute Engine [198] and Microsoft Windows Azure [16] started their IaaS services.

Platform as a Service (PaaS)
This model aims at developers by providing comprehensive development and run environments to easily create, test and deploy applications. The user gets presented with a pre-configured environment by the provider, which means he does not have to deal with management or install of the virtualized infrastructure. The provider defines and configures as well all the development tool kits and environments, programming languages, APIs. libraries and databases. In extensive PaaS offerings users can immediately start to develop or deploy applications, without even installing tools on their machines. This model therefore enables developers a rapid propagation of their applications, location independent multi-tenant development and minimal entry effort and cost. Well known PaaS offers are Google App Engine [199] and Microsoft Windows Azure [16].

Software as a Service (SaaS)
In the SaaS model the user is provided with the capability to use the applications running on the providers cloud infrastructure. The users does neither manage, nor control the infrastructure or its components, the operations system or any application capabilities with the exception of settings within the applications. The user usually accesses the rented applications through either the web browser or an adapted program interface. In contrast to traditional software usage the customer subscribes to the service or uses a pay-per-use model. It is quite common that SaaS provider use PaaS or IaaS infrastructures or third party providers to benefit form the high elasticity. Popular SaaS providers are Google Apps [200], Google Docs [201], Microsoft Office 365 [202] and Salesforce.com [203].

Deployment Models

The NIST definition identifies four different deployment models. As seen in Figure 2 these are: Public cloud, Private cloud, Hybrid cloud and Community cloud.

Private clouds are used exclusively by one single organization, for example by a company for serving its internal needs. These cloud are also often referred to as internal clouds, since the infrastructure is commonly run inside the private network of the using entity. The cloud system may be operated and managed by the organization itself, a third party or a combination of both.

Public clouds represent the exact opposite. Here the usage of the cloud is not limited to one user respectively one organization, instead the services are offered to a larger group of customers or the general public. This cloud type is also called external clouds, since they are managed, hosted and operated by a third-party vendor outside the influence of the customer. The services are commonly accessible through web-applications, web-services or communications protocols like the Remote Desktop Protocol (RDP) or Secure Shell (SSH). A well known provider for public cloud services is the Amazon Elastic Compute Cloud (EC2).

Community clouds are used exclusively by a specific community of consumers. This usually are organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations cf. [10]). The infrastructure may exist on or off premises, as well as may be owned, operated and managed by one or more of the community entities or a third party or any combination of the mentioned before.

Hybrid clouds present a composition of two or more of the mentioned above cloud deployment models. The infrastructures are bound together but still remain unique entities. This is done by enabling data and application portability through the use of standardized or proprietary technology.

Reference Architecture

Different cloud architectures include various components. In terms of a wide acceptance the NIST proposed architecture is chosen to give an overview of possible components. The NIST architecture identifies five major actors. Table 1 shows these actors and gives the definition of them. The descriptions are taken directly from the NIST Cloud Computing Reference Architecture [2].

Actor	**Definition**
Cloud Consumer	A person or organization that maintains a business relationship with, and uses service from, Cloud Providers.
Cloud Provider	A person, organization, or entity responsible for making a service available to interested parties.
Cloud Auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers.
Cloud Carrier	An intermediary that provides connectivity and transport of cloud services from Cloud Providers to Cloud Consumers.

Table 1: Cloud Computing Actors from NIST [2]

The Cloud Provider is the entity that delivers the cloud service to the Cloud Consumer. Since the delivered service are to be consumed over a broad network access a third party. the Cloud Carrier is used to provide connectivity between the both. This can for example be the Internet Service Provider or in case of a private cloud it would be the IT department responsible for the network. In some cases different roles can be adopted by one organization this may be the case especially with private clouds but also if the Cloud Provider at the same time is a ISP and therefore takes the role of the Cloud Carrier.

A Cloud Provider usually owns one ore more different data centers (facilities). Here the physical hardware for the cloud infrastructures and the environment needed (cooling, electric power, ...) to support it is hosted. In the presented architecture this is grouped together as the *physical resource layer*. Figure 4 shows the overview of the components of the NIST cloud architecture.

In between the *physical resource layer* and the *service layer*, which provides the three different service model (IaaS, PaaS, SaaS) there is the *resource abstraction and control layer*. These three layers form the *Service Orchestration* module which is used to compose and deliver the actual cloud services. The *service layer* on top is where the Cloud Provider defines the access interfaces for the Cloud Consumer. In the course of this it is possible but not necessary that SaaS applications can be built on top of PaaS components and these again can be built on top of IaaS components. So for example, a PaaS environment could be hosted on top of a virtual machine from an IaaS cloud.

The *resource abstraction and control layer* contains all system and software components that the Cloud Providers uses to abstract the physical resources into virtualized cloud resources. This may include hypervisors or virtual machine monitors, virtual storage, virtual machines and so on. The virtualization in this layer allows resource pooling, dynamic allocation and measurement of the consumed services. Here the control (allocation, access control and usage monitoring) of the cloud resources is realized.

In addition to this core component, there is the *cloud service management* and modules for *security* and *privacy*. The cloud service management module includes the functions which are needed by the Cloud Provider for the management and operation of the provided cloud services. These are grouped into three different Cloud Provider perspectives. Firstly the *business support*, which includes the processes supporting the service offering and dealing with clients, like the customer management, contact management, accounting & billing and reporting & auditing. Then there are the processes involved in the *provisioning and configuration* of the cloud services. These include the monitoring and metering as well as the SLA management. Lastly there are the mechanisms to support *portability and interoperability*, which shall enable the service customer to freely merge and compose cloud services between different cloud providers. A detailed description of all these three modules and their individual components can be found in the NIST publication [2, p. 15].

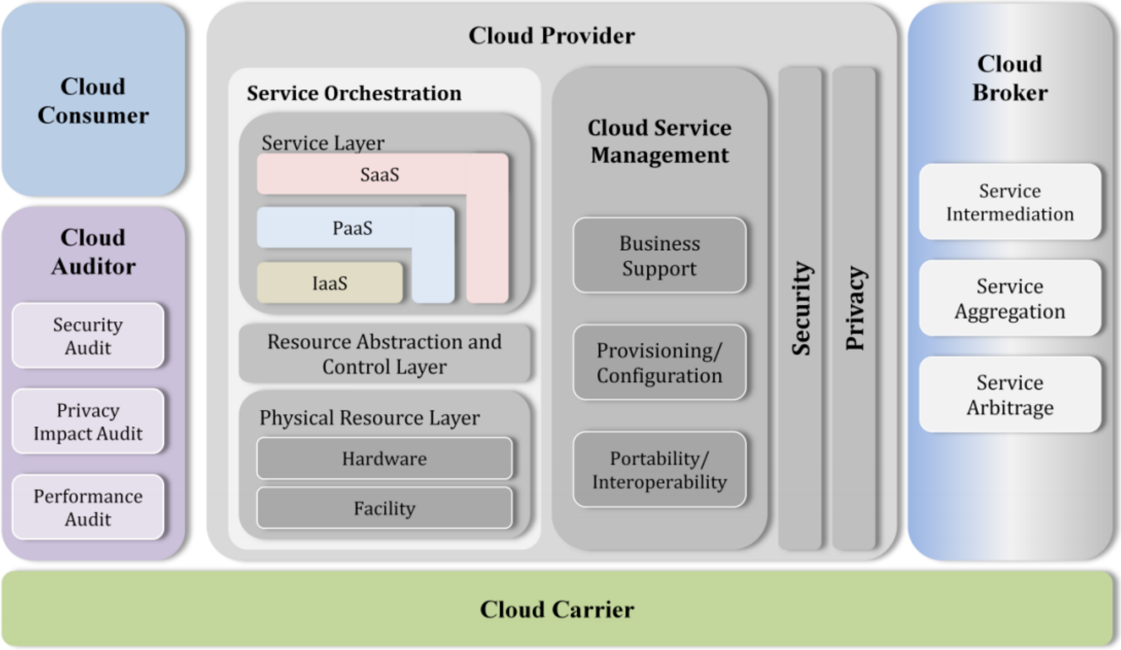


Figure 4: Cloud Computing Reference Architecture from NIST [2]