universität wien

DISSERTATION

A Negotiation Framework for Consumer-Provider Contracting in the Cloud based on an Economic Cost Model

Verfasser

Mag. Dr.techn. Werner Mach

angestrebter akademischer Grad

Doktor der Wirtschaftswissenschaften (Dr.rer.oec.)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 784 175
Dissertationsgebiet lt. Studienblatt:	Wirtschaftsinformatik
Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. Erich Schikuta

To my wife Sylvia and our children Bernhard, Ursula and Andreas

Contents

Abstract

Dealing with electronic goods requires electronic contracting. Successful electronic contracting which satisfies all involved parties is a challenging issue. A participating party's contracting decision process is sophisticated and driven by various kinds of parameters. The liquidity of electronic markets depends on the speed and accuracy of getting agreements between partners. The Cloud is a reliable commodity to provide business value. Negotiation and Re-Negotiation are the prerequisites for each successful contracting process between two or more partners. One of the negotiation process' key elements is the way of how decisions are done. In case of human partners, the decisions are done based on their experience and personal preferences. To automate these processes without human interaction is a challenging task.

In this thesis we propose a novel negotiation and re-negotiation framework for autonomous web service contracting. The framework can handle bi-directional negotiating as well as auctioning. We show which components are necessary and crucial for this purpose. We developed a new cloud cost model that enables making more accurate decisions for consumers and providers. We improved the accuracy of common cost models by considering variable cost in addition to fixed cost. A new developed business rule ontology enables the framework storing all kinds of business related knowledge to make company-specific decisions during negotiation. Furthermore we developed a novel negotiation approach, the so-called "bazaar-style" negotiation in contrast to the present "super-market" negotiation style which is a simple "take it or leave it" approach. That means that a provider offers his services with fixed service parameters and the consumer accepts or rejects his offer. We extend this approach by allowing ranges for each service parameter. This gives consumers as well as providers the ability to adapt offers and counter-offers during negotiation. For the "bazaar-style" negotiation we developed an analytical model which allows to predict the probability of service level agreement matching.

Finally, we created several implementations that justified our approach and validated our results. For forecasting service level agreement matching we developed our own simulation. In order to implement and simulate the framework which we developed within this thesis, we used already known and common simulation environments such as *CloudSim*. In this simulation environment we executed and analyzed typical scenarios.

Zusammenfassung

Der Handel mit elektronischen Gütern erfordert elektronische Verträge. Erfolgreiches elektronisches Vertragswesen das alle beteiligten Parteien zufriedenstellt, ist eine Herausforderung. Der Entscheidungsprozess für die Beteiligten ist anspruchsvoll und durch verschiedenste Parameter getrieben. Die Liquidität von elektronischen Märkten hängt von der Geschwindigkeit und Genauigkeit ab mit der es zu einer Einigung zwischen Partnern kommt.

Die Cloud ist ein zuverlässiges Handelsgut, um damit Geschäftserfolge zu erzielen. Verhandeln und Gegen-Verhandeln sind die Voraussetzungen für jeden erfolgreichen Verhandlungsprozess zwischen zwei oder mehreren Partnern. Eine der Schlüsselelemente des Verhandlungsprozesses ist die Art und Weise, wie Entscheidungen gefällt werden. Im Falle von Menschen als Partner werden die Entscheidungen anhand ihrer Erfahrungen und ihrer Vorzüge durchgeführt. Diesen Prozess zu automatisieren, ohne jegliche menschliche Interaktion, ist eine große Herausforderung.

In dieser These stellen wir ein neuartiges Verhandlungsframework für autonomes Verhandeln von Web-Services vor. Das Framework beherrscht bi-direktionales Verhandeln sowie das Verhandeln in Auktionen. Wir zeigten, welche Komponenten für diesen Zweck kritisch und notwendig sind. Wir entwickelten auch ein neues Kostenmodell für Cloudsysteme, das Konsumenten und Anbieter gleichermaßen ermöglicht präzisere Entscheidungen zu treffen. Wir verbesserten die Genauigkeit von herkömmlichen Kostenmodellen durch die Berücksichtigung von variablen Kosten in Ergänzung zu den sonst nur verwendeten fixen Kosten. Eine neu entwickelte Systematik für Unternehmensregeln ermöglicht es, jede Art von geschäftsrelevantes Wissen im Framework zu speichern, um unternehmensspezifische Entscheidungen während der Verhandlungen zu treffen.

Darüber hinaus entwickelten wir einen neuartigen Verhandlungsansatz, die sogenannte "Bazaar-ähnliche" Verhandlung. Im Gegensatz dazu ist der derzeit übliche "Super-Markt" Ansatz ein "Nimm oder Lass es" Prinzip. Das heißt, dass ein Anbieter seine Services mit fixen Parametern anbietet und der Konsument das Angebot annimmt oder ablehnt. Wir erweiterten den bisherigen Ansatz, indem wir eine Bandbreite für jedes Service zulassen. Das ermöglicht den Konsumenten sowie den Anbietern ihre Angebote und Gegenangebote während der Verhandlung anzupassen. Für diese "Bazaar-ähnlichen" Verhandlungsmethode entwickelten wir auch ein analytisches Modell zur Vorhersage der Wahrscheinlichkeit mit der es zu einer Einigung kommt.

Abschließend überprüften wir unsere Ergebnisse anhand verschiedener Implementationen. Für das Vorhersagemodell wurde eine gesonderte Simulation entwickelt. Um das gesamte Framework, welche wir in dieser Arbeit entwickelt haben, zu simulieren, haben wir vorhandene und bekannte Simulationsumgebungen wie *CloudSim* verwendet. In dieser Simulationsumgebung haben wir im Anschluss typische Szenarien analysiert.

Acknowledgements

First and foremost, I would like to thank my advisor, Erich Schikuta, for showing me the research path, as well as for all his suggestions and assistance during my work and for his advice and expert guidance. This is my second thesis under Erich's supervision and covers a completely different field of research compared to the first thesis. Erich and I had a lot of very fruitful discussions during our research work. I am very proud that our research resulted in a considerable number of publications. In addition, I would also like to thank Benedikt Pittl, a Master student, for supporting me in writing our research papers together with my supervisor. And also thanks for their hard work of implementing our framework and for helping to improve my thesis.

Last, but not least, I would like to thank my wife, Sylvia, and my children, Bernhard, Ursula and Andreas, for their abundant patience and understanding when I spent hours on my computer conducting my research.

I have done all my research work while working full time. Research is completely different from the work I do for a company and is very satisfying to me. Spending hours and hours in front of my computer was not a burden, it was a great pleasure for me.

Werner Mach
Vienna, Austria
June 2014

Remarks

I have tried very hard to find all owners of the pictures to get the allowance to use their pictures in my work. If there is any copyright violation, please contact me.

I use "we" and "our" in my thesis as a gesture of respect to the research community. This convention is part of my attitude towards that community.

In the chapter *Conclusion* I focussed on 10 main points of my thesis. Seven out of these 10 are dedicated to this work while three are statements of my life experience and personal philosophy.

List of Publications

Following publications are all peer reviewed from at least 3 independent reviewers, which were the results of this thesis. The chapters 1, 3, 4, 5, 6, 7 and 8 are partly derived from these publications.

- Werner Mach and Erich Schikuta. A Consumer-Provider Cloud Cost Model Considering Variable Cost. In *International Conference on Cloud and Green Computing (CGC 2011) within IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, DASC 2011*, 12-14 December 2011, Sydney, Australia.

  This publication is Be-ranked in the B-list of the Faculty of Computer Science at the University of Vienna, Austria.

  This paper presents an analytical economic cost model for cloud computing aiming for comprising all kind of cost of a commercial environment. Extending conventional state-of-the-art models considering only fixed cost we develop a concise but comprehensive analytical model which includes, besides fixed cost, also variable cost allowing for developing and evaluating business strategies for cloud environments. These strategies can be used for cloud providers and cloud consumers as well. The major goal of our model is to comprise all important economic fundamentals and methods. Thus, this new model supports the decision process between business cases to apply and allows cloud consumers and cloud providers both to derive their own business strategies and to analyze the respective impact to their business. Further by this model the energy efficiency of cloud systems can be evaluated respective to chosen business models.

- Werner Mach and Erich Schikuta. A generic negotiation and re-negotiation framework for consumer-provider contracting of web services. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services (iiWas2012)*, pages 348–351, New York, NY, USA, 2012. This publication is in the B-list of the Faculty of Computer Science at the University of Vienna, Austria.

  Electronic contracting is key issue for establishing liquid markets dealing with electronic goods. This paper presents a framework for automatic negotiation between Web services. The major goal of the framework is comprising all necessary components for negotiation and re-negotiation. The capabilities and the components are described both for Web service consumer and provider. The re-negotiation frame- work is based on independent control loops and associated knowledge bases for consumer and provider implementing specific business strategies. An economic cost model considering variable cost and a business rule repository are main parts of the knowledge bases. Furthermore the framework can handle auctioning by using predefined workflows as so-called auctioning plug-ins.

- Ralph Vigne and Werner Mach and Erich Schikuta. Towards a Smart Webservice Marketplace. In *IEEE Conference on Business Informatics (CBI2013)*, Vienna, Austria, 2013. This publication is B-ranked in the B-list of the Faculty of Computer Science at the University of Vienna, Austria.

  Electronic contracts are crucial for future e-Business models due to the increasing importance of Web services and the cloud as a reliable commodity enabling service-based value chains. Negotiation is the prerequisite for establishing a contract between two or more partners. These contracts are usually based on Service Level Agreements (SLAs). In this paper we present the framework of a smart Web service marketplace, which allows for automatic, autonomous, and adaptive negotiation and re-negotiation of Web services based on economic principles. Our approach enables market based

service trading following a bazaar style and extends the classical supermarket approach typical for service negotiation today. We extend the WS-Agreement standard by feasible workflows to support auctioning for negotiation and re-negotiation. A specific highlight of our framework is the mapping of business strategies defined by economic goals of the respective organization into an ICT enabled framework. It facilitates autonomic agents acting as organizational representatives stipulating SLAs without human interaction. This allows for business transactions transparently to the environment but adhering to business objectives of the originating organization.

- Werner Mach and Benedikt Pittl and Erich Schikuta. A Business Rules Driven Framework for Consumer-Provider Contracting of Web Services. In *iiWAS, 2013 , 229.* This publication is in the B-list of the Faculty of Computer Science at the University of Vienna, Austria.

  This paper represents an economic cost model for cloud computing aiming at comprising all kinds of cost of a commercial environment. To extend conventional state-of-the-art models considering only fixed cost, we developed a concise but comprehensive analytical model, which includes also variable cost allowing for the development and evaluation of business strategies for cloud environments. These strategies can be used for both cloud providers and cloud consumers. The major goal of our model is to comprise all important economic fundamentals and methods. Thus, this new model supports the decision-making process to be applied with business cases and enables cloud consumers and cloud providers to define their own business strategies and to analyze the respective impact on their business. On the basis of this model, also, the energy efficiency of cloud systems can be evaluated according to chosen business models.

- Werner Mach and Erich Schikuta. Toward an economic and energy-aware cloud cost model. In *Concurrency and Computation: Practice and Experience Volume 25, Number 1, January 2013 , 2471-2487* Wiley Journal. This publication is Ae-ranked in the A-list of the Faculty of Computer Science at the University of Vienna, Austria.

  This paper represents an economic cost model for cloud computing aiming at comprising all kinds of cost of a commercial environment. To extend conventional state-of-the-art models considering only fixed cost, we developed a concise but comprehensive analytical model, which includes also variable cost allowing for the development and evaluation of business strategies for cloud environments. These strategies can be used for both cloud providers and cloud consumers. The major goal of our model is to comprise all important economic fundamentals and methods. Thus, this new model supports the decision-making process to be applied with business cases and enables cloud consumers and cloud providers to define their own business strategies and to analyze the respective impact on their business. On the basis of this model, also, the energy efficiency of cloud systems can be evaluated according to chosen business models.

- Werner Mach and Benedikt Pittl and Erich Schikuta. A Forecasting and Decision Model for Successful Service Negotiation. In *11th IEEE International Conference on Services Computing June 27 - July 2, 2014*, Anchorage, Alaska, USA, IEEE Computer Society. This publication is Ae-ranked in the A-list of the Faculty of Computer Science at the University of Vienna, Austria.

  Future e-business models will rely on electronic contracts which are agreed dynamically and adaptively by web services. Thus, the automatic negotiation of Service Level Agreements (SLAs) between consumers and providers is key for enabling service-based value chains. The process of finding appropriate providers for web services seems to

be simple. Consumers contact several providers and take the provider which offers the best matching SLA. However, currently consumers are not able forecasting the probability of finding a matching provider for their requested SLA. So consumers contact several providers and check if their offers are matching. In case of continuing faults, on the one hand consumers may adapt their Service Level Objects (SLOs) of the required SLA or on the other hand simply accept offered SLAs of the contacted providers.

Thus, this paper proposes an analytical forecast model, which allows consumers to get an assessment of the probability to find matching providers. Additionally, we present an optimization algorithm based on the forecast results, which allows adapting the SLO parameter ranges in order to find at least one matching provider. Not only consumers, but also providers can use this forecast model to predict the prospective demand. So providers are able to assess the number of potential consumers based on their offers too. Justification of our approach is done by simulation of practical examples checking our theoretical findings.

- Werner Mach and Benedikt Pittl and Erich Schikuta. A Prediction Model for the Probability of SLA Matching in Consumer Provider Contracting of Web Services. In *CoRR arXiv abs/1401.2440*, 2014.

1 Introduction

1.1 Motivation

An important characteristic of Cloud markets is the liquidity of the traded good. A sufficient number of market participants is necessary for the proper function of the market. Only if following conditions are fulfilled, providers and consumers are willing to join:

- Resource consumers, if they are able to find what they need quickly.
- Resource providers, if they can be fairly certain that their resources will be sold.

Electronic contracts have become very important due to the increasing popularity of web services and Cloud resources as a reliable commodity to provide business value. Negotiation is the prerequisite for a successful contract between two ore more partners. These contracts are usually based on SLA's (Service level agreements). As described in [1], service providers can make use of SLA technology to advertise and offer their service capabilities while consumers are able to formalize their service level objectives through SLAs. These contracts are described in protocols like WS-Agreement [2]. WS-Agreement has become a standard for providing electronic contracts. The reason for the success of WS-Agreement is the flexibility and ability for using it in automated service integration in Cloud based systems. WS-Agreement defines the involved partners, the context of the agreement and describes the necessary guarantees of the defined service levels.

Visualization enables providers to create a wide range of resource types and allows consumers to specify their needs precisely. The SLA often differs slightly.

As simulation in previous research shows, large resource variability of both sides results in a large number of resource definitions. Therefore the probability of matching decreases rapidly. But to ensure sufficient liquidity in the market the probability should be high. The matching probability can be used as a measure to determine how attractive a market would be to providers and consumers.

In recent years, a large number of commercial Cloud providers have entered the utility computing market, offering a number of different types of services. There are resource providers who only provide computing resources like *amazon.com* or *Tsunamic Technologies* and SaaS providers who sell their own resources together with their own software services like *Google Apps* or *Salesforce.com*. In the current market, providers only sell a single type of resources (with the exception of *amazon.com*). This limited number of different resource types enables a market creation, since all demand is channeled towards very few resource types.

To fully exploit the potential of open markets, a large number of providers and consumers is necessary. A large number of potential traders might inflate the variety of resources which leads to the problem that the supply and the demand are spread across a wide range of resources.

To give traders few restrictions, an approach is needed which allows traders to define their resources (or requirements) freely while facilitating SLA matching. Current adaptive SLA matching mechanisms are based on semantic ontologies like Web Ontology Language OWL [3] and OWL-S (former DAML-S) and other semantic technologies. However, none of these approaches address the issues of the open market and deal with semi-automatic definition of SLA mappings enabling negotiations between inconsistent SLA templates.

At present the knowledge base used in Self-Governing ICT (Information and Communication Technology) Infrastructure Architectures is not further explained. There is no detailed description showing e.g. how it works and which kind of information are stored exists. No existing framework for re-negotiation in clouds includes a cost model as knowledge base and none of them considers business rules to give answers to typical business models. A model which considers both, energy- and cost-efficiency doesn't exist. The related work discussion has shown that the existing cloud cost models neglect traditional economic principles. Cost models that consider variable cost are better suited for our economy situation today. This distinction is also necessary to derive more accurate business strategies. The work discussion emphasized the following gaps:

- **Lack of economically based frameworks.** Currently we only find cloud cost models with breakdown of cost types. The models, which are not integrated in a framework, have the main focus on resource and power consumption optimization.
- **How can traditional economic fundamentals and methods be used in a negotiation framework.** Existing cost models for clouds consider only fixed cost, they neglect a distinction between fixed and variable cost which would increase accuracy and reliability of results. To implement a comprehensive cloud cost model operating production factors, the production, sales theory and investment and finance have to be taken into consideration.
- **Lacking business models for both cloud-consumer and -provider.** Business models are more investigated for cloud provider than for consumer. This research proposal aims the goal describing and implementing business models for both.
- **Missing re-negotiation mechanisms in frameworks.** As service level agreements (SLA) templates have fixed values, no individual offer can be provided. Therefore only the supermarket approach in negotiation exists. Supermarket approach means that the consumer can only agree to fixed values of service levels of the provider or otherwise has to leave it. The idea of this research proposal is to use not only fixed values of service level parameters but rather ranges of parameter values.
- **Service Value Chains.** The financial value perspective for cloud services tends to shift from CAPEX (Capital Expenditures) to OPEX (Operating Expenditures) to a pay-as-you-go model which influences the cashflow. Sources of revenue and outgoing cash expenditure are on a usage basis based on a unit such as time, volume, or component. The main goal of the pay-as-you-go model is to reduce the risk of the business capital by replacing initial investments with e.g leasing fees and maximizing the leverage of that capital to make additional investments. Consequently, it is possible to react quicker in response to new technologies [4].

The arising questions are:

- How should the Cloud Cost Model be built to support fundamental economic methods?
- Which technical and economic parameters are sensitive for the input?
- How can this cost model be used as a knowledge base for negotiation frameworks?
- What framework has to be built for developing and evaluating business strategies for both consumers and providers?
- Can the model answer typical business model questions for both consumers and providers ?

- Can the framework be extended from a static to a dynamic view of business (e.g. reactions to SLA violation during operation, support online auctions)?
- What distributed architectures can be used for the approach? Must a new one be built?

We have taken the following steps in our research work to find answers to the previous questions by:

- Analyzing existing Cloud Cost Models.
- Developing a comprehensive analytical cloud cost model supporting all traditional economic methods.
- Creating a business rules ontology to be used in knowledge base systems.
- Investigating current SLA matching mechanisms.
- Developing a prediction model for estimating SLA matching probability for both participants (consumer and provider)
- Studying existing negotiation frameworks in terms of autonomous negotiation and re-negotiation behaviors.
- Creating a framework for autonomous negotiation and re-negotiation for both, consumer and provider.
- Initial implementation of the framework and studying typical business scenarios of consumer-provider contracting.

1.2 Overview

This thesis presents a study of all components necessary to build an autonomous negotiation and re-negotiation framework for consumer-provider contracting of web services. First of all we investigated current cloud cost models. Based on this research we developed a cloud cost model that supports fixed and variable cost. We have found that current cost models consider fixed cost only. Therefore it is not possible to calculate the cost accurately enough for delicate business decisions. As a consequence we developed a completely new cloud cost model based on traditional economic principles like resources as production factors and distinguishing between fixed and variable cost.

The second major step of our research was the creation of a business rule ontology. This ontology is necessary for standardized describing business rules. We also developed an effective algorithm for the transformation of business rules written in natural language into a machine readable presentation. Standardized machine readable business rules are the precondition for using it in a knowledge base.

Next we investigated current SLA matching mechanisms. The commonly used super-market ("*take it or leave it*") approach is not flexible enough for using it in negotiation and especially in re-negotiation processes. We extended the super-market approach by using ranges for each SLA parameter. Defining ranges for SLA parameters in SLA templates are defined in standards like WS-Agreement, but not used in matching mechanisms. Using ranges instead of fixed values, we were able to predict the possible SLA matching probability. Based on these findings we developed an analytic prediction model for both, consumer and provider. This prediction model is an efficient tool for adapting SLA parameters during the whole negotiation process. Further, it is also useful for estimating possible results without starting the negotiation process.

Based an all these results we were able to define a comprehensive negotiation and re-negotiation framework. Before choosing this definition we carefully investigated all existing negotiation frameworks. In the course of this thesis we developed a symmetric framework that can be used for both participants. A symmetric framework is a framework in which all components are identical for both, consumers and providers.

The implementation of the developed framework gives the opportunity of investigating typical business scenarios. Simulations of these scenarios are used for doing extensive surveying of their behaviour. We are convinced that our framework stimulates the liquidity of electronic market places.

1.3 Organization of this Thesis

In chapter 1 we describe the motivation for our research endeavour. This chapter encompasses the main research questions as well as a short overview. The goal of chapter 2 is to introduce a rough state of the art analysis of negotiation, auctioning and service level agreements. In this chapter an introduction of autonomous systems, knowledge bases and business rules is presented. Traditional economic principles, current cloud cost models and black board systems are introduced. All these findings are used in our negotiation and re-negotiation framework.

Chapter 3 describes our developed business rule ontology. As a result of our research we introduce a comprehensive analytical SLA matching prediction model in chapter 4. A cloud cost model considering variable cost and based on traditional economic principles is shown in chapter 5. In chapter 6 the generic negotiation and re-negotiation with their components is described in detail.

All our research is implemented in a real simulation environment to simulate typical business scenarios of both service consumer and providers. Chapter 7 is dedicated to show all the implemented components and present a discussion on the results.

The thesis ends with chapter 9 that contains further research which is followed by a conclusion of this thesis as well as ten statements of the thesis.

1.4 Working Terms

Before describing our research we want to clarify the terms we use in this thesis.

Terms like Service Level Agreements, Consumer, Provider, Negotiation/Re-Negotiation and so on are used in the area of computer science, but sometimes conflicting definitions exist. Thus, it is important to set working definitions in the context of this thesis, in order to precisely state the way of how the terms are used.

The contracts made within the service negotiation are SLAs. A SLA is detailed with so called Service Level Objectives (SLOs), which determine the Quality of Service (QoS) [5, 6]. So the quality of the provided services is represented by QoS, parameters. Downtime, Bandwidth and Response Time are examples for QoS parameters [7]. In this thesis we use the terms SLO parameters and service parameters synonymously.

SLO are expressed in terms of so-called Service Level Indicators (SLI) [8]. These SLIs measure a service such as mean time to restore.

A typical negotiation scenario consists of a party offering services, which we call provider and a party buying services, which we call consumer.

In the course of this thesis we use the term "Service" for hardware as well as for software services. A hardware service can be e.g. a cloud storage service whereas a software service can be e.g. a streaming service.

Each service can be described by its functional as well as its non-functional requirements. The functional agreements represent the services provided. Non-functional agreements describe e.g. the quality of these services.

2 State of the Art : Concepts, Models and Methods

In this chapter we give an overview of current research in the field of consumer-provider service contracting. The primary systems, knowledge bases and business rules are explained. We describe traditional economic principles and current cloud cost models. Additionally, we provide a short introduction in autonomous systems for ICT systems.

2.1 Service Level Agreements

The importance of electronic contracts is crucial for future e-Business models due to the increasing importance of Web services and the cloud as a reliable commodity providing business value [9]. Negotiation is the prerequisite for a successful SLA between two ore more partners. As described in [1], service providers can make use of SLA technology to advertise and offer their services capabilities while consumers are able to formalize their service level objectives through SLAs. A Service Level Agreement is a formal negotiated agreement and an explicit contract between a provider and a consumer of a service. A SLA is defined through a negotiation process between consumer and provider [10].

Service consumers subscribe services from service providers. Therefore they conclude a contract. These contracts can be represented by Service Level Agreements, SLAs. [11] SLAs are widely accepted for governing IT services [7] [12].

Typically, a SLA contains the names of the service provider and service costumer as well as the period of validity. Additionally functional and non-functional agreements are attached to the SLA.

These contracts are described in protocols like (Web Service Agreement) [13]. The WS-Agreement specification is a standardization effort conducted by the Open Grid Forum (OGF) in order to facilitate creation and monitoring of SLAs. It is a simple request-response protocol for agreement creation and monitoring.

WS-Agreement has become a standard for providing electronic contracts. The reason for the success of WS-Agreement is the flexibility and ability for using it in automated service integration in cloud based systems. WS-Agreement defines the involved partners, the context of the agreement and describes the necessary guarantees of the defined service levels.

The SLA Lifecycle can be split up into six phases [14]:

1. development of service and service templates,
2. discovery and negotiation of a SLA,
3. service provisioning and deployment,
4. execution of the service,
5. assessment and corrective actions during execution (parallel phase to execution of the service), and
6. termination and decommission of the service.

2.1.1 WS-Agreement Language and Protocol Specification

The following short overview is derived from the WS Agreement specification *Web Services Agreement Specification (WS-Agreement)* [13] and describes a Web Service Protocol for establishing agreements between two parties (serviced provider and -consumer). The WS agreement specification uses a extensible XML language for specifying agreements and agreement templates. The specification has three parts:

- a XML schema
- a set of port types
- operations for creation, expiration and monitoring of agreements

WS-Agreement standardizes the terminology, concepts, structure and types of agreement terms, agreement template with creation constraints, a set of port types and operations for creation, expiration and monitoring of agreements. A Web Service Description Language (WSDL) is also defined to express the message exchanges and resources needed to indicate the state of the agreement. The WS-Agreement specification defines a XML data structure that allows applications to be easily read and operate with them.

The objective of WS-Agreement from Open Grid Forum (OGF) is the following: "*The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of service providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime.*"

"*An agreement between a service consumer and a service provider specifies one or more service level objectives both as expressions of requirements of the service consumer and assurances by the service provider on the availability of resources and/or on service qualities.*" [13]

The WS-Agreement specification addresses the following requirements:

- Creating agreements for services defined by any domain specific service terms. The service objective description referencing the elements defined in the service description.
- Allowing creation of agreements for existing and new services.
- Using of any condition specification language (any domain specific or standard condition expression language) for defining service level objectives.
- The protocol must provide symmetry for supporting a large number of scenarios and also for the modeling of user specific scenarios.
- Composable with various negotiation models and negotiation protocols.
- Must be standalone for simple agreement creation independent of any negotiation model.
- Independence of the agreement document structure and the protocol used.

2.1.1.1 WS-Agreement and WS-Agreement Template Structure

The typical structure of an agreement and an agreement template is depicted in figure 2.1. The tag *Agreement* covers the entire encapsulated agreement containing an agreement context and a collection of agreement terms. *Name* is an optional tag and is not a unique identifier for the agreement. The required element *Agreement Context* is not specified in the terms and describes who the parties are, which kind of service is offered and the duration period. Each agreement consists of service definition *Terms* and zero or more guarantee terms.

Based on an agreement a client can make an offer to an agreement factory. This so-called creation offer has the same structure as an agreement. The agreement factory announces the types of offers it is willing to accept in terms of agreement templates. The optional *Agreement Creation Constraints* are designated for providing values for the terms may take in a concrete agreement.

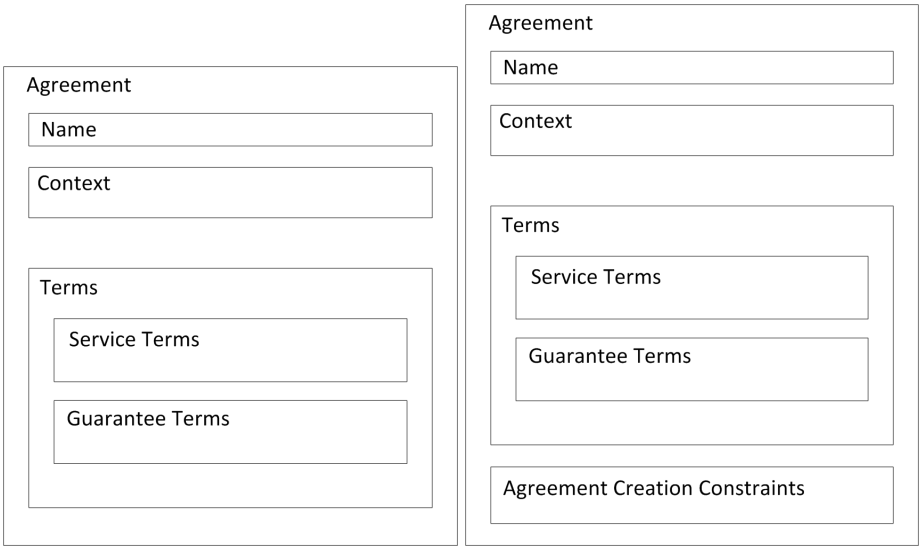


Figure 2.1: *Agreement Structure (left), Agreement Template Structure (right)*

2.2 Negotiation and Auctioning

This section represents a short overview about negotiation and auctioning. In this thesis we distinguish between negotiation and re-negotiation. Negotiation is defined as the process defining SLAs, re-negotiation is the process altering existing SLAs, e.g. in case of a SLA violation. Negotiation is a proactive process while re-negotiation is a reactive process.

2.2.1 Negotiation

In current SLA research the negotiation between provider and consumer is only insufficiently considered. In most cases *one-phase negotiations* are being used to keep the effort for the negotiation small. In a one-phase negotiation, service providers offer their services in form of agreement templates. A template may contain a number of alternative service descriptions with different service quality. The consumer can choose one of these templates which fulfills his requirements best. After choosing the offered template consumer and provider create an agreement. This approach is like buying in a supermarket: the provider offers a set of products and the consumer chooses one or more of them.

In [13] Negotiation and Re-negotiation are defined as:

> *Negotiation is a process between an agreement initiator and an agreement responder to reach an acceptable agreement offer from an initial agreement template. Agreement offer negotiation is a non-binding, bi-lateral process that comprises exchange of information in order to find a consensus for acceptable agreement offers.*
>
> *Re-negotiation is a process between an agreement initiator and an agreement responder to reach an acceptable agreement offer in order to alter an existing agreement.* Altering an existing agreement is achieved by creating a re-negotiated

agreement, which supersedes the original agreement. Hence, an existing agreement can only be re-negotiated once, but the process can be repeated with the new (superseding) agreement. The number of such re-negotiations is not limited. Re-negotiation of an existing agreement may have direct impact on the provisioning of active services.

The Open Grid Forum has defined an extension to the WS-Agreement specification [13], the WS-Agreement Negotiation version 1.0 [15] which allows multi-round negotiation necessary in many scenarios. However, the WS-Agreement Negotiation protocol does not explicitly support auctioning and biding. Nevertheless, the protocol can be used for communication. Auctions and biding are complex negotiation processes [16]. Peter Wurman et.al. [17] developed an internet-based platform for price-based negotiation - the Michigan Internet AuctionBot. It was designed to serve as an auction server for humans as well as software agents. But nevertheless auction protocols are one-to-many negotiations and are out of scope of the WS-Agreement Negotiation model. Auction protocols require alternative negotiation approaches.

2.2.2 Auctioning and Bidding

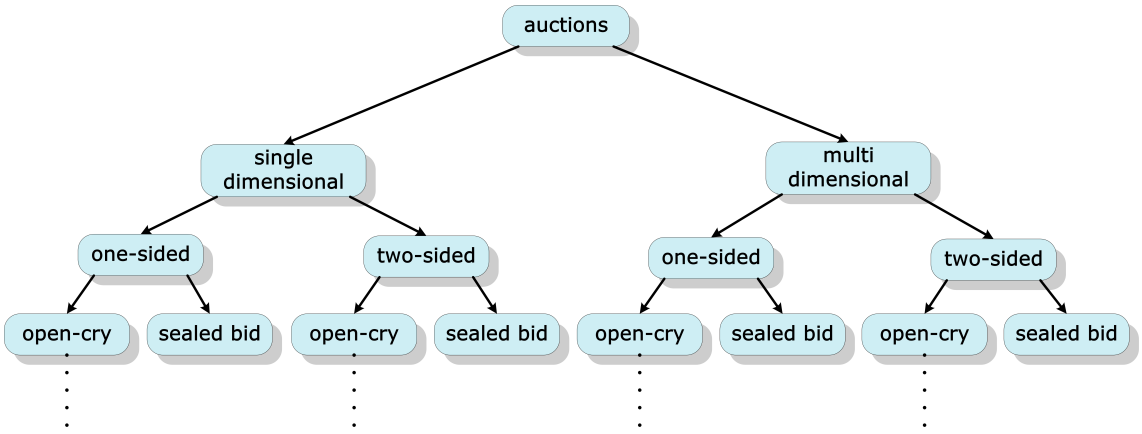


Figure 2.2: *A Zoology Categorization of Auctions*

Auctions and biding are complex negotiation processes [16]. Peter Wurman et.al. [17] developed an internet-based platform for price-based negotiation - the Michigan Internet AuctionBot . It was designed to serve as an auction server for humans as well as software agents.

Unfortunately, this auction mechanism focuses on one negotiated issue: the price. They extend their taxonomy to include multi-dimensional auctions in a followup paper [18]. A comprehensive overview and in-depth description of all kind of auctions are presented by Simons Parson, Juan A. Rodrguez-Aguilar and Mark Klein . Parson et.al. have extended the taxonomy of auctions from Yoav Shoham [19] and is depicted in figure 2.2. The classification of features of auctions are derived from Parson.

- Auctions can be single-dimensional, or multi-dimensional.
- Auctions can be one-sided or two-sided.
- Auctions can be open-cry or sealed-bid.
- Auctions can be first-price or kth-price.
- Auctions can be single-unit or multi-unit.

- Auctions can be single-item or multi-item (combinatorial).

Auctions considering only one aspect within a bid, such as the price are called *single dimensional* auctions.

Contrary to *single-dimensional*, *multi-dimensional* auctions consider several aspects like quality and delivery date within a bid..

In a *one-sided* auction only one participant submits a bid and the auctioneer has to decide which one is the winning bid.

During *two-sided* auctions, buyers as well as sellers submit bids and the auctioneer matches buyers and sellers.

In a *sealed-bid* auction only the auctioneer has access to the bids, whereas in a open-cry auction every bidder has access to the bids.

In a *first-price* auction, the winning bidder pays the price of the winning bid in a kth-price auction, the kth ranked bid. The second-price auction is the most familiar because it eliminates dominant bidders.

Single auctions deal with one good (or a bundle of it) and in *multi-unit* auctions several units of a good can be sold.

In a *Sealed first-price auction* all bidders submit sealed bids.

Participants in an *English auction* bid against each other by beating the previous bids.

Dutch auctions start with the highest price which is lowered until a buyer submits a bid. The first bidder is the winning bidder.

In *Reverse auctions* the roles of the buyer and seller are reversed which means that the sellers place offers. The buyer selects the offer with the lowest price.

The goal of the *Vickrey auction* is to find a price which represents the market value. The bidder who submits the highest bid wins but pays the amount of the highest non-winning bid and an increment. The disadvantage of this auction is that other bidders who have no reasonable chance to win can influence the second bid by submitting bits which exceed their willingness to pay [20].

2.3 Self Governing Infrastructures and Autonomous Systems

In this section we give an introduction to the basic principles of self governing infrastructures and autonomous systems. We use these principles for designing an autonomous negotiation and re-negotiation framework [21] [22]. The main goal of both involved parties is to create and operate SLAs with a minimum of human interaction, but also to negotiate and agree upon legally binding electronic contracts. Balancing these objectives is a non trivial task. Both of the parties, consumer and provider, have their own business rules and a specific business strategy to make their decisions. Automated service negotiation is defined as the process of establishing business relations between service providers and consumers [23, 24, 25, 26, 1, 10].

2.3.1 Concept of an Autonomic Manager

The concept of an autonomic manager for self governing systems depicted in figure 2.3 lists four states:

- Monitor
- Analyze
- Plan
- Execute

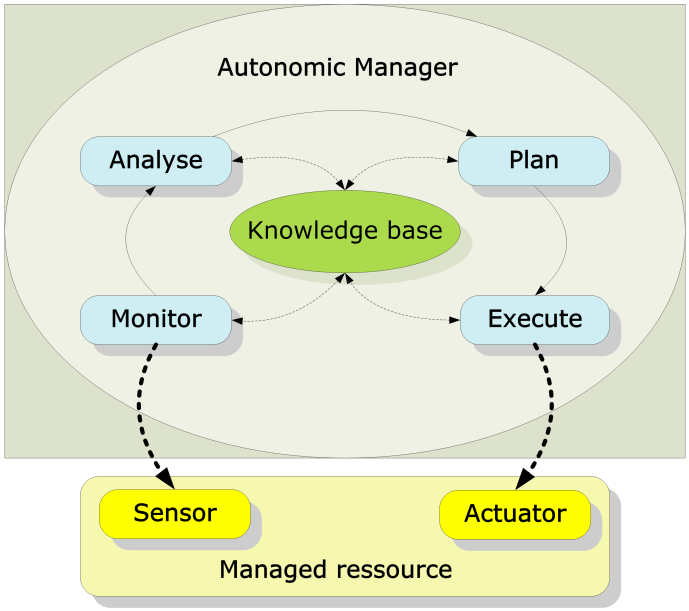


Figure 2.3: *Concept of an Autonomic Manager*

The abbreviation of these actions is the so-called *MAPE*. All actions taken are based on a knowledge base. Sensors for the managed resource are necessary for monitoring and actuators are used for changing the managed resources.

2.4 Business Rules

Many authors [27, 28, 29] and [30] cope with business rules. Unfortunately, each of these authors provides different definitions. Boyer and Mili [31] summarize several definitions by outlining two characteristics of business rules.

- **Business rules are about business.**
  A company's business motivation is the driver for creating business rules. For example, this motivation can aim at maximizing profit or minimizing risks. Business rules are formulated to attain these aims [31]. Business rules may not only represent the organization's business motivation but also legislation, regulations, external standards and best practices [29].

- **Business rules focus on structure and behavior.**
  Boyer and Mili [31] distinguish between two types of rules. Structure rules describe the structure of a business, e.g. *"A sale consists of a buyer, a product and a bill."*. Behavior rules describe the way how business reacts in response to an event, e.g. *"If loan is requested then liabilities have to be checked."*

An insurance company will have rules coping with the insurance premium calculation whereas a bank will have rules for granting loans. A simple rule of an insurance company may be *"IF a customer has an income less than 1000$, THEN the credit enquiry is refused"*. This example shows that business rules can be described by simple statements building upon *"IF-THEN"* structures. Each company has thousands of such simple business rules. However, they may not be documented explicitly or they are hidden in software. Witt [29] emphasizes that the rules have to be accessible for all stakeholder who are influenced by the rules. Furthermore it must be possible to update the rules in an adequate time. Rules in natural language are easy to understand by the business users.

However, natural language rules may be ambiguous and the same rules can be formulated in many different ways. For clear natural language rules you have to constrain the vocabulary as well as the syntax. In order to standardize the format and the business terminology, the Object Management Group published the Semantics of Business Vocabulary and Business Rules (SBVR) as a standard for defining business rules in natural language. A SBVR vocabulary consists of terms and facts, whereas the facts represent the relationship between terms. Based on defined terms and facts, rules can be formulated complying to the SBVR by defined fact models. Fact models are used to standardise vocabulary by defining the terminology used within rules [29].

Business rules are executed within business processes by both, IT-systems and humans [31]. Within business processes, business rules represent the decision logic [32]. In order to automate business processes by IT-systems, business rules have to be defined in a computer readable way. We distinguish between two types of business rules:

- **Operative Rules** define the actions triggered by an event, e.g.
  R1: Each service must *specify* a start time and an end time.
  R2: The service start time *specified* in each service must *be not earlier than* the service request time.

- **Structural Rules** define the way of how various constructs are defined by the organization, e.g.
  R3: A premium customer *is* defined as a customer whose total revenue *is greater than* 50000$ [1] in the last 5 years.
  R4: A premium discount *is* by definition exactly 20 percentage of each service price.

Typical tools for handling business rules which origin from the semantic web initiative are the following:

- **Rules Interchange Format.** The Rules Interchange Format (RIF) is a standard published by the W3C. Rules based on the RIF are formulated in XML. The W3C distinguishes between three Rules Interchange Format dialects: core dialect, basic logic dialect and production rules dialect. The purpose of the RIF is to act as an interchange format which allows conversation between different rule languages. RIF is a sub-language of RuleML [33].

- **RuleML.** The Rule Markup Language Initiative is an organization aimed at standardizing the different rule markup languages by providing specifications for XML based rule languages. RuleML [34] is a rule family which encompasses a rule modeling language as well as a rule meta logic. RuleML has also contributed to the SWRL. RuleML itself is a markup language which allows to represent forward and backward rules in XML. The difference between forward and backward rules will be explained in another section.

- **SWRL.** The Semantic Web Rule Language (SWRL) was published by the W3C too. The SWRLis based on XML and interleaved with OWL.

- **SPIN.** The SPARQL [35] Inferencing Notation (SPIN) rules were published by the W3C and are built upon the query language SPARQL. SPARQL rules allow to calculate property values based on other properties, check constraints, perform data validation and isolate a set of rules to be executed under certain conditions.

- **Jena Rules.** Jena rules [36] are based on the so-called turtle syntax which can be used to represent RDF graphs. These rules are used within the Apache Jena API which supports the W3C standards RDF, RDFS, RDFa, OWL and SPARQL.

[1] The SBVR convention is to mark quantities, dates and times using a double underline

Usually these rules are executed and triggered during business processes.

2.4.1 Business Processes

A business process describes a cross departmental series of steps to produce valuable output for the customer. A process could, for example, describe the handling of a customer order within a company [37] [38]. In the following sections we provide a description of the anatomy of a process. As [39] shows, each process has five components:

- **Trigger.** A trigger kick-starts a process. The trigger of the order process could be a customer placing an order.
- **Input.** Inputs are resources needed for executing the process. This could, for example, be an order document.
- **Enabler.** Enablers represent the necessary equipment for the process execution like a software system for processing an order.
- **Guide.** A guide points out how to execute a process.
- **Output.** As processes aim at customer satisfaction the result of a process is something valuable for the customer.

As already mentioned, business processes are executed by both IT systems and humans. In order to automatize business processes by IT systems the process has to be transformed into a workflow, i.e. a workflow describes how to execute a process. Hence, the business logic, which can be represented by business rules, has to be accessible for IT systems. Usually this is done by mapping the business logic into software code. Unfortunately, business users who are responsible for maintaining and creating business rules are not able to handle business rules stored in software. Thus, they have to communicate with IT specialists who implement the rules in software code. Not only the creation of new rules but also the change requests of already existing rules require the skills of an IT specialist. Typically, the IT specialist may not know details about a rule's business domain whereas the business user may not know anything about software development. Hence misunderstandings can occur and consequently this approach can be considered as being error prone [40]. Further problems arise when rules are hidden in several code paths instead of a central repository [40]. Rules stored this way are hard to manage and violate many principles which rules should comply [27]: Rules should

- ... be managed directly by the people who have the essential knowledge.
- ... be put down in writing and published.
- ... be written in natural language.
- ... be independent of workflows.
- ... be based upon facts. The facts should be based on concepts.
- ... control the behaviour.
- ... be driven by the important business factors.
- ... be manageable by authorized people.
- ... be handled from a single source.
- ... be administrated.

Rules managed by Business Rules Management Systems are able to comply with these principles.

Table 2.1: *Simplified decision table*

Storage capacity (Tera Byte)	Discount (%)
1,3	0
4,7	2
8,10	5

2.4.2 Business Rules Management Systems

The prominent principle published by [27] reinforces, that business users must be able to formulate the business rules on their own. Generally, there are four ways of how business users can create business rules [41]:

1. **Programming Language.** Business user can use the modelling language UML or programming languages like Java to define business rules. However, usually business users do not know theses languages.

2. **New language.** Business users can write their rules in a novel language similar to a natural language which is readable by both, IT systems and humans.

3. **User interfaces.** User interfaces can be used to support business people managing rules.

4. **Restrict natural language.** The natural language approach could be limited to a set of vocabularies sufficient to create rules for a special business domain.

All proposed possibilities, besides the first one, require the definition of a vocabulary or ontology which can be processed and interpreted by both, humans and IT systems. An exact definition of a vocabulary or ontology prevents ambiguities of terms [41].

Business Rules Management Systems (BRMS) storing the business rules representing the business logic of a company in a repository and are assisting in the management of the rules by providing several tools and interfaces. This enables reusability, faster development, clearer auditability and consistency [41]. Many organisations offer Business Rules Management Systems [42, 43, 44]. Two very popular BRMSs are Drools and IBM WebSphere ILOG JRules: First of all, two business rules management systems (BRMS) and their way of how they capture business rules are described.

- **Drools.** Drools [45] is a comprehensive open source BRMS framework based on Java focusing on business logic integration. There are several ways to formulate rules in Drools. One method is to build decision tables in Microsoft Excel ®. Table 2.1 shows a simplified decision table for the pricing model of a cloud provider. The left column shows the storage capacity whereas the right column shows the discount. If a consumers buys, for example, 5 Terabyte, he gets a discount of 2%.

  Another way to define rules within the Drools framework is to write rules in the so-called *mvel* dialect. The rules formulated in this language look similar to source code.

- **IBM WebSphere ILOG JRules.** IBM WebSphere ILOG JRules [31] is a comprehensive commercial BRMS platform. This platform allows to create rules in different ways, too. There is a "rule studio" assisting the creation of decision tables. But there is also a mechanism to create rules by a restricted natural language.

2.4.3 Business Models

Business models have the goal to create values, build new businesses or improve existing businesses. There exists a broad range of business model definitions, e.g. by Alt and Zimmermann [46]: *"For the business model discussion [...] we will distinguish six generic elements of a business model: Mission, Structure, Process, Revenues, Legal Issues, Technology"*. Gordijn [47] gives a definition which includes activities for creating values and covers the whole life cycle of products or services: *"We define e-Business models as conceptual models that show how a network of actors (a value constellation) creates, exchanges and consumes objects of value by performing value adding activities."* Osterwalder et. al. [48] give a more general and comprehensive definition of business models: *"A business model describes the rationale of how an organization creates, delivers, and captures value. The business model is like a blueprint for a strategy to be implemented through organizational structures, processes, and systems"*.

For the development of our framework we used Osterwalder's business model. We chose his definition of business models to show how business rules can be derived from a business model expressed in natural language. He describes a business model by means of nine basic building blocks that show the logic of how a company intends to make money. These nine blocks cover four main areas of a business: customers, offer, infrastructure and financial viability.

- **Customer Segments** are the most important parts of each business model. Different groups of customers should be handled in different ways. Organization and their processes must consider these groups.

- **Value Propositions** describe the specific products and services for a specific customer segment.

- **Channels** include the description of how the company communicates to reach the customer segments. The purpose of the channels is to deliver value proposition and is the interface to the customer.

- **Customer Relationships** describe the company's type of relationship to each customer segment, e.g. personal or automated or a mixture of them.

- **Revenue Streams** hold the representation of all cash streams to each customer segment. Different pricing mechanisms can be used for each revenue stream. Osterwalder distinguishes between two types of revenue streams: revenues resulting from one-time customer payments or from ongoing payments.

- **Key Resources** create value propositions in each company. The resources can be physical, financial, intellectual or human depending on the type of business.

- **Key Activities** should be described to make and keep the business model work, e.g. for a hardware manufacturer supply chain management is the key activity.

- **Key Partnerships** represent the description of the network of a company. These partnerships reach from a buyer-seller partnership to strategic alliances.

- **Cost Structure** describes the most important cost during operation of the business model. That means all cost of creating, generating and maintaining value and customer relationships. This cost can be derived directly from the previous building blocks such as key resources, key activities and key partnerships.

2.5 Traditional Economic Principles

In this section we give a short introduction to traditional economic fundamentals and methods described in the standard economic literature [49, 50, 51]. This section is mainly derived from [49] and [50].

Traditional economic science covers the following scopes:

- Operating Production Factors,
- Production,
- Sales Theory and
- Investment and Finance

2.5.1 Operating Production Factors

Producing a good requires to combine production factors. Production factors are e.g. manpower, machine employment, materials, energy, auxiliary materials and so on. Each production factor is named $R_1 \dots R_n$.

The theory of production is concerned with the functional dependencies between the amount of production factors and the amount of produced goods.

These functional dependencies can be described in production functions. The goal is to find regularities between input and output of the factors under strongly simplified assumptions [52, 49]. Based on these simplified assumptions, cost theory tries to find the functional dependencies between the amount of production factors used and the resulted cost. The amount of output m is a function of the quantity of the input factors $r_1 \cdots r_n$. $R_1 \dots R_n$ are the production factors and $r_1 \cdots r_n$ are the amount of input.

$$m = f_1(r_1, r_2 \dots r_n) \tag{2.1}$$

The amount of output units is directly significant for the revenue. Therefore the revenue E is a function of the production factors too.

$$E = f_2(r_1, r_2 \dots r_n) \tag{2.2}$$

Depending on the production process complex production functions can exist. The relationship between the production factors can be reduced to two general cases.

1. The production factors can be substituted by each other (Substitutability).
2. The production factors can only be used in the same proportion (Limitationality).

Substitutability.

Figure 2.4 depicts two production factors r_1 and r_2 . The area of the rectangle between $\overline{r_1}$ and $\overline{r_2}$ represents all possible combinations of r_1 and r_2. An amount of revenue E can be assigned to each point in the area. If we consider the rectangle as plane we can draw the corresponding revenue E vertically to each point. This leads to a revenue mountain. For better understanding we introduce the term *marginal revenue*.

The *marginal revenue* is the increment of amount of the revenue that can be achieved with a (theoretically) infinite increase of the usage of a factor. Typical forms of such revenue mountains exist (see also figures 2.5 and 2.6):

- Constant marginal revenue,
- Increasing marginal revenue,

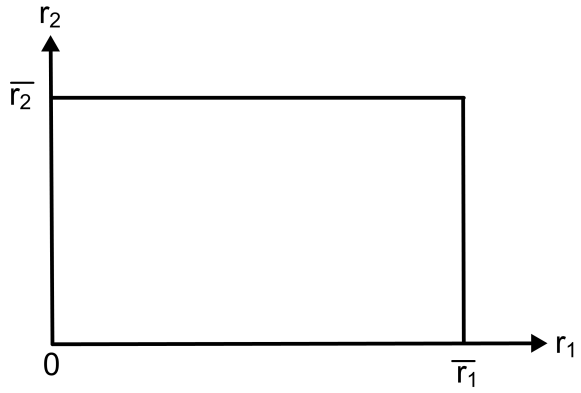


Figure 2.4: *Substitutability of Production Factors*

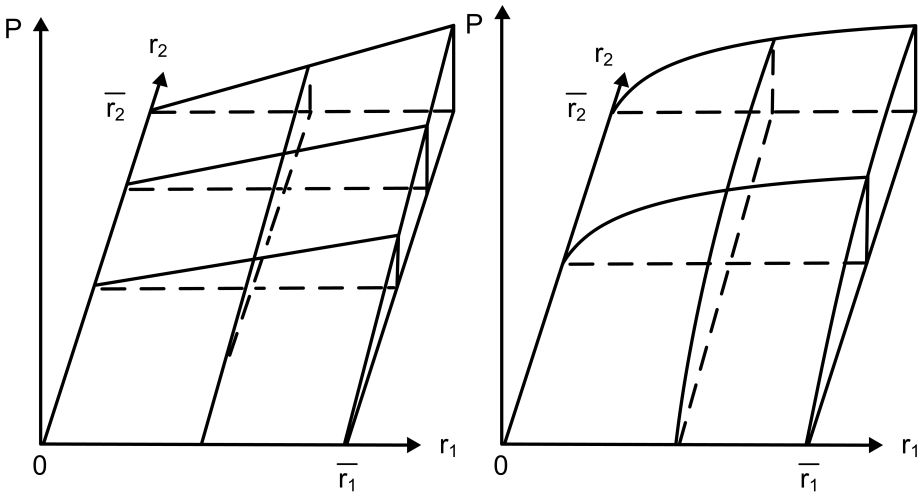


Figure 2.5: *Constant Marginal Revenue (left), decreasing Marginal Revenue (right)*

- Decreasing marginal revenue, and
- Increasing and then decreasing marginal revenue

Limitationality.

Limitationality exists if the production process does not allow to substitute one production factor for another. These factors are also called limited production factors.

If we have production factors that can be substituted for each other until the amount of one factor is zero, we have the case of *alternative substitution*. We can use the production factors alternatively. If we have a combination process of using R_1 and R_2 with a minimum amount of one of the factors, we call it *limited substitution* (see figure 2.7).

2.5.2 Production

Relationship between Production- and Cost-Functions.

Cost are the priced amount of production factors. As defined in equation 2.2 the revenue is a function of the amount of used production factors. The functional dependence of overall revenue on factor input is defined by the cost function:

$$C = f(m) \tag{2.3}$$

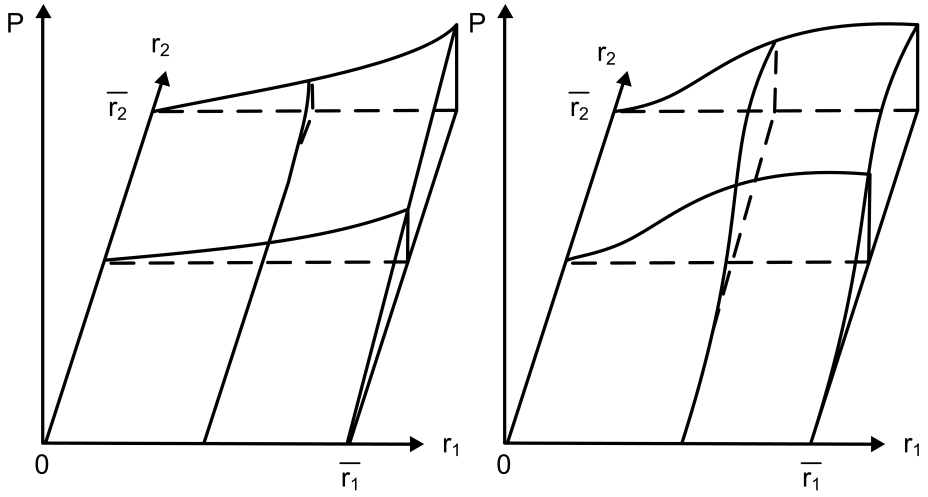


Figure 2.6: *Increasing Marginal Revenue (left), increasing and decreasing Marginal Revenue (right)*

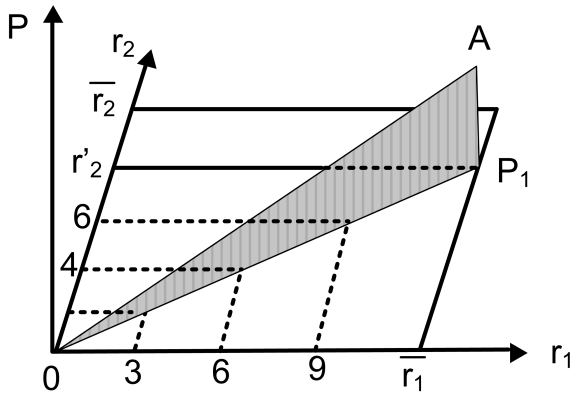


Figure 2.7: *Limited substitution*

From equation 2.2 we can derive the total revenue if we rate all production factors with money.

$$E_t = f(r_1 \cdot p_{r1}, r_2 \cdot p_{r2}, \cdots, r_n \cdot p_{rn}) \tag{2.4}$$

or in term of cost of amount production factors

$$E_t = f(C_1, C_2, \cdots C_n) \tag{2.5}$$

$$E_t = f(C) \tag{2.6}$$

This still remains a production function, but usually we want to know the cost for given total revenue. Thus, we invert the function

$$C = f(E_t) \tag{2.7}$$

Basic Cost Concepts.

The total cost within a period of production consist of a variety of types of cost. Considering the type of cost dependent on the activity rate q (the amount of production output) we distinguish between *fixed cost* C_f and *variable cost* C_v as depicted in figure 2.8.

Fixed cost C_f means that the cost do not react if we change the activity rate q. Such cost are e.g. depreciation of machines, rent, interests or personal cost of common people.

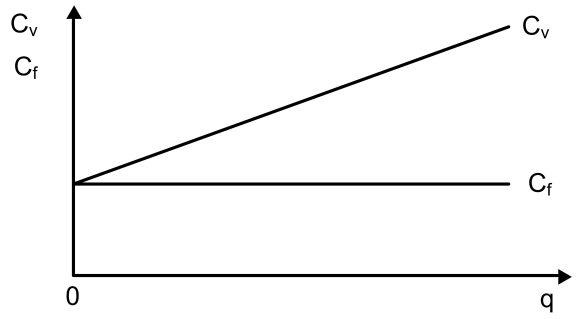


Figure 2.8: *Fixed and Variable Cost*

On the other hand variable cost C_v are cost that react during the change of the activity rate. Variable cost can be subdivided into proportional, progressive and regressive cost.

Average Cost Function.

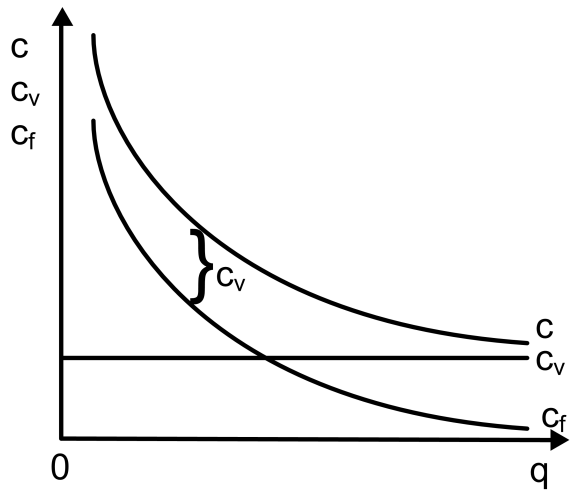


Figure 2.9: *Average cost per quantity without stepped cost*

The function for the average cost is the result of adding the two cost functions for fixed and variable cost $c_f = f(m)$ and $c_v = f(m)$. The average cost function is depicted in figure 2.9. The variable average cost are constant. The asymptotic decreasing fixed cost per piece are shifted with the amount of the fixed cost resulting in average cost function

Importance of the Decision Period.

To decide which cost are fixed or variable it is necessary to determine their impact on the amount of the output. For example, there are cost of regressive or progressive behaviour with variation of the amount of output.

We distinguish between:

- Cost are relevant to the decision
- Cost are irrelevant to the decision

The length of the time period has a strong impact on the decision on fixed and variable cost. The longer the time period, all types of cost are variable.

Importance of Subdivision of Production Factors.

Another reason for deciding if cost are fixed or variable is the fact that some production factors can not be subdivided. The theoretical assumption in this section that all production factors can be subdivided to an unlimited degree, does not hold in reality. That means that variable cost are not continuous, they have steps.

2.5.3 Sales Theory

Sale is the last step in a production process, that means, selling of goods and services. In newer literature sale is often called marketing.

Market Research as Planning Tool for Sales

Market research is the systematic scientific method for collecting information of the market.

It is important to understand the behavior and relationship of all factors in the market to have all information to determine the right decisions in production. We have to analyze the demand, the competition and the way of distribution.

2.5.3.1 Price Policy

Price policy is the basis for all decisions to determine prices of all parts of the production process and sales process. The fundamentals of this policy is the knowledge of the market. A market is defined as the interaction of demand and supply. Usually, a market can be divided into the following three structural characteristics.

- Only one seller respective one buyer
- Several sellers respective buyers
- A great number of sellers respective buyers

We can build all combinations of these three basic characteristics. Another way to describe markets are by their behavior. The theory of the behavior of markets is mainly divided into two types by E.Schneider [53]:

- Monopolistic behavior: The firm expects that its sale is only affected by the behavior of the buyers and not by its competitors.
- Competitor bounded: The firm expects that its sale is affected by the behavior of the buyers and also by its competitors.

Competitor bounded can be subdivided into atomistic and oligopolistic behavior.

Atomistic means that the firm expects that its sale also depends on other sellers but it does not expect that its price adjustment leads to a changing of the prices of the competitors.

Oligopolistic behavior assumes that every price adjustment causes a reaction of the competitors.

2.5.3.2 Price Elasticity of the Demand

Typically, a seller (firm) tries to maximize its profit. Profit is the difference between total revenue minus total cost. To increase the total revenue it is necessary to decrease the total cost or to increase the price. But if the price changes the amount of sales (number pieces sold) is also changed. The typical demand curve is depicted in figure 2.10. The price elasticity of the demand is defined as the ratio between relative variation of the amount

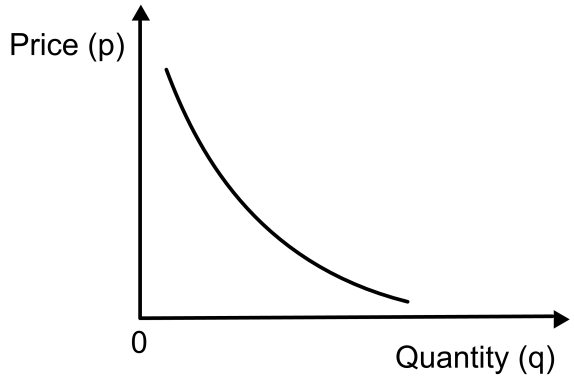


Figure 2.10: *Demand Curve*

	A([55])	**B**([57])	**C**([59])
	user perspective	*user perspective*	*provider perspective*
cost types	Storage IO request Computing power Time consumed	Network Storage Administration CPU	Server Infrastructure Power draw Network

Table 2.2: *Cloud Cost Models Overview*

to the relative variation of the price alteration. The price elasticity coefficient e is defined as:

$$e = -\frac{\Delta m/m}{\Delta p/p} \tag{2.8}$$

If the coefficient is $e < 1$ the demand is elastic, if $e > 1$ the demand is inelastic.

2.6 Cloud Cost Models

In this section we introduce the current state of research regarding cloud cost models.

Current cost models for cloud computing has so far been developed that makes a non specific distinction between variable and fixed cost [54, 55, 56, 57, 58, 59, 60].

Table 2.2 shows the cost types analyzed in [55][57][59] and [61]. Additionally, the cost types of the introduced cloud cost model are listed. Non of the reviewed research papers distinguishes between fixed and variable costs. Moreover, all research lack in a generic formula representing the cloud's costs.

The cloud cost calculation from [55] focuses on cloud computing consumers and is based on Amazon's pricing model. Amazon offers two different pricing models [55].

1. In the first pricing model, the consumer has to pay for each hour an instance is running. You can use instances with different memory and computing power.

2. The second pricing model considers the consumed Input/Output transactions as well as the memory consumption.

Just as [55], [57] neglects the distinction between fixed and variable cost and focuses on cost optimization for the cloud consumers. Hence both, [55] and [57] disregard the cost structure of a cloud provider. [56] uses Amazon's pricing model too. Both, [55] and [56]

do not cope with the cost structure of a cloud provider. [59] concentrates on cloud data centers. However, neither fixed nor variable cost are considered.

Cost models like [60] have parameters such as CPU and I/O, but also make no distinction between fixed and variable cost.

Cost models that consider variable cost are better suited for the economy situation today. This distinction also allows to implement more accurate business strategies.

Traditional business models are largely based on fixed cost operating models. These operating models are driven of large capital investments to leverage economies of scale to produce incremental profit in the case of increasing volume. This results in spreading operating cost to larger and larger units sold. The prediction of the products‘ sales is stable enough to allow companies to allocate labor and capital in order to support demand.

Variations of demand can only be compensated for low frequencies. In the past typical product life cycles were measured in years, therefore this kind of operating models could be used. However, product life cycles are shortened from years to months. Also rapidly evolving consumer preferences in global markets require more flexible cost models to give quick answers to changing demands. There is a specific need for cloud computing today, where only dynamic, adaptive and precise business decisions allow for economic success of this new technology trend.

2.7 Quality Measures for Computational Resource and Service Markets

In this section we describe quality measures of service markets. In current research most of provided definitions are suitable for finance or electronic trading markets [62, 63]. Only few approaches exist to define performance indicators for service markets [64]. We describe how quality measures are defined in electronic markets and how these measures can be adopted to service markets.

2.7.1 Influence Factors

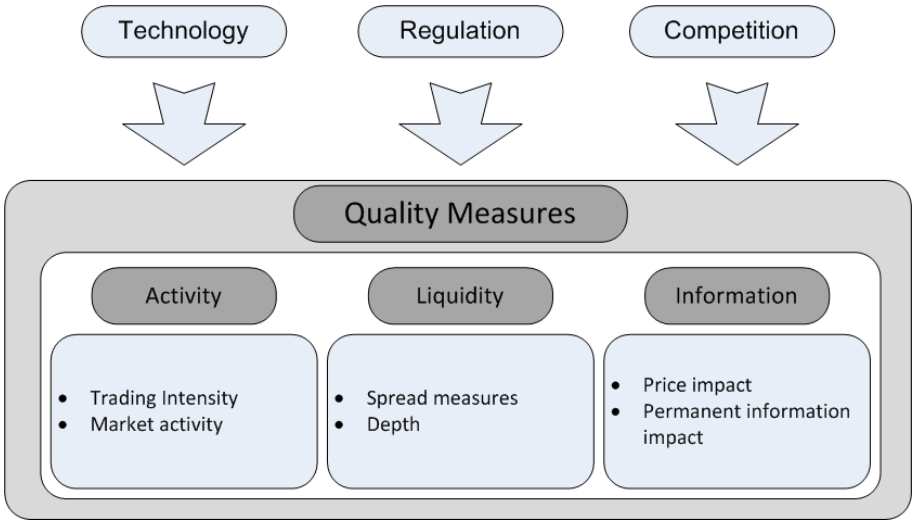


Figure 2.11: *Influence Factors on Quality Measures*

In finance and electronic trading markets we find two main influence factors [65, 66]:

- External factors
- Internal factors

These influence factors are depicted in figure 2.11. External factors like technology, regulation and competition directly influence the market quality. Business structure, IT systems and the market microstructure are considered as important external influence factor. The business structure is defined by the type of customers and products. The influence of IT systems is determined through the trading system and the degree of automation. Execution systems and their transparency define the market micro structure.

The most important market quality measures of financial and electronic markets are:

- Activity
- Liquidity
- Information

Activity can be measured by using the number of trades (*Trade Count*), the trading volume (*Turnover*), or the average trade size (*Trade Size*) [63, 65]. Activity measures the trading intensity.

Harris defined liquidity to be a better measure for the attractiveness of a market for traders than activity. Harris defined liquidity as

"the ability to trade large size quickly, at low cost, when you want to trade" [67]

2.7.2 Speed and Spread Measures

Spread measures are very common in finance business. Spread measures of liquidity are calculated on a daily basis and use basis points as unit.

In finance literature *Quoted Spread*, *Effective Spreads* and *Realized Spreads* are considered as the most important spread measures [63].

$$QuotedSpread_{i,t} = (Ask_{i,t} - Bid_{i,t})/(2 \cdot Mid_{i,t}) \tag{2.9}$$

where $Ask_{i,t}$ is the ask price for a stock i at time t, $Bid_{i,t}$ the respective bid price, and $Mid_{i,t}$ the mid quote.

The effective spread represents the actual transaction cost when an incoming market order is executed against a limit order.

$$EffectiveSpread_{i,t} = D_{i,t} \cdot ((Price_{i,t} - Mid_{i,t})/Mid_{i,t}) \tag{2.10}$$

where $Price_{i,t}$ is the execution price and $D_{i,t}$ the trade direction (-1 for market sell and +1 for market buy).

An effective spread has two parts. These two parts are the realized spread (liquidity suppliers revenue) and the price impact.

$$RealizedSpread_{i,t} = D_{i,t} \cdot ((Price_{i,t} - Mid_{i,t+x})/Mid_{i,t}) \tag{2.11}$$

where x is a time interval between 5 and 30 minutes.

Spread measures account for the width of liquidity. Depth on the other hand is the other dimension of liquidity. Depth measures the quoted volume of limit orders in the order book at a given price [65]. Depth at the best bid is therefore:

$$Depth_{i,t} = (Vol \cdot Bid_{i,t} + Vol \cdot Ask_{i,t})/2 \tag{2.12}$$

where $Vol \cdot Bid_{i,t}$ and $Vol \cdot Ask_{i,t}$

To approximate the information content of a trade, the price adjustment after a trade can be used:

$$PriceImpact_{i,t} = D_{i,t} \cdot ((Mid_{i,t+x} - Mid_{i,t})/Mid_{i,t}) \tag{2.13}$$

2.7.3 Liquidity Measures

In Theissen [62] liquidity is defined as follows:

> "A market is considered to be liquid when assets can be bought or sold rapidly and at a price close to the equilibrium value. This definition incorporates two dimensions, time and cost." [62]

The application of quality measures of finance markets to computational resource markets can not be done directly. A comprehensive definition covering all aspects of markets is not available. The structure and behavior of service markets differs in several characteristics. One of these differences is the kind of the trading goods. The ad hoc arising questions are:

- What are the trading goods? The specific service (Processing Power, Disk Space, Memory, VM,..) or the whole SLA template/ agreement?
- What is the trading volume? The amount of services? Number of SLA's ? The price of an SLA?

2.7.4 Pareto Optimality

In Rosenschein [68] definitions and design conventions for automated negotiation among computers are presented. Rosenschein defines constraints for Pareto Optimality in automated negotiation within an agent based automated negotiation system.

They define Pareto Optimality

> "*that no agent could derive more from a different agreement, without some other agent deriving less from that alternate agreement*" [68]

It is also important to define that not a *Global Optimality* would be achieved by the agents. That means that not the sum of all involved agents in a system are maximized. Rosenschein speaks of *self-motivated agents* that only care about their own utilities, not the sum of the system utilities. They assume that both agents are utility maximizers. Decisions made by the two agents are completely independent. The agents pay no attention to what the other agents are getting.

Rosenschein defines

> that a deal δ *dominates* δ' if δ is better for at least one agent and not worse for the other. Neither agent prefers δ' over δ, and at least one of the agents prefers δ over δ'.
>
> Definition for pure deals δ and δ'
>
> - We will say that δ *dominates* δ', and write $\delta \succ \delta'$, if and only if $(Utility_1(\delta), Utility_2(\delta)) \succ (Utility_1(\delta'), Utility_2(\delta'))$
> - We will say that δ *weakly dominates* δ', and write $\delta \succeq \delta'$, if and only if $(Utility_1(\delta), Utility_2(\delta)) \succeq (Utility_1(\delta'), Utility_2(\delta'))$
> - We will say that δ *is equivalent to* δ', and write $\delta \equiv \delta'$, if $\forall k(Utility_k(\delta) = Utility_k(\delta'))$
>
> *"A deal is pareto optimal if there does not exist another deal that dominates it. A pareto optimal deal cannot be improved upon for one agent without lowering the other agent's utility from the deal."* [68]

2.7.5 Quality Measurement in Service Negotiation

In this thesis we use

"How close is the result of a negotiation to a Pareto Optimal solution"

as a suitable quality measurement in service negotiation. The reason is that this measurement can be used for bi-lateral and multilateral negotiations. In future research we want to consider time and trade volume to further improve quality measurements.

3 A Business Rule Ontology

In this chapter we describe our newly developed generic business model ontology. The ontology is used for formally describing a specific business. The novelty of our ontology is the great flexibility to define the characteristics of any kind of businesses. In this ontology a mapping of business rules expressed in a natural language and its corresponding constraints is included.

3.1 Generic Business Model Ontology

Figure 3.1 illustrates our generic business model ontology as UML class diagram. Each business model consists of building blocks similar to Osterwalder's framework and constrain data [48]. Our business model ontology contains 1..n building blocks. That means the resulting business model is not limited and can easily be extended or shrunk to fit the participant's needs.

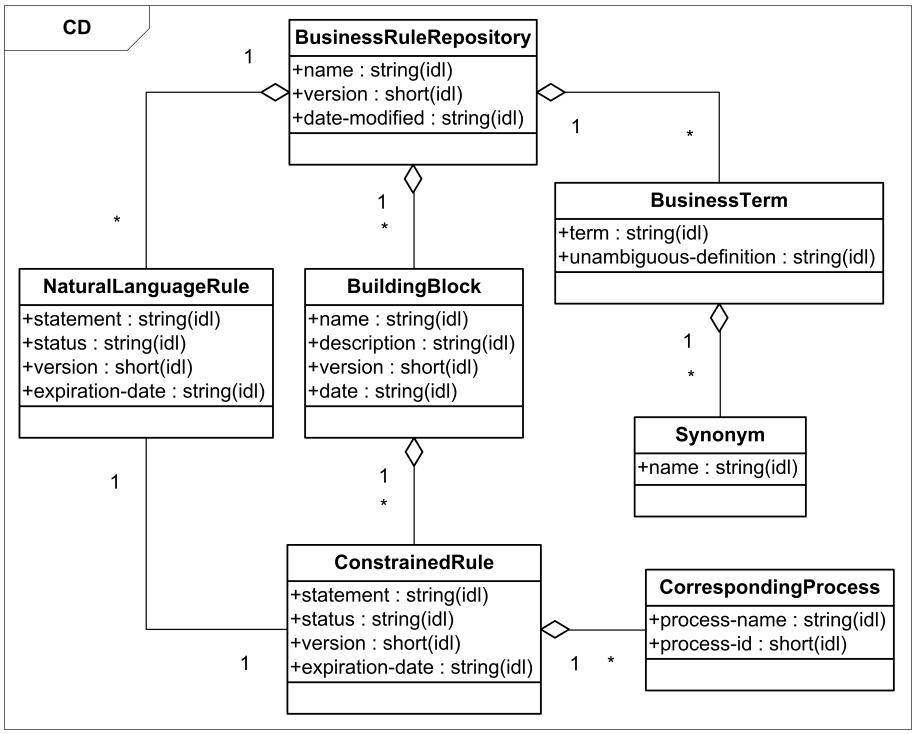


Figure 3.1: *Generic business model ontology*

The business rule repository ontology consists of the following elements:

- **Business Rule Repository** is the root element and has the attribute *name* and *date-modified*. The business rule repository has 1..n building blocks, 1..m business rules expressed in natural language and 1..p business term definitions.
- **Building Block** holds the information about the core description and characteristics of the company's business. The attribute *description* is used for defining the building

block. Each building block has 1..n associated constraint rules. The constraint rules are derived from the business rules described in natural language.

- **Natural Language Rule** defines the business rules expressed in natural language. It is important to have the attribute *version* and *date-modified* for supporting the history management of business rules.
- **Business Terms** describe each term used in the constrained rules.
- **Synonyms** are only used for better understanding defined business terms, so that all participants involved have a clear understanding about the meaning of each term. Each business term can have one or more synonyms.
- **Constrained Rule** defines the business rule derived from the natural language rule and is used in the processes instead of the natural language rule.
- **Corresponding Process** holds the information in which process a rule is used. Each rule is used in one or more processes.

By using this generic ontology a company is able to build its own specific business rule repository. The process for generating a business rule repository is depicted in algorithm 1:

Algorithm 1: Generating Business Rule Repository

```
Data: Business Rules defined in Natural Language 1..R
Result: Constrained Business Model Repository
build constrains for the business model;
define appropriate building blocks 1..B;
for each business rule R do
    for each building block B do
        if business rule R is related to building block B then
            translate business rule R from natural language to constrained language;
            store translated rule in building block B;
            fill in meta-data for rule R (status, version, expiration date);
            break;
```

3.2 Realization by Semantic Web Tools

The W3C has published the XML based standards Ontological Web Language and the Resource Description Framework for defining ontologies.These two standards origin from the semantic web technology development. These technologies are platform independent, exchangeable, comprehensive and widely-accepted. Thus, these technologies can be used for business applications utilizing rules. For the management of the knowledge base SPARQL [69] and Jena [70] can be used.

Summing up, RDF and OWL are used for ontologies, whereas SPARQL is used to query these ontologies, and Jena is used to execute rules from the knowledge base.

3.2.1 Resource Description Framework

The Resource Description Framework is a W3C standard for building ontologies [71]. Ontologies are knowledge bases which contain knowledge of a specific domain. In RDF, knowledge is represented by so-called resources and their relationship to other resources. A resource represents an object of the real-world like a car or a book. RDF as well as OWL are technologies for creating ontologies.

3.2.2 Ontological Web Language

RDF can not be used to describe complex ontologies. Consequently the Ontological Web Language (OWL), which extends RDF Schema, was developed to build more complex ontologies. The W3C published three OWL standards which are called OWL Full, OWL DL and OWL Lite. OWL Full is the biggest standard which encompasses both, OWL DL and OWL Lite. OWL Lite is the smallest standard and is part of OWL Full and OWL DL [3].

3.2.3 SPARQL

RDF and OWL are used to store knowledge. SPARQL Protocol And RDF Query Language (SPARQL) is a language for querying this knowledge. Just as RDF and OWL, SPARQL was published as standard by the W3C [69]. SPARQL queries use the turtle data format and typically consist of three sections.

- **PREFIX.** The prefix section is used to determine namespaces.
- **SELECT.** The select sector constraints the result of the SPARQL query. Here you can state the variables which should be resulted from the query.
- **WHERE.** The where section consists of several statements which represent triples. The structure of the statements is identical with the statements in RDF. Additionally, you can use variables as placeholders in SPARQL within statements. A variable starts with a question mark and can be used as subject, predicate and object. The where sector in listing 3.1 contains three different variables. The first line of the where sector defines that the variable *?loyaltyCard* contains everything that has a *belongsTo* relationship to a *person*. From each of these things, the expiration date and the category are stored into the variables *?date* and *?category*. As the select section shows, these two variables are returned. So this simple SPARQL query returns all categories and expiration dates of all loyalty cards.

Listing 3.1: *SPARQL query*

```
PREFIX uv: <http://univie.ac.at>
SELECT ?date ?category
WHERE {
  ?loyaltyCard uv:belongsTo uv:Person .
  ?loyaltyCard uv:validTo ?date .
  ?loyaltyCard uv:category ?category .
}
```

3.2.4 Jena

Apache Jena is a Java based framework for building semantic web applications [70]. The framework provides an API for processing OWLs as well as RDFs and allows to query them via SPARQL. Additionally, Jena provides a rule-based inference engine which allows to execute Jena rules. Listing 3.2 shows a rule in the turtle syntax. This rule is used to determine premium customers.

Listing 3.2: *Jena example rule*

```
@prefix j.0: http://www.univie.ac.at
[premium: (?s rdf:type j.0:customer)
          (?s j.0:customer_Revenue ?c)
          greaterThan(?c,60)
          ->
          (?s rdf:type j.0:PremiumCustomer)

]
```

1. First of all, prefix definitions can be declared before the rule is defined. A prefix is initiated by the *@prefix* keyword followed by the prefix name and the URI.

2. The rule itself is defined within squared brackets. The rule may start with the rule name. In listing 3.2 the rule is called *premium*.

3. Jena supports the forward, backward and promiscuous rule mode. Forward rules have the structure T_1, T_2, T_n ->T_0. T stands for triple. The structure of the forward rule can be interpreted as if T_1, T_2, T_n are matching, T_0 is executed. Backward rules have a similar structure like T_0 <- T_1, T_2, T_n. T_0 is executed is the so-called goal. Rules are executed "backward" to see if data is available satisfying the goal. It is also possible to combine forward and backward rules which requires Jena's promiscuous mode. The rule in listing 3.2 is a backward rule as it contains the ->arrow.

4. The rule in listing 3.2 has three triples.
   T_0: ?s rdf:type j.0:PremiumCustomer
   T_1: ?s rdf:type j.0:customer
   T_2: ?s j.0:customer_Revenue ?c

   Within these triples, variables are used. In T_1, a customer is stored in variable ?s. In T_2, the revenue of the customer, which is stored in ?s, is loaded into the variable ?c. Finally this variable is used in a so-called *built-in*. A *built-in* is similar to a procedure in a programming languages. In listing 3.2 the *greater-than built-in* is used. This greater-than checks if the revenue is greater than 60. In this case T_0 is executed which means that a customer becomes a premium customer. Each customer who complies with this rule becomes a premium customer by getting the type "premium customer."

The resulting business model ontology for building the business rules repository can be used in an automated negotiation and re-negotiation framework.

4 SLA Matching Prediction Model

In this chapter we introduce the results of our research regarding a novel analytical SLA matching prediction model for forecasting matching results in the phase of SLA matching. With this prediction model we are able to calculate the SLA matching probability depending on the chosen SLA parameters.

4.1 Introduction to SLA Matching

The so-called supermarket approach is typical for service negotiation today: Consumers buy resources via SLAs from providers for a fixed price. Neither the price nor SLOs are negotiated between consumers and providers. Therefore, a consumer with a specific demand looks for providers offering the demanded SLA. Typically, the consumer subscribes to the SLA of the cheapest provider. This situation is elaborated by the following example: In table 4.1 a consumer demands a SLA encompassing three services: 10 units of *Service 1*, 55 units of *Service 2* and 42 units of *Service 3*.

The providers offer fixed SLAs, which the consumer can take or leave. As shown in the table, *Provider 1* has two offers whereas *Provider 2* has one offer. None of theses offers fulfil the consumers demand exactly. The offers are a result of the providers' business strategies or the providers' capabilities.

Table 4.1: *Consumer demand - supermarket approach*

		Provider 1		**Provider 2**
	Demand	**Offer 1**	**Offer 2**	**Offer 1**
Service 1	10	10	30	15
Service 2	55	50	70	49
Service 3	42	35	63	20

In a bazaar style situation, the providers offer service parameters as ranges. Such a situation is shown in table 4.2 which extends table 4.1: *Provider 1* merges *Offer 1* and *Offer 2*. By merging the two offers, *Provider 1* may be able to server consumers which were not satisfied with the two offers.

Instead of offering fixed prices as shown in table 4.1, providers are able to negotiate the price with the consumers considering the current workload and demand. Thus, the bazaar approach enforces flexibility which results in adequate prices.

The consumer has ranges, too: As already described, a service can be described by using service parameters. It is unlikely that a consumer demands a service with exact values. It is more likely that a consumer demands e.g. a streaming service which has a streaming rate *between* 80 and 96 kbps to comply with the proxy. So instead of fixed demand values, the consumer sets ranges representing its required lower and upper bound enabling flexibility and negotiation with the provider. Using the bazaar style as shown in table 4.2, Provider 1 is matching. Provider 2 remains non-matching.

A more detailed description for the bazaar style is provided within this thesis.

Further, a prioritization of service parameters is done by the consumer. *Service 1* has the lowest priority whereas *Service 2* has the highest priority. As we explain in section 4.4,

this prioritization can be used within a simple algorithm to find providers if the original demand cannot be supplied.

Table 4.2: *Consumer demand - bazaar approach*

	Priority	Demand	Provider 1	Provider 2
Service 1	3	9-15	10-30	15
Service 2	1	55-59	50-70	49
Service 3	2	41-43	35-63	20

Today the market situation described above is already business reality, as Amazon offers its cloud services alternatively to the typical supermarket approach.

(i) Instead of buying e.g. computing power for a fixed price, consumers can bid for computing power using the **EC2 Spot Instances** model. Based on the bids, Amazon assigns not-used computing power to the consumers which profit from cheaper prices. (ii) Using **EC2 Spot Reserved Instances** consumers can reserve computing power for instances. Reserved instances can be sold by the consumer using the Reserved Instance Marketplace.

To start a service negotiation process it is necessary for the consumer to find an appropriate provider. Thus, the consumer is interested in answers to the following questions:

- What is the probability of finding a provider offering a SLA matching my request?
- Which service parameters of the SLA, have to be adapted and to what extent, in order to find at least one matching provider?
- How many providers should I contact in order to be practically sure to find at least one matching provider?

This section presents answers to these questions by defining an analytical model forecasting the probability of finding appropriate providers for a requested SLA. We follow a twofold approach by defining a theoretical model and proving its applicability by practical simulation. Further we present an algorithm optimizing a consumer's SLA in order to practically ensure to find at least one appropriate provider. To omit redundancy, we focus our analysis on the consumer's point of view. However, all answers given can be applied to the provider as well. The justification of our proposed approach is done within a simulation framework proving our theoretical findings on practical examples.

The layout of this chapter is as follows. In the next section 4.2 we present the assumptions of our analytical forecast model and identify the distribution function for SLO interval overlaps by a simulation approach. The analytical forecast model is presented in section 4.3 followed by the negotiation interval optimization algorithm in section 4.4. To elaborate practically the analytical method a comprehensive use case and its justification by simulation is given in section 4.6. Finally, in section 4.7, the simulation prototype implementing our negotiation model is described.

4.2 Model Assumptions

Our forecast model is based on some assumptions regarding providers and consumers.

- **Provider.** The SLO parameters offered by the providers are ranges and can consequently be represented as intervals. We call such an interval *provider interval*. Some of the intervals may cover the whole market whereas others may be very small. The size and position of the ranges of the providers are independent of each other.

- **Consumer.** The consumer's requested SLO parameters are ranges and can be represented as intervals, too. Such an interval is called *consumer interval*.

Figure 4.1 illustrates a typical situation on the market: A provider offers a SLO parameter from x_1 to x_2. The consumer requests this SLO parameter from y_1 to y_2. The overlapping range for this SLO parameter reaches from y_1 to x_2. We call the difference $x_2 - y_1$ negotiating range which will be relevant for the negotiation between provider and consumer. As shown in figure 4.1, the probability of finding a provider offering a service parameter which matches the consumer's request is equal to the probability that these two intervals are overlapping. For forecasting the probability we need both, the requested

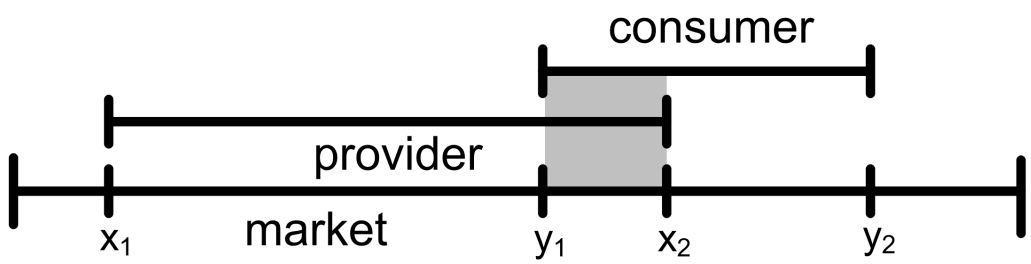


Figure 4.1: *Provider and consumer interval*

SLO parameters ranges of the consumer and the offered service parameter ranges of the provider. Unfortunately, the service parameter ranges of the providers can change quickly and no repository containing all these data in real-time is available in practice. Therefore we are forced to make assumptions about the provider ranges: In our simulation, which is described in section 4.2.1, the length as well as the position of the *provider intervals* are generated randomly. Thus, the consumer's SLO parameter ranges are covered by the providers with the same probability, independently of the ranges' position and the SLO type.

Some work on overlapping intervals was done in the past and is condensed at *mathpages.com* [72]. On this website a mathematical analysis for the probability calculation of overlapping, randomly generated, intervals is derived. However, this approach has one essential constraint: it is assumed that the intervals have exactly the same length. Regarding figure 4.1, this constraint would enforce that the consumer's and the provider's range have the same length. Obviously, this constraint violates our assumption that the provider's interval length is created randomly and may be different to the consumer's interval length. Thus, to the best of the authors' knowledge, an analysis for calculating the probability of overlapping intervals with different lengths doeas not exist.

One goal of our approach is to keep the forecast model easily adaptable regarding to changes of interval generation and modifications of interval lengths.

- The forecast should be easily adaptable in response to interval generation changes
- Changes of interval length limits should be considered by our forecast, too

Thus, we decided to build a simulation which generates *provider intervals* and checks if they overlap with *consumer intervals*. Based on the simulation's result it is possible to determine the probability distribution of overlapping intervals with different lengths. If the interval generation changes or the length limits of intervals are modified the simulation adapts respectively. In turn the modified simulation will be executed several times in order to determine the new probability distribution and consequently deliver new forecasts. The calculation details will be explained in the following sections.

4.2.1 Identification of Probability Function

We applied a simulation approach for identifying the interval overlapping probability function which is the basis for our analytical model in section 4.3. Goal of the simulation is to generate intervals and check if providers' offers matches consumers' requests. By executing this simulation with a large number of experiments, we calculate the underlying probability distribution of *provider and consumer interval* overlaps. Algorithm 2 sketches execution of the simulation. The algorithm counts the number of providers until a matching provider is found. By executing the algorithm with a large number of experiments, we get a distribution which shows how many providers are necessary to find a match. Our results and a more detailed description will be presented in section 4.2.2.

A simplified simulation flow is illustrated in figure 4.2: A consumer with a desired SLA, which consists of one SLA parameter, looks for a fitting provider in a market with 20 providers. The providers are represented as dark rectangles and their intervals are generated randomly. The numbers in the rectangles represent the provider number.

The consumer contacts one provider after the other until finds a provider whose interval is overlapping. In this case, the last provider is matching, as the intervals are overlapping.

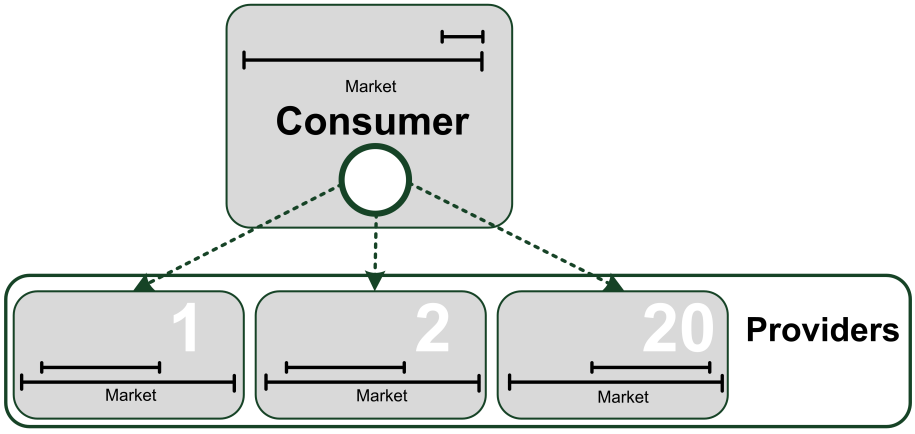


Figure 4.2: *Simulation flow*

A key issue within the simulation is the interval generation representing the provider range. The *consumer interval's* length and position are given. *Provider intervals* are generated in two steps.

1. Without loss of generality we make two constraints:

   Constraint 1: The whole market of each SLO parameter has a minimum of 0 and a maximum of 100.

   Constraint 2: The length of the *provider interval* is between 0 and 100.

   Each *provider* and *consumer interval* lies within this market range. A provider can offer a service parameter covering the whole market or just 1 unit of the market. The length of the *provider interval* is generated based on the uniform distribution considering the minimum and maximum of Constraint 2. For calculating the distribution of many small intervals and few long intervals, a further step is necessary.

2. After calculating the temporary uniform distributed length, the position of the intervals has to be generated. Simply randomly selecting a start point between 0 and 100−*provider interval length* would lead to a unsatisfactory result: A *consumer interval* placed in the near of the middle will have a higher probability of matching with a provider than those placed near of margins. The following two examples clarify this property.

- A *consumer interval*, placed **leftmost**, matches with *provider intervals* whose start points are within the *consumer interval*.
- A *consumer interval*, placed **at the middle**, matches with *provider intervals* whose start points are within the *consumer interval*. Further, a *provider interval* whose start point is smaller than the *consumer interval's* start point but whose end point is greater than the *consumer interval's* start point matches with the *consumer interval*.

Algorithm 2: Finding matching providers

```
Data: SLA from consumer
Result: Number of matching providers
for each experiment do
    for each provider on market do
        for each service parameter S do
            generate random interval representing provider range;
            if consumer and provider interval ¬overlapping then
                /*provider ¬matches*/
                break;
            if S is last parameter in SLA then
                /*provider matches*/
                start next experiment
```

However, as already mentioned, the position of the *consumer interval* should be negligible. To avoid this problem, we determine the position of the interval by defining flexible boundaries as shown by following equation.

$$\begin{aligned} min &= 0 - length/2 \quad max = 100 + length/2 \\ center_{\text{interval}} &= random(min, max) \end{aligned} \tag{4.1}$$

With this position generation the *consumer interval's* position is irrelevant. The so generated random intervals are *provider intervals*. However, this position calculation can lead to *provider intervals* adapting the market range as figure 4.3 shows.

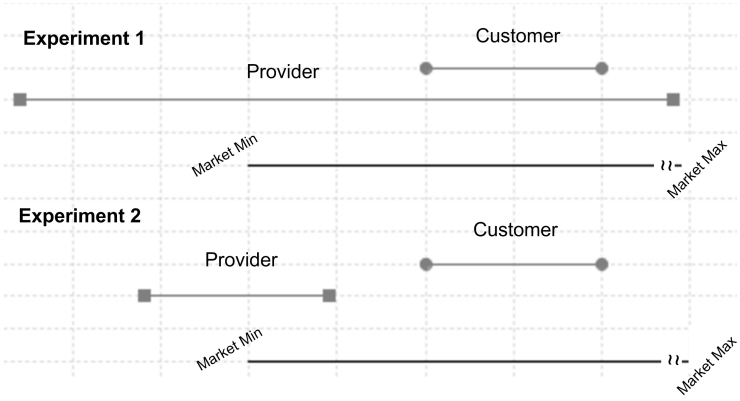


Figure 4.3: *Provider and consumer interval*

As the *consumer interval* is in the market range, we can simply cut the protruding interval part. This cut leads to few providers with a big range and a lot of providers with a small range. The distribution is exemplarily represented in the histogram in figure 4.4. For our forecast prediction model we assume that the service availability on the market follows a uniform distribution.

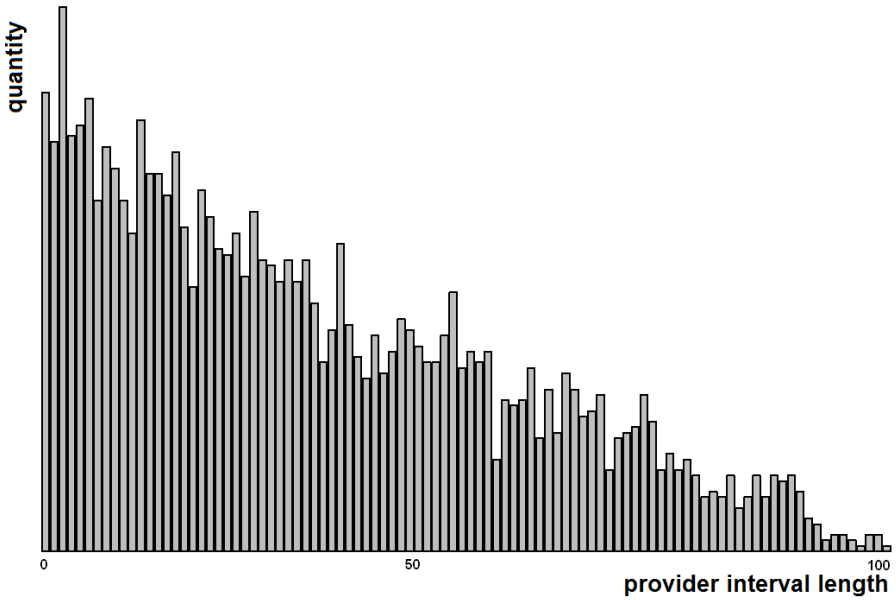


Figure 4.4: *Provider interval length simulation*

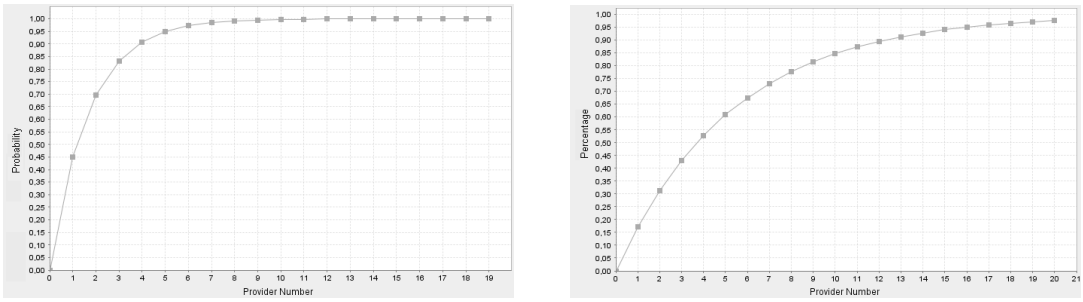
Table 4.3: *Simulation results for interval length 20*

provider	1	2	3	4	...
first matches	448844	247577	136070	75437	...

4.2.2 Simulation Result

The first simulation coped with consumer SLAs which contain one SLO parameter only. The goal was to count the number of providers needed to find one provider offering the requested SLO parameter. We executed the simulation with 1 million experiments for each *consumer interval* with a length of 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. Table 4.3 shows the results of the simulation for a *consumer interval* with a length of 20. In 448844 cases the first provider was the first matching provider, in 247577 cases the second provider was the first matching provider and so on. The total number of experiments was 1000000. Figure 4.5a visualizes the results of the simulation represented in table 4.3. The abscissa shows the providers' number, the ordinate shows the cumulative numbers of first matches, expressed as percentage. As the simulation shows, about 70% of first matches occur in the first two providers. The simulation reveals, that in 44.9% of all experiments, the *consumer interval* overlaps with the first randomly generated *provider interval*. Based on this simulation result, we conclude that the probability of overlapping of a *consumer interval* of length 20 and a random *provider interval* is about 44.9%.

The results shown in table 4.3 are valid for a *consumer interval* with a length of 20. Table 4.4 represents the numbers of first matches of the first provider for all the other *consumer interval* lengths expressed in percentages. Again, we conclude that these numbers represent the probabilities of overlapping with a random generated *provider interval*. With an increasing *consumer interval* length the probability, that the provider matches, increases. The number of experiments which match with the second, third,... provider are not relevant for our conclusions. In our second simulation we simulated the search for a provider offering a matching SLA containing more than one service parameter. If a SLA consists of n service parameters, a matching provider has to offer n service parameters respectively. Figure 4.6 shows an example of finding a SLA containing two service param-



(a) *Probability of finding matching providers for 1 interval of length* 20

(b) *Probability of finding matching providers for 2 intervals of length* 10 *and* 20

Figure 4.5: *Overlapping probability for arbitrary length*

Table 4.4: *Simulation results for all interval lengths*

interval length	10	20	30	40	50
first matches	38.1%	44.9%	51.7%	58.7%	65.6%
interval length	60	70	80	90	100
first matches	72.4%	79.3%	86.2%	93.2%	100%

eters A, B. The consumer's service parameter A has the length of 20 whereas parameter B has a length of 10. Provider 1 offers a matching service parameter A as well as a matching service parameter B. Consequently, provider 1 offers a matching SLA. Provider 2 offers a matching service parameter A but service parameter B is not matching. So provider 2 does not offer a matching SLA. The result of the simulation with these two services

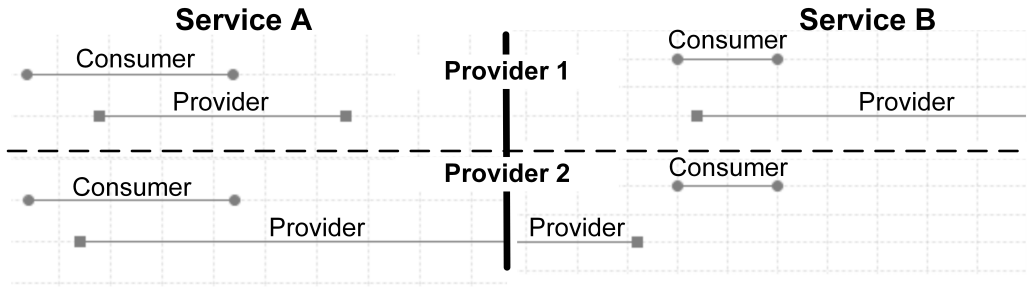


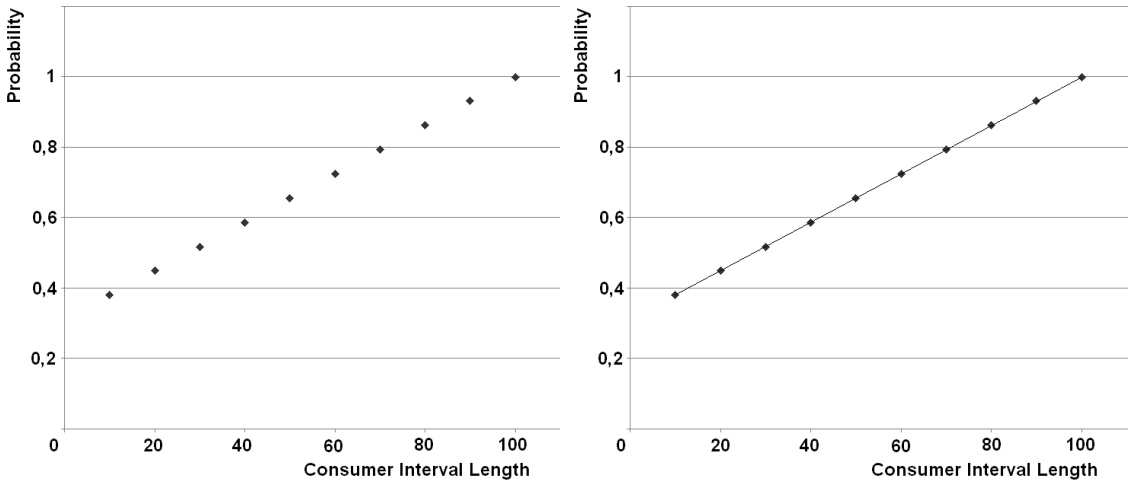
Figure 4.6: *Intervals of 2 providers and 2 services*

A and B is visualized in figure 4.5b. Analogously, to figure 4.5a this diagram represents the cumulative number of first matches expressed as percentages. The abscissa shows the providers' number whereas the ordinate represents the cumulative number of first matches expressed as percentage. The diagram shows, that about 17% of the first matches occur in the first provider. Finding a provider offering a matching SLA with more than one service is more difficult than finding a provider with one matching service parameter.

4.3 Analytical Prediction Model

Based on the simulation results we set up an analytical model forecasting the probability of finding matching SLAs given (i) the number of *consumer intervals* representing the service parameters (ii) and the length of these *consumer intervals*. First, we examine the simulation results shown in table 4.4. These numbers represent the overlapping probability of *consumer interval* and *provider interval*. For the forecast calculation, we need a formula returning the probability of overlaps for a *consumer interval* of any length. The results

of table 4.4 are visualized in figure 4.7b. The abscissa represents the *consumer interval* length. The ordinate represents the probability that the *consumer interval* overlaps with a randomly generated *provider interval*. This representation reveals a linear correlation between the *consumer interval* length and the probability that the *consumer interval* overlaps the *provider interval*. To prove the interval-length/probability correlation we use linear regression analysis.



(a) *Scatter plot - one parameter matching* **(b)** *Trend line - one parameter matching*

Figure 4.7: *Overlapping probability for arbitrary length*

4.3.1 Linear Regression Analysis

The simulation returned results for intervals with lengths 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. In order to get probabilities for all interval lengths we follow a regression trend line approach. Precondition for using regression analysis is the classification of the used variables into dependent and independent variables [73]. The independent variable is x, the interval length, and the dependent variable is y, the matching probability. The depend values are represented as the *first matches* lines in table 4.4. Interval length are listed in the *interval length* lines in table 4.4.

First, the average values of the two variables are calculated.

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = 55 \quad \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} y_i = 0.6901 = 69.01\% \tag{4.2}$$

The slope of the regression trend line is calculated by

$$s = \frac{\sum_{i=1}^{n}((x_i - \bar{X})(y_i - \bar{Y}))}{\sum_{i=1}^{n}(x_i - \bar{X})^2} = 0.00688667 \approx 0.0069 \tag{4.3}$$

The slope is about 0.69. The general linear equation is given by

$$y - \bar{Y} = s \cdot (x - \bar{X}) \tag{4.4}$$

We already calculated the variables $\bar{Y}$, $\bar{X}$ and s. Transforming the equation gives the linear expression shown in (4.6). This trend line (see figure 4.7b) describes the predicted

probability of overlapping intervals depending on the length of the *consumer interval* x.

$$y - 0.6901 = 0.00688667 \cdot (x - 55) \tag{4.5}$$

$$y = 0.00688667 \cdot x + 0.31133315 \tag{4.6}$$

After calculating the trend line the quality of approximation has to be checked. The vertical difference between the observed value and the predicted value, based on the trend line, is the residual R. The sum of all positive and negative residuals has to be 0 [73]. The sum of all squared differences "sum of square error (SSE)" measures the quality of the trend line. Generally, the smaller it is, the better is the approximation by the trend line.

$$SSE = \sum_{i=1}^{n} (y_{i_{observed}} - y_{i_{predicted}})^2 = 2.9333 \cdot 10^{-6} \tag{4.7}$$

The coefficient of determination R^2 represents the relation of SSE to SS_t. SS_t represents the total sum of the squared differences from the observed values to the average. By dividing SSE by SS_t, the SSE is standardized.

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SS_t} = 1 - \frac{\sum_{i=1}^{n} (y_{i_{observed}} - y_{i_{predicted}})^2}{\sum_{i=1}^{n} (y_{i_{observed}} - \bar{Y}_i)^2} \\ &= \frac{\sigma^2_{y_{predicted}}}{\sigma^2_{y_{observed}}} = 0.99999347 \end{aligned} \tag{4.8}$$

The R^2 is a standardized ratio, measuring the goodness of fit with a maximum of 1 and a minimum of 0. A R^2 of 1 means 100% of the variance is shared between the two variables.R^2 represents the ratio of the variances which in our calculation below is nearly 1. This means that the the trend line calculated by equation (4.6) is a very good approximation, as it shows a linear relationship between the two variables.

4.3.2 Probability Calculation

Equation (4.6) allows to calculate the probability of finding a provider, offering a single matching service parameter depending on the *consumer interval's* length. We call this probability "single probability". In order to get the probability of finding a whole SLA containing more than one SLO, all its single probabilities P_i have to be combined as shown in equation (4.9). It is assumed that the single probabilities are independent of each other.

$$P_{total} = \prod_{i=1}^{n} P_i = P_1 \cdot P_2 \ldots P_n \tag{4.9}$$

To clarify our model we introduce an example.

4.3.2.1 Example 1

Given is a consumer requested SLA consisting of three service parameters A, B and C. Service $A's$ length is 20, service $B's$ length is 30 and service $C's$ length is 10. First, we can calculate the probability of finding a matching provider for each service (equation (4.6))

by equation (4.10).

$$\begin{aligned} P(\text{Serv.A}) &= 0.0068886 \cdot 20 + 0.311333 = 44.91\% \\ P(\text{Serv.B}) &= 0.0068886 \cdot 30 + 0.311333 = 51.80\% \\ P(\text{Serv.C}) &= 0.0068886 \cdot 10 + 0.311333 = 38.02\% \end{aligned} \tag{4.10}$$

The probability of finding a provider, offering a matching SLA, can be calculated by multiplying the single probabilities:

$$\begin{aligned} &P(SLA) = P(\text{Serv.A}) \cdot P(\text{Serv.B}) \cdot P(\text{Serv.C}) = \\ &44.91\% \cdot 51.80\% \cdot 38.02\% = 8.84\% \end{aligned} \tag{4.11}$$

Thus, the probability of finding a matching SLA, consisting of the three service parameters A, B, and C, is 8.84%. However, it is more important for the consumer to know the probability of finding at least one matching provider by a given number of providers than to know the probability of finding a provider with a matching SLA. In our case, the number of experiments represents the number of providers which are contacted. This factor is limited by the number of providers on market. The number of successes is the number of matching providers. The probability of success for one experiment depends on the consumer's interval length (equation (4.9)). The binomial distribution expects that the experiments are independent of each other. This means that finding a matching provider does not influence the result of the other experiments. The general form for calculating the binomial distribution is given in equation (4.12), where n represents the number of experiments, k the number of successes, and $P(k|p,n)$ represents the probability that exactly k successes occur within n experiments. p is the probability that one event succeeds.

$$P(k|p,n) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{4.12}$$

However, we do not need the probability that exactly k successes occur by executing n experiments as shown in equation (4.12). As already mentioned, we need the probability that at least one success, which represents the finding of one matching provider, occurs. We need to use the inverse probability. Thus, we subtract the probability of *no event succeeds* from 1 in order to get the probability of *at least one event succeeds*:

$$P(\text{at least one success}) = 1 - P(\text{no success}) \tag{4.13}$$

4.3.2.2 Example 2

We extend our example from section 4.3.2.1. The probability of finding a matching SLA is 8.84%. The consumer is interested in the probability of finding at least one matching provider. The market, which is consulted by the consumer, has 20 providers. First, we calculate the probability of *no success* using equation (4.12).

$$P(k|p,n) = 1 \cdot 0.0884^0 \cdot (1-0.0884)^{20-0} = 0.1571 \tag{4.14}$$

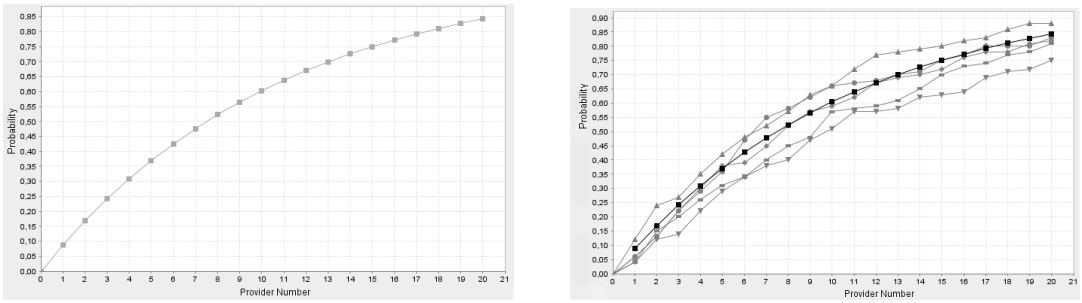
The probability of finding exactly zero matching providers, by contacting 20 providers, is 15.71%. Now we have to apply the inverse probability in order to get the probability of finding at least one matching provider.

$$1 - P(\text{no success}) = 1 - 0.1571 = 0.8429 \tag{4.15}$$

The probability of finding at least one provider consulting 20 providers is about 84.29%. We validate our calculation by comparing the results to the simulation results. Therefore we execute the simulation with a SLA containing the same services as described in the example presented in section 4.3.2.1. Figure 4.8a depicts the simulation result. The

probability of finding at least one provider offering a matching SLA is about 84% by consulting 20 providers. So the simulation delivers approximately the same result as the calculation. This fact is summarized in figure 4.8b which illustrates 5 simulations (grey lines), whereby each simulation was executed with 100 experiments. The dark line is the calculated forecast. The calculated line is approximately the average of the simulated lines. Summing up, the following list recaps the steps for calculating the probability to find at least one matching provider:

- Calculate the probability of finding a provider for each single service parameter of a SLA using the trend line equation.
- Combine the single probabilities by multiplication resulting the probability of finding a provider offering a matching SLA.
- Finally, use the binomial distribution in order to calculate the probability of finding at least 1 matching provider.



(a) *SLA simulation for 20 providers* **(b)** *SLA simulation and forecast calculation*

Figure 4.8: *Overlapping probability for arbitrary length*

4.4 Optimization Algorithm

Consumers may face the typical situation that the probability returned by the probability forecast is too low. In such cases, the consumer may want to know how to set the intervals in order to make sure to find at least one matching provider for a specific SLA. Therefore we developed an optimization algorithm within our simulation. Within this thesis we assume that a probability of greater than 99% means that a match is guaranteed. For this case we use the term *sure*. Precondition for the optimization is to assign priorities to the service parameter of the requested SLA. A high priority means that the service parameter is very important for the consumer and that the interval should not be adapted in order to find at least one matching provider. The optimizer starts extending the service parameter interval with the lowest priority. If this extension is not enough to reach a probability of 99%, the next service parameter with the second lowest parameter is extended. This algorithm is depicted in algorithm 3.

In this section we further extend the example from section 4.3.2.2. The probability of finding at least one matching service was about 84% and consequently lower than the limit of 99%. The priorities of the SLA are shown in table 4.5. Algorithm 3 calculates the results shown in the rightmost column of table 4.5. Service C, the service with the lowest priority, was extended to a length of 84, the other services remained stable. With these service parameter lengths, the consumer can be sure to find at least one provider offering a matching SLA by consulting 20 providers.

Algorithm 3: Interval optimization

```
Data: Requested SLA service parameters prioritized
Result: Adapted service parameter intervals
sort list of services by priority;
set iterator to service with lowest priority;
while true do
    calculate probability of finding at least one provider;
    if probability larger than 99% then
        break;
    if length==100 then
        next service;
    else
        service length++;
```

Table 4.5: *Service priorities*

Service	Priority	Length
Service A	2	20
Service B	1 (highest)	30
Service C	3 (lowest)	84

4.5 Negotiation Range Analysis

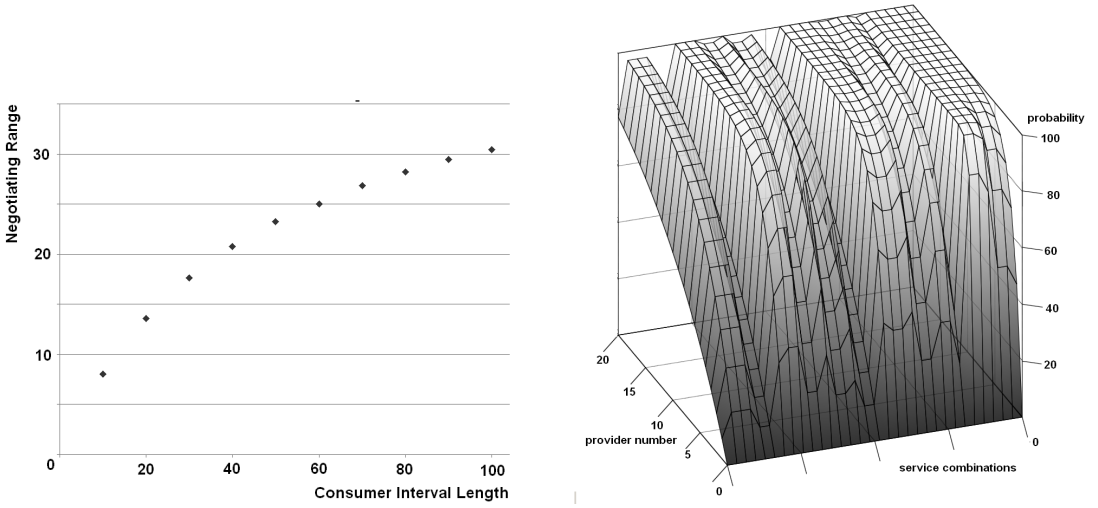
Until now we only checked, if the *provider intervals* overlap with the *consumer intervals*. We neglected the depth of overlapping. A deep intersection leads to a big negotiating range, whereas a small intersection leads to a small negotiating range. To the best of the authors' knowledge no mathematical analysis of the negotiating depth is available. Therefore we executed a simulation in order to identify a distribution. Again, the *provider interval's* length as well as position, are generated randomly. We executed the simulation with a *consumer interval* length of 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. For each consumer length, we executed the simulation 1000000 times and calculated the average negotiating range. The results of our simulation are shown in table 4.6.

Table 4.6: *Overlapping distance*

interval length	10	20	30	40	50
negotiation range	8.01	13.56	17.63	20.76	23.22
interval length	60	70	80	90	100
negotiation range	25	26.86	28.23	29.44	30.43

In order to forecast the negotiation range of a SLA, containing any number of services with any length, we have to approximate the simulation results. The results of table 4.6 are visualized in figure 4.9a. It can be seen that there is not a linear correlation between the negotiating range and the *consumer interval* length. In this case, a linear regression trend line would lead to inappropriate forecasts. By a logarithmic transformation $x_{new} = ln(x)$ of the data we can apply the linear regression trend line as approximation. The simulation after transformation is given in table 4.7 and visualized in figure 4.10a. Concluding from the scatter plot we apply linear regression to approximate the data analogously to section 4.3.1.

First of all, the averages of the two variables are calculated. Instead of the x values we use the ln-transformed x values. $\bar{X}$ and $\bar{Y}$ are calculated with the formulas in equation (4.2). Therefore $\bar{X} = 3.813$ and $\bar{Y} = 22.314$. The slope s is calculated with the formula



(a) *Negotiating range* **(b)** *Probability for at least 1 matching provider*

Figure 4.9: *Negotiation range*

Table 4.7: *Overlapping distance transformed*

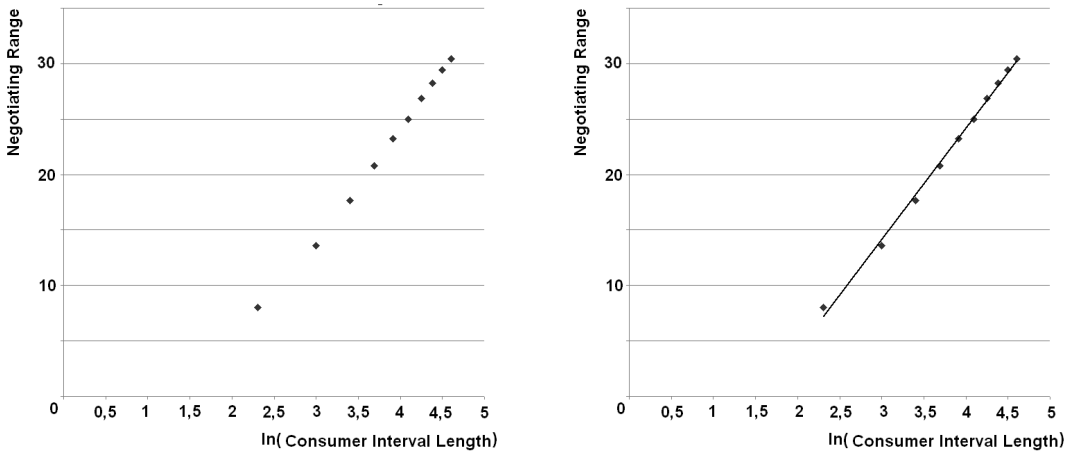
ln(interval length)	2.303	2.996	3.401	3.689	3.912
negotiation range	8.01	13.56	17.63	20.76	23.22
ln(interval length)	4.094	4.248	4.382	4.5	4.605
negotiation range	25	26.86	28.23	29.44	30.43

in equation (4.3) and consequently $s = 10.01$.

The equation below shows the transformed general linear equation which is illustrated in equation (4.4).

$$y = 10.01 \cdot x - 15.85413 \tag{4.16}$$

After calculation of the trend line, which is visualized in figure 4.10a, we have to check its quality by calculating its coefficient of determination. For the SEE and consequently the R^2 we used the formulas of the equations (4.7) and (4.8). So the $SSE = 1.607$ and the $R^2 = 0.9967$



(a) *Scatter Plot* **(b)** *Trend Line*

Figure 4.10: *Negotiation range transformed*

By the coefficient of determination near of 1, the approximation can be considered as pretty good. Due to the fact that we used a logarithmic transformation, we have to be careful to make a prediction by the trend line, e.g. if a consumer wants to predict its negotiating range for one service of length 60, a logarithmic transformation has to be applied by $x_{new} = ln(60) = 4.094$. The new x value can be used in the trend line equation.

$$y_{predicted} = 10.01 \cdot 4.094 - 15.85413 = 25.127 \tag{4.17}$$

So the predicted negotiating range of about 25.1 is in line to the simulated value 25, as shown in table 4.7.

4.6 Model Use Case

In this section we want to provide a comprehensive example using all the approaches we developed before. We assume that a consumer requests a SLA containing five services as shown in table 4.8 on the left side. By this example we will answer the following questions:

Table 4.8: *SLA services and negotiation range*

Service Name	Range	Priority	Neg. Ranges	Int. Opt.
Service A	20	1 (highest)	14.14	20
Service B	30	2	18.19	30
Service C	20	3	14.14	84
Service D	70	4	26.67	100
Service E	80	5 (lowest)	28.01	100

- What is the probability of finding at least one matching provider?
- What is the expected negotiating range?
- Which intervals should be adapted to be virtually sure to find at least one matching provider?
- How many providers have to be consulted in order to be virtually sure to find at least one matching provider?

First of all, we start calculating the single probability of finding a matching provider for each service parameter of the SLA by using the trend line equation (4.6):

$$\begin{aligned} &P(\text{Serv.A}) = 44.9\% \quad P(\text{Serv.B}) = 51.79\% \\ &P(\text{Serv.C}) = 44.9\% \quad P(\text{Serv.D}) = 79.34\% \\ &P(\text{Serv.E}) = 86.22\% \end{aligned} \tag{4.18}$$

Now we combine the single probabilities by multiplying them as shown in equation (4.9).

$$\begin{aligned} &P_{\text{total propability}} = 0.449 \cdot 0.518 \cdot 0.449 \cdot 0.793 \cdot 0.8622 = \\ &0.071423 \end{aligned} \tag{4.19}$$

In order to get the probability of finding at least one matching provider consulting 20 providers, we have to use the binomial distribution as defined in equation (4.12).

$$P(k|p,n) = 1 \cdot 0.071423^0 \cdot (1 - 0.071423)^{20-0} = 0.227 \tag{4.20}$$

The probability of finding one matching provider is about 22.7%. We get the probability of finding at least one matching provider by the inverse probability equation (4.13).

$$1 - P(\text{no success}) = 1 - 0.227 = 0.773 \tag{4.21}$$

The probability of finding at least one matching provider is 77.3%. Figure 4.10b illustrates the result as 3D plot. The ordinate shows the probability of finding at least one matching provider. The service combination axis represents all possible service combinations. The left most service combination is the combination containing all five services. The probability of finding a matching provider for these five services is 77.3% consulting 20 providers. The right most service combinations are the single services A, B, C, D and E. The number of combinations are given on the "service combinations" axis in figure 4.10b.We have to sum up the number of combinations using combinations consisting of 1, 2, 3, 4 and 5 services. For $n = 5$ it results in 31 combinations.

$$\sum_{k=1}^{5} \frac{5!}{(5-k)! \cdot k!} = 31 \tag{4.22}$$

Calculating the negotiating range we do a logarithmical transformation of the *consumer intervals'* lengths (x values) in order to use equation (4.16). Equation (4.16) delivers the results shown in table 4.8 on the right side. The sum of all negotiating ranges is 101.15.

Finally, we adapt the service parameters in order to find a provider with a probability of at least $> 99\%$. The priorities of the services are shown in table 4.8. Executing the optimization algorithm 3 leads to the optimized interval lengths shown in the rightmost column of table 4.8. Three services have to be adapted in order to find a provider with a probability greater than 99% consulting 20 providers. If the consumer does not want to extend his service intervals, he has to increase the number of services consulted. Instead of consulting 20 providers, the consumer has to consult 63 providers in order to find at least one matching provider. Table 4.9 shows the probability of finding at least one matching provider depending on the providers consulted. This number can be calculated by an iterative approach. The number of providers is increased by one, until the probability of finding at least one provider exceeds 99%.

Table 4.9: *Probability depending on number of providers consulted*

Providers	Probability	Providers	Probability
20	77.3%	50	97.6%
25	84.4%	55	98.3%
30	89.2%	60	98.8%
35	92.5%	61	98.989%
40	94.9%	62	98.995%
45	96.5%	63	99.1%

4.7 Simulation Prototype

In this section, we present our prototypes and give an overview of their features.

We developed two prototypes in Java, the "Simulator" and the "Cockpit". The Simulator's user interface is depicted in figure 4.11. The Simulator reads the consumer's service parameter properties such as length and position from a XML file. These properties can be modified within the GUI. Before starting simulation, the GUI allows to set the random *provider interval* generation. The *provider intervals* can be generated as described. Another option would be to generate the provider's intervals with a random position and fixed lengths. The Simulator visualizes the simulation experiments by representing the consumer's and provider's ranges as intervals. Further the simulated probability distribution of finding at least one matching provider for a single service parameter as well as the probability distribution of finding at least one matching provider for a whole SLA is visu-

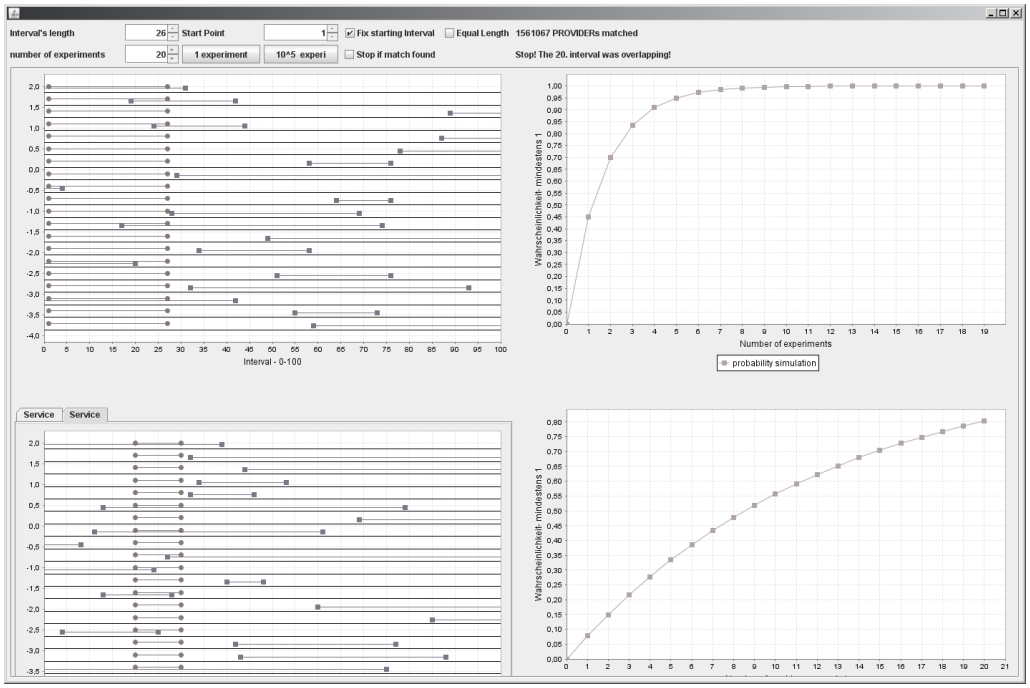


Figure 4.11: *Simulator user interface*

alized. We used these graphics in figure 4.5a and 4.5b. The goal of the second prototype Cockpit is to visualize the calculated forecasts. A screen shot of the Cockpit is given in figure 4.12. This prototype allows to set the number of services, the number of providers and to modify the *consumer interval* lengths as well as their priorities. The results are visualized as already shown in figure 4.8b and 4.10b.

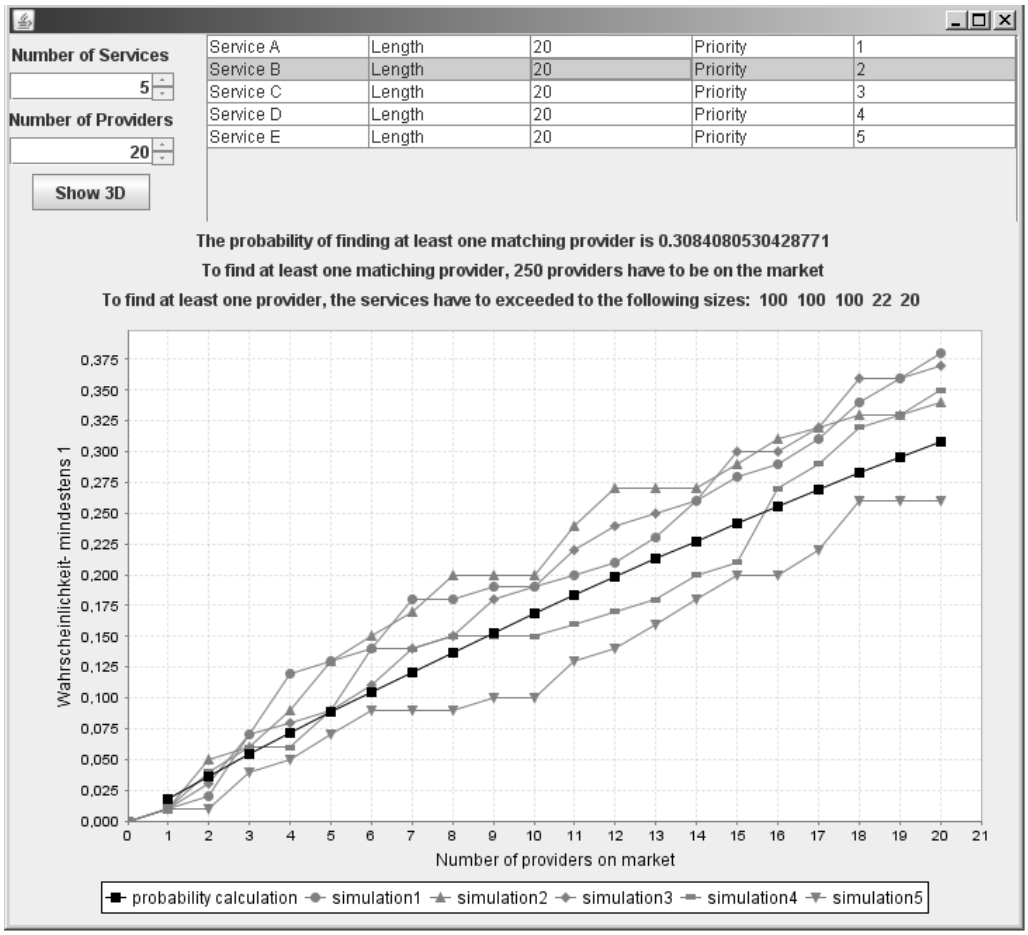


Figure 4.12: *Cockpit user interface*

5 Cloud Cost Model considering variable Cost

In this section we introduce the results of our research. We have invented an economic cloud cost model using traditional economic principles, fundamentals and methods. Our cloud cost model contains a mapping between traditional economic principles and the cloud environment. Current cloud cost models only consider fixed cost. We show that using variable cost leads to more precise decisions during the negotiating process.

5.1 Model Fundamentals

In the following we describe the way of how the fundamentals and methods of economy can be applied on an economic cloud cost model. We introduce the mappings onto the cloud environments for all fundamental economic methods.

5.1.1 Traditional Economics Mapping

As mentioned in section 2.5, distinguishing between variable cost as well as fixed cost enables application of traditional methods of economics to the cloud cost model.

Operating Production Factors.

In our model we consider a cloud environment as a traditional production company. This model has "production" factors defined similarly to a conventional production process.

Production factors used in our model are:

- storage devices
- servers
- network devices

Produced goods in our model are:

- storage capacity
- performance (processing power)
- network bandwidth

We assume to have limited substitution of production factors. That means e.g. for the production factor "servers" that we can add or remove only whole servers and not parts of them. In future versions of our model we want to consider full substitutability.

The storage capacity is measured in Terabyte. The performance or processing power is measured as the throughput in Server Side Java Operations (ssj_ops) defined in [74]. The network bandwidth is measured in Mbit/s.

The further discussion is based on data taken from a SPECpower_ssj2008 benchmark test [75] developed by SPEC (Standard Performance Evaluation Council) which focuses on performance and power consumption. Typically, the benchmark test consists of ten target levels. For each level the power consumption is related to the performance. The most important performance factors are the Java Virtual Machine (the transactions are

executed by a Java application (ssj = server side java)), multiple ssj instances, affinity to ssj instances and hardware and operating system settings. Power factors are the operating system power management, power supplies, BIOS fan speed control and storage configuration. ssj operations per Watt is the most important indicator of this benchmark and represents the energy efficiency. It is calculated as the ratio of the sum of all ssj operations scores for all target loads and the sum of all power consumption averages in Watts for all target loads.

We carefully chose this benchmark as the energy efficiency in this benchmark is also measured during these steps. The benchmark lists many servers from different vendors varying from high efficient to low efficient performance per Watt in terms of energy consumption. Thus, this benchmark enables us to apply variable cost in our model. A general overview of performance evaluation methods and metrics can be found in [75]. For the energy consumption/target load relation we use the regression trend line as approximation describing statistically the relationship between variables e.g. energy consumption and target load. This method allows to find the trend line with the least total distance to all observed values [76].

In our example the dependent variable y represents energy consumption, the independent variable x represents the target load. First of all we start calculating the barycenter of all values by calculating the average x and y value. The data for the regression trend analysis are taken from the SPEC benchmark. The values of the variable x are depicted in figure 5.1 on the x-axis and represent the percentage of the load from active-idle to full target load. The range of the x values is from 0% to 100%. The y axis shows the values for the power consumption as percentage of the power consumption at full load relative to the active idle power consumption. The data taken from the SPEC benchmark are listed in table 5.1 to calculate the y values. These data are only an excerpt of the entire benchmark.

Table 5.1: *Server power consumption in % of target load [watt]*

Server	@100%	@90%	@80%	@70%	@60%	@50%	@40%	@30%	@20%	@10%	@idle
1	475	430	406	358	323	289	260	234	211	185	101
2	232	210	190	173	158	148	137	126	116	104	64
3	883	822	756	684	613	555	504	460	419	369	216
4	518	488	457	430	400	361	324	290	262	234	142
5	259	242	226	207	190	175	162	150	139	126	85
6	125	117	106	95	83	72	65	58	51	46	35
7	652	616	576	538	500	465	434	404	375	337	267
8	172	162	151	142	131	121	112	104	96	83	56
9	244	228	212	196	180	166	151	137	122	106	74
10	218	206	193	175	160	148	134	122	114	100	67
11	220	204	186	170	153	141	131	121	111	97	65
12	902	814	733	691	650	608	568	533	499	458	360
13	218	206	193	181	165	151	138	124	111	95	80
14	60,9	57	54	51	48	44	41	38	34	30	25
...	...	...	...	...	...	...	...	...	...	...	...

For $x = 10\%$ of target load we calculate the y value:

$$y = \frac{Average\_watts\_@100\%\_of\_target\_load - Average\_watts\_@10\%\_of\_target\_load}{Average\_watts\_@10\%\_of\_target\_load - Average\_watts\_@active\_idle} \quad (5.1)$$

For $x = 20\%$ of target load we calculate the y value:

$$y = \frac{Average\_watts\_@100\%\_of\_target\_load - Average\_watts\_@20\%\_of\_target\_load}{Average\_watts\_@20\%\_of\_target\_load - Average\_watts\_@active\_idle} \tag{5.2}$$

and so on.

We calculate the average values of $\bar{X}$:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = 50.0 \tag{5.3}$$

For the data taken from the SPEC benchmark of each server we use all percentages (10% to 100%) of the target load to calculate the average value $\bar{Y}$:

$$\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} y_i = 53.013 \tag{5.4}$$

Now we calculate the slope s of the trend line by

$$s = \frac{\sum_{i=1}^{n}((x_i - \bar{X})(y_i - \bar{Y}))}{\sum_{i=1}^{n}(x_i - \bar{X})^2} = 0.9631 \tag{5.5}$$

The generalized linear equation is

$$y - \bar{Y} = s \cdot (x - \bar{X}) \tag{5.6}$$

Due to the fact that we know every variable without x and y we can calculate the linear equation of the trend line. With this function we get expected energy consumption by a given target load.

$$y - 53.013 = 0.9631 \cdot (x - 50.017) \tag{5.7}$$

$$y = 0.9631 \cdot x + 4.8583 \tag{5.8}$$

Residuals R are the vertical between the predicted value, based on the trend line, and the observed value. The sum of positive differences is equal to the sum of negative differences which leads to a total sum of 0.

$$R = \sum_{i=1}^{n} y_{i_{observed}} - y_{i_{predicted}} = 0 \tag{5.9}$$

The sum of square error SSE uses squared values representing the sum of squared residuals. SSE describes how well the line fits; the smaller SSE the better is the approximation. SS_t represents the total sum of squares (proportional to the sample variance).

$$SSE = \sum_{i=1}^{n}(y_{i_{observed}} - y_{i_{predicted}})^2 = 89747.024 \tag{5.10}$$

The coefficient of determination R^2 measures the relation of SSE to SS_t.

$$R^2 = 1 - \frac{SSE}{SS_t} = 1 - \frac{\sum_{i=1}^{n}(y_{i_{observed}} - y_{i_{predicted}})^2}{\sum_{i=1}^{n}(y_{i_{observed}} - \bar{Y_i})^2} = \frac{\sigma^2_{y_{predicted}}}{\sigma^2_{y_{observed}}} \tag{5.11}$$

The R^2 ratio measures the goodness of fit [77]. R^2 takes on values between 0 and 1, whereas 1 means that 100% of the variance are shared between the two variables [78]. The variance is a measure of dispersion. [79]

To get the coefficient of determination we must calculate the variance of the observed values as well as of the predicted values. The variances of the observed and predicted values is calculated using the following formula.

$$\sigma^2_{y_{observed}} = \frac{\sum_{i=1}^{n}(y_{i_{observed}} - \bar{Y})^2}{numberofvalues} = 959.668 \tag{5.12}$$

$$\sigma^2_{y_{predicted}} = \frac{\sum_{i=1}^{n}(y_{i_{predicted}} - \bar{Y})^2}{numberofvalues} = 927.562 \tag{5.13}$$

The coefficient of determination is the ratio of the variances.

$$R^2 = \frac{\sigma^2_{y_{predicted}}}{\sigma^2_{y_{observed}}} = 0.9665 \tag{5.14}$$

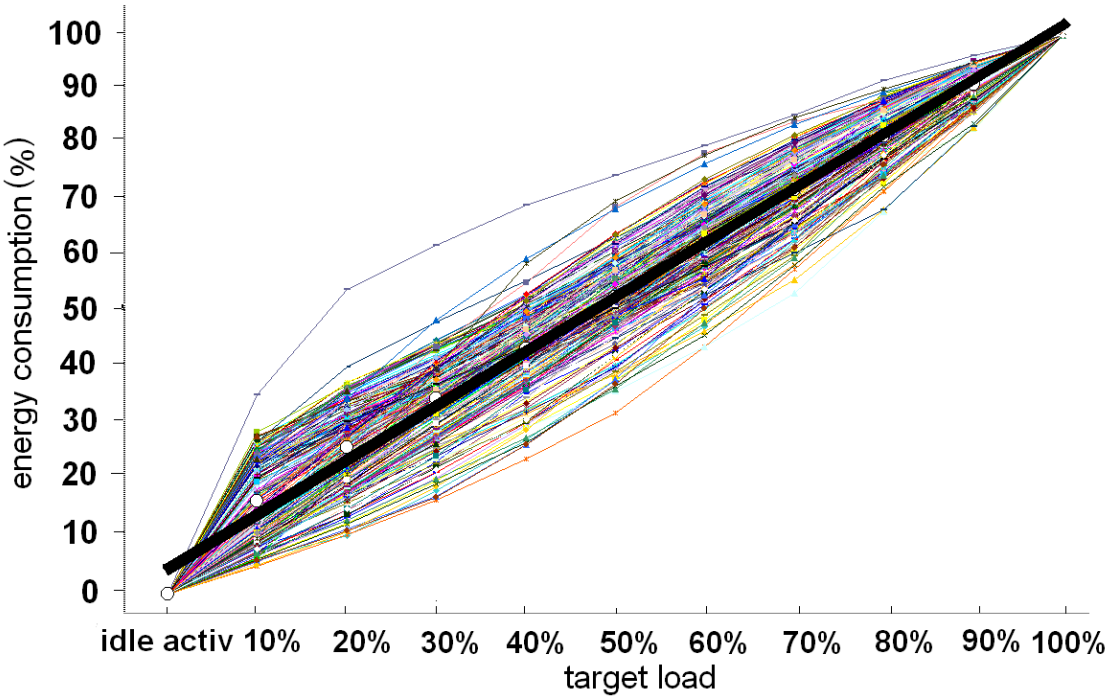


Figure 5.1: *Linear power consumption*

Based on this apparatus we analyze the benchmark results to verify the linearity of the power consumption from active idle, to full load. The results are depicted in figure 5.1.

The server power consumption during processing can be approximated by linear regression. This is true for all servers. In figure 5.1 all servers with their power consumption/processing power functions are shown.

In our cloud cost model we distinguish between the following fixed production cost:

- depreciation
- occupancy

- administration
- power consumption during idle time
- network infrastructure

Moreover we distinguish between the following variable cost:

- power consumption
- network bandwidth

The model is based on variable cost as well as fixed cost. Our fixed cost comprise typical fixed cost as stepped cost such as the administration cost and the broadband cost. These cost are not increased by each additional unit produced, they are increased by a significant output change, e.g. the administration cost increase with every hundred servers. We aim for keeping the model concise, therefore we decided to neglect stepped cost in the whole model. In further extensions of this model these stepped cost will be added, too.

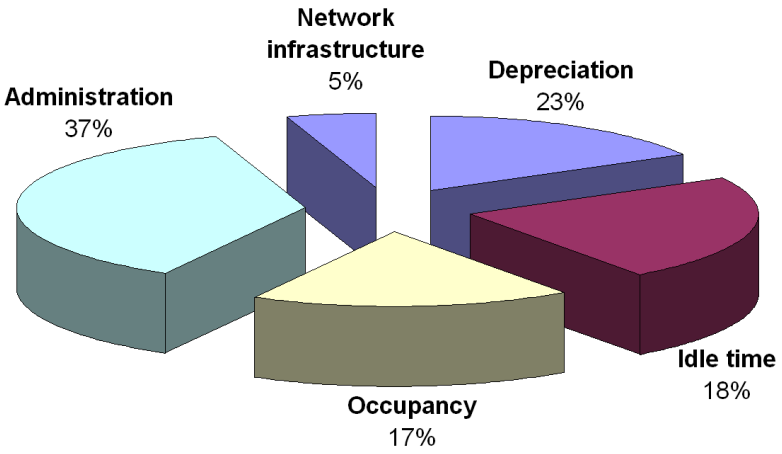


Figure 5.2: *Cost Overview*

By selecting the factors of production, the profit can be calculated. Due to the fact that each additional disc space, processing power and bandwidth increases the output directly, these functions are linear. Such a linear relationship is depicted in figure 5.3.

5.1.2 Architecture

Our cost model is based on three data stores, one for the servers, one for the hard discs and one for network bandwidth. The server database stores all the server-relevant information such as *ssj* operations per Watt, memory, price and CPU. The architecture of our model is depicted in figure 5.4. The input and output parameters of the model are listed in table 5.2 and 5.3.

5.1.3 Performance per Energy

The Standard Performance Evaluation Corporation is currently working on a new project called Server Efficiency Rating Tool (SERT) [80]. SERT's aim is to provide a first order of approximation of energy efficiency across broad range for application environments and to create a rating tool for overall energy efficiency and a measuring tool for power, performance and inlet-temperature. SERT consists of a test harness, the director, the workload, the SPEC PTDaemon and the reporter [74].

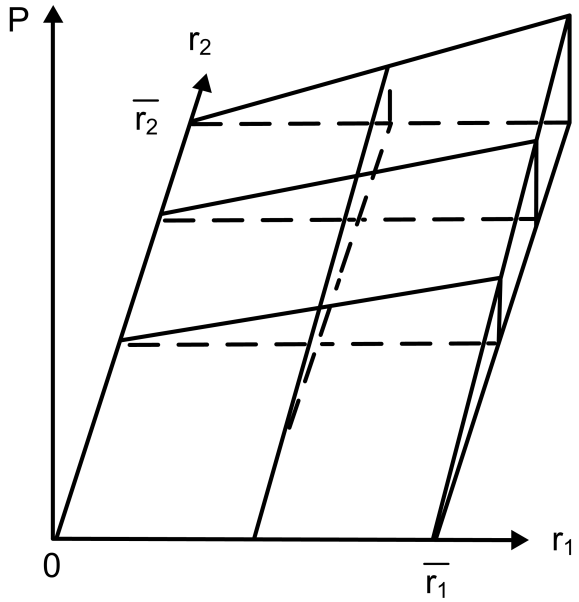


Figure 5.3: *Production Factor Combination*

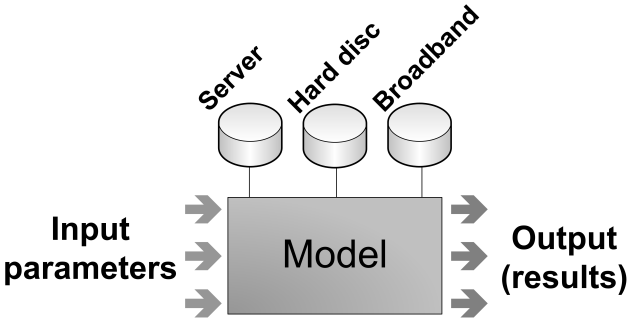


Figure 5.4: *Model Architecture*

Due to the fact that the price difference between servers with similar efficiency is negligible, we decided to subdivide servers into efficiency groups. The hard disk database stores the capacity, price and the required power in Watt.

5.2 A Variable Cost Based Cloud Cost Model

We divide our model into two sections: one for *servers* and one for *hard discs*. Each server and hard disc stored in the database can be selected. Based on the units and the server ssj operations per Watt the model calculates the required energy demand in kilowatt-hour (kWh) (for both idle time and busy time) and the total ssj operations per Watt. The fixed cost include power consumption during idle time; power consumption during busy time is part of the variable cost. Furthermore the number of racks and the layout efficiency can be defined to get the footprint required for the cloud computing environment. Based on the footprint and the cost per square feet ($sqft$) the total housing cost are calculated.

The server depreciation cost are based on the total acquisition cost and the economic life. The administration cost are comprised of system monitoring, project management, engineering, installation and maintenance cost [81].

Power consumption cost during idle time, administration cost, occupancy cost and de-

preciation cost are fixed cost. Power consumption cost during busy time are variable cost.

The cost of networking equipment for cloud computing environments is primarily caused by switches, routers, and load balancers [59].

Before calculating transaction volume and revenue, the maximum price (the price where the demand is zero) and the maximum quantity (the quantity which is sold if price is zero) of the linear price consumption curve have to be identified.

Now we describe the input and output parameters of our model. As mentioned in section 2.5 the total cost are comprised of fixed cost and variable cost.

$$C_{total} = C_{fix} + C_{var} \tag{5.15}$$

The abbreviations and their respective descriptions of our model parameters are listed in table 5.2 and table 5.3.

Model Parameters Overview

Table 5.2: *Model input parameters*

acronym	description
$area_{HD}$	Required area for hard disc in sqft
$area_S$	Required area for server in sqft
C_{ops}	Occupancy cost per sqft
C_{ppk}	Power cost per KWh
EL	Economic life
k_{HD}	gradient of the price consumption curve storage
k_S	gradient of the price consumption curve computation power
q_{HD}	Storage quantity
q_S	Computation quantity
$ppssj$	Price per ssj_ops.
ssj_ops	ssj operations
TB	Terabytes

5.2.1 Fixed Cost

The depreciation cost of servers and hard disks, occupancy cost, administration cost, power consumption cost during idle time and the cost of network gears like routers and switches are fixed cost:

$$C_{fix} = C_{depr} + C_{occ} + C_{admin} + C_{IT_S} + C_{IT_{HD}} + C_{net} \tag{5.16}$$

The economic life and the server and hard disc acquisition cost determine the monthly depreciation cost:

$$C_{depr} = \frac{(C_{acqu_S}) + (C_{acqu_{HD}})}{12 \cdot EL} \tag{5.17}$$

The occupancy cost consist of the footprint needed for the server racks and the area needed for the hard disc racks:

$$C_{occ} = area_S \cdot C_{ops} + area_{HD} \cdot C_{ops} \tag{5.18}$$

Table 5.3: *Model Output*

acronym	description
C_{acqu}	Total acquisition cost
$C_{acqu_{HD}}$	Hard disc acquisition cost
C_{acqu_S}	Server acquisition cost
C_{admin}	Administration/maintenance cost
$C_{admin_{HD}}$	Administration/maintenance cost for hard discs
C_{admin_S}	Administration/maintenance cost for servers
C_{broad}	Broadband cost
C_{depr}	Depreciation cost
C_{fix}	Fixed cost
C_{IT_S}	Idle time server cost
$C_{IT_{HD}}$	Idle time hard disc cost
C_{net}	Network infrastructure cost
C_{occ}	Total occupancy cost
C_{total}	Total cost
C_{var}	Variable cost
$C_{var_{HD}}$	Variable cost of hard disc
C_{var_S}	Variable cost of server
r	revenue
pc	profit contribution
p_{HD}	Price for storage per unit
p_S	Price for computation power per unit
$power_{IT_{HD}}$	Power consumption hard disc idle time
$power_{IT_S}$	Power consumption server idle time
$power_{HD}$	Power consumption of hard disc
$power_S$	Power consumption of server
tv	transaction volume
T_{ssj_ops}	Total ssj_ops
T_{TB}	Total tera byte

The total administration cost are comprised of the hard disc administration cost and the server administration cost:

$$C_{admin} = C_{admin_{HD}} + C_{admin_S} \tag{5.19}$$

The server power cost consumption during idle time multiplied by the price per KWh defines server power cost:

$$C_{IT_S} = power_{IT_S} \cdot C_{ppk} \tag{5.20}$$

The hard disk power cost consumption during idle time multiplied by the price per KWh defines hard disc power cost:

$$C_{IT_{HD}} = power_{IT_{HD}} \cdot C_{ppk} \tag{5.21}$$

5.2.2 Variable Cost

Server variable cost, hard disc variable cost and the broadcast cost comprise the variable cost of our model:

$$C_{var} = C_{var_S} + C_{var_{HD}} + C_{broad} \tag{5.22}$$

The contribution of each variable cost part to the equation above is defined respectively:

$$C_{var_S} = \frac{(power_S - power_{IT_S}) \cdot C_{ppk}}{T_{ssj\_ops}} \cdot ssj\_ops \tag{5.23}$$

$$C_{var_{HD}} = \frac{(power_{HD} - power_{IT_{HD}}) \cdot C_{ppk}}{T_{TB}} \cdot TB \tag{5.24}$$

5.2.3 Price-Consumption Curve

The price in a linear price-consumption curve with negative slope is calculated from the quantity multiplied by the gradient of the price-consumption curve:

$$k_S = \frac{\Delta p}{\Delta q} \tag{5.25}$$

and

$$k_{HD} = \frac{\Delta p}{\Delta q} \tag{5.26}$$

$$p_S = q_S \cdot k_S \tag{5.27}$$

$$p_{HD} = q_{HD} \cdot k_{HD} \tag{5.28}$$

5.2.4 Transaction Volume

The transaction volume comprises the price per server and hard disk multiplied by the sold ssj_ops and Terabyte:

$$tv = q_S \cdot p_S + q_{HD} \cdot p_{HD} \tag{5.29}$$

5.2.5 Revenue

Finally the revenue is comprised of the total transaction volume minus total cost:

$$r = tv - C_{total} \tag{5.30}$$

5.3 Application of the Cloud Cost Model

In this section we describe the way of how to use our variable cost based cloud cost model and present its applications on practical examples.

5.3.1 Linear Optimization Problem

Problem statement: A cloud provider has 30000$ and is going to expand its capacity. However the provider does not know which server is the most profitable one.

Based on our model, we can provide a solution for this optimization problem. In the following example we present a simple linear optimizing solution without considering the effect of compound interest.

The cloud provider can choose between two servers. Server 1 is more efficient but the acquisition cost are higher than for server 2 (see table 5.4).

Table 5.4: *Servers*

Server	Acquisition cost	ssj_ops	Watt	profit contribution
Server 1	7612$	2843	181 W	72,70$
Server 2	3000$	1909	385 W	30,49$

We assume that the price for one ssj_ops is 0,03$. The profit contribution is comprised of the transaction volume minus the variable cost.

$$tv_{Server1,2} = ssj\_ops_{Server1,2} \cdot ppssj \tag{5.31}$$

The following calculation shows that the profit contribution of server 1 is higher than the profit contribution of server 2.

$$tv_{Server1} = 2843 ssj\_ops \cdot 0,03\$ = 85,29 \tag{5.32}$$

$$tv_{Server2} = 1909 ssj\_ops \cdot 0,03\$ = 57,27 \tag{5.33}$$

The variable cost are the power consumption cost during busy time (energy consumption during idle time are part of the fixed cost).

$$C_{var_{Server1,2}} = KWh_{Server1,2} \cdot C_{ppk} \tag{5.34}$$

The following calculation shows that the variable cost (power consumption cost) of server 2 are higher than server 1.

$$C_{var_{Server1}} = \frac{181W \cdot 24 \cdot 30}{1000} \cdot 0.0966\$ = 12.59 \tag{5.35}$$

$$C_{var_{Server2}} = \frac{385W \cdot 24 \cdot 30}{1000} \cdot 0.0966\$ = 26.77 \tag{5.36}$$

The profit contribution is comprised of the transaction volume minus the variable cost.

$$pc_{Server1,2} = tv_{Server1,2} - C_{var_{Server1,2}} \tag{5.37}$$

Server 1 has the highest profit contribution, however the acquisition cost are higher.

$$pc_{Server1} = 85.29\$ - 12,59\$ = 72.7\$ \tag{5.38}$$

$$pc_{Server2} = 57.27\$ - 26,77\$ = 30.50\$ \tag{5.39}$$

The profit contribution function should be a maximum. Nevertheless we must consider the constraint that the provider is going to spend maximal 30000$ for acquisition.

$$ProfitContribution : 72.7\$ \cdot x + 30.50\$ \cdot y \rightarrow Max \tag{5.40}$$

$$condition : 7612\$ \cdot x + 3000\$ \cdot y < 30000\$ \tag{5.41}$$

We can distinguish between two variants: purchase server 1 or purchase server 2. The number of servers is calculated by the following formula:

$$NumberOfServer = \frac{max.acquisitioncost}{C_{server}} \tag{5.42}$$

Variant 1: If the provider spends all the money for server 1, he is able to buy 3.9 servers.

$$NumberOfServer1 = \frac{30000\$}{7612\$} = 3.9 \tag{5.43}$$

This leads to a profit contribution of

$$ProfitContribution_{server1} = 72.7\$ \cdot 3.9 = 283.53\$ \tag{5.44}$$

Variant 2: If the provider spends all the money for server 2, he is able to buy 10 servers.

$$NumberOfServer1 = \frac{30000\$}{3000\$} = 10 \tag{5.45}$$

This leads to a profit contribution of

$$ProfitContribution_{server1} = 30.50\$ \cdot 10 = 305\$ \tag{5.46}$$

By buying ten servers 2 the provider optimizes the profit, because the profit contribution of variant 2 is higher.
Now we will make a calculation considering the power consumption cost. The calculation has a further constraint limiting the power consumption to 800 KWh. Thus, the profit contribution is:

$$72,7\$ \cdot x + 30,5\$ \cdot y \rightarrow Max \tag{5.47}$$

$$condition1 : 7612\$ \cdot x + 3000\$ \cdot y < 30000\$ \tag{5.48}$$

$$condition2 : 130,32KWh \cdot x + 277,2KWh \cdot y < 800KWh \tag{5.49}$$

The optimal profit is calculated by solving the equation using values of tables 5.5 and 5.6.

$$\frac{30000\$ - 7612\$ \cdot x}{3000\$} = \frac{800KWh - 130,32KWh \cdot x}{277,2KWh} \tag{5.50}$$

$$\begin{aligned} 277,2KWh \cdot (30000\$ - 7612\$ \cdot x) = \\ 3000\$ \cdot (800KWh - 130,32\$ \cdot x) \end{aligned} \tag{5.51}$$

The optimal profit contributions is reached with 3.44 servers 1 and 1.268 servers 2. *x=3,44 y=1,268* Thus, the profit contribution is:

$$3,44 \cdot 72,70\$ + 1,268 \cdot 30,50\$ = 288,76\$ \tag{5.52}$$

This server combination reduces energy consumption. However, the profit contribution is lower than the first one (305$), which is depicted by figure 5.5.

We further extended our example by assuming that the energy price increases by 3% per year.

Table 5.5: *Price per KWh*

year	price
1	0,0966$
2	0,099498$
3	0,102948$
4	0,1063644$

Table 5.6: *Power consumption comparison*

Server	Variation 1	Variation 2	Power consumption
Server 1	-	3,44	181W
Server 2	10	1,268	385W
Power consumption	2772KWh	799,79KWh	

As can be seen in table 5.7 the energy cost difference increases from 192,46$ to 211,84$ (Δ 19,38). Due to the fact that the transaction volume remains constant the profit contribution difference (Δ 16,24) decreases to -3,14. Within a time period of four years the

Table 5.7: *Variant comparison*

Variant	Year	Power con.	Price per KWh	Total	Δ pc
1	1	2772KWh	0,0966$	267,78$	
2	1	779,79KWh	0,0966$	75,32$	192,46$
1	4	2772KWh	0,1063644$	294,84$	
2	4	779,79KWh	0,1063644$	82,94$	211,84 $

efficient servers prove to be more profitable than the cheaper inefficient servers. Moreover, it has to be considered that some power consumption cost are fixed cost which reduce the payback period.

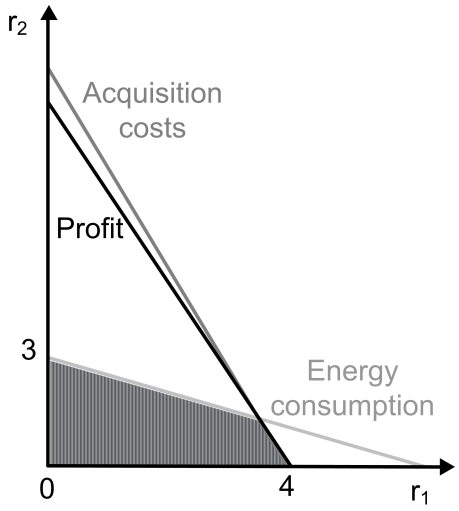


Figure 5.5: *Linear Optimization*

5.3.2 Optimal Server Combination

For a cloud provider who plans a new cloud environment it is necessary to define the number and type of servers to buy to get desired computational power at optimal cost. As described in section 5.1 we can choose servers with different performance per power characteristics based on the benchmark results of SPEC_Power_ssj2008 ®. We can use servers with higher or lower efficiency. That means more or less Server-Side-Java business applications performance per power. These results are stored in our model data store and can be used as input parameter to our model.

The number of combinations having distinct performance per power results for these servers can be calculated by $\frac{n!}{k!(n-k)!}$ where n is the number of server categories we can choose from the database (number of classes of performance/power results), and k is the number of different server types we use in the model. The number of combinations can increase dramatically, e.g. if the number of categories gets larger than five and the number of total servers is larger than one hundred, the number of combinations exceeds the billion.

With our model we calculate the fixed cost of server combinations and their behavior during variation of the combination of different servers with different cost and performance/power. As depicted in figure 5.6 the fixed cost are changing from efficient servers to inefficient servers depending on the combination. The result is normalized in percentage

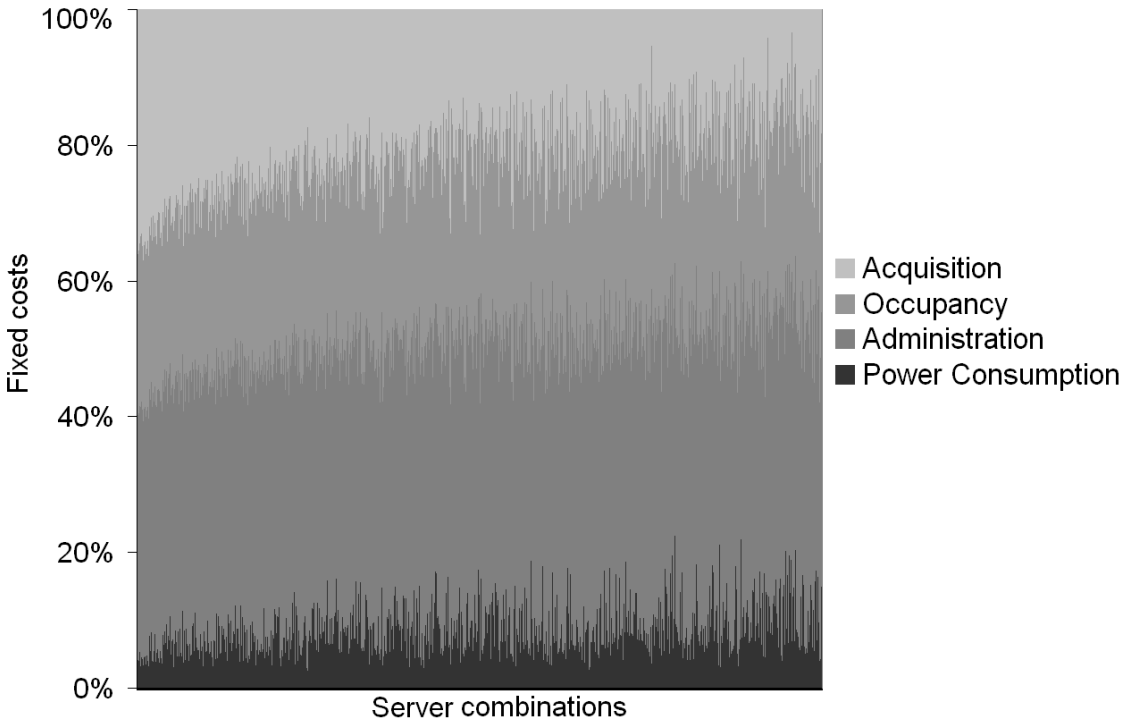


Figure 5.6: *Fixed cost versus server combinations*

to make the difference more evident. The same result can be used without normalization to choose the appropriate server combination for a specific limit of fixed cost.

6 An Autonomous Negotiation and Re-Negotiation Framework

In this chapter we introduce our novel generic framework for both, consumer and provider and describe all necessary components. We have carefully studied important existing framework models and concepts [82] [83] [84] [85] [86] [87] [88]. The novelty of our approach is the ability of autonomous negotiation based on our newly developed knowledge base. With this framework we are able to execute any kind of negotiation processes such as auctioning. The framework also enables simple and complex negotiation processes.

6.1 Framework Architecture

In this section we give an overview about the principal architecture of our negotiation framework. As depicted in figure 6.1 the framework is symmetric; each component ex-

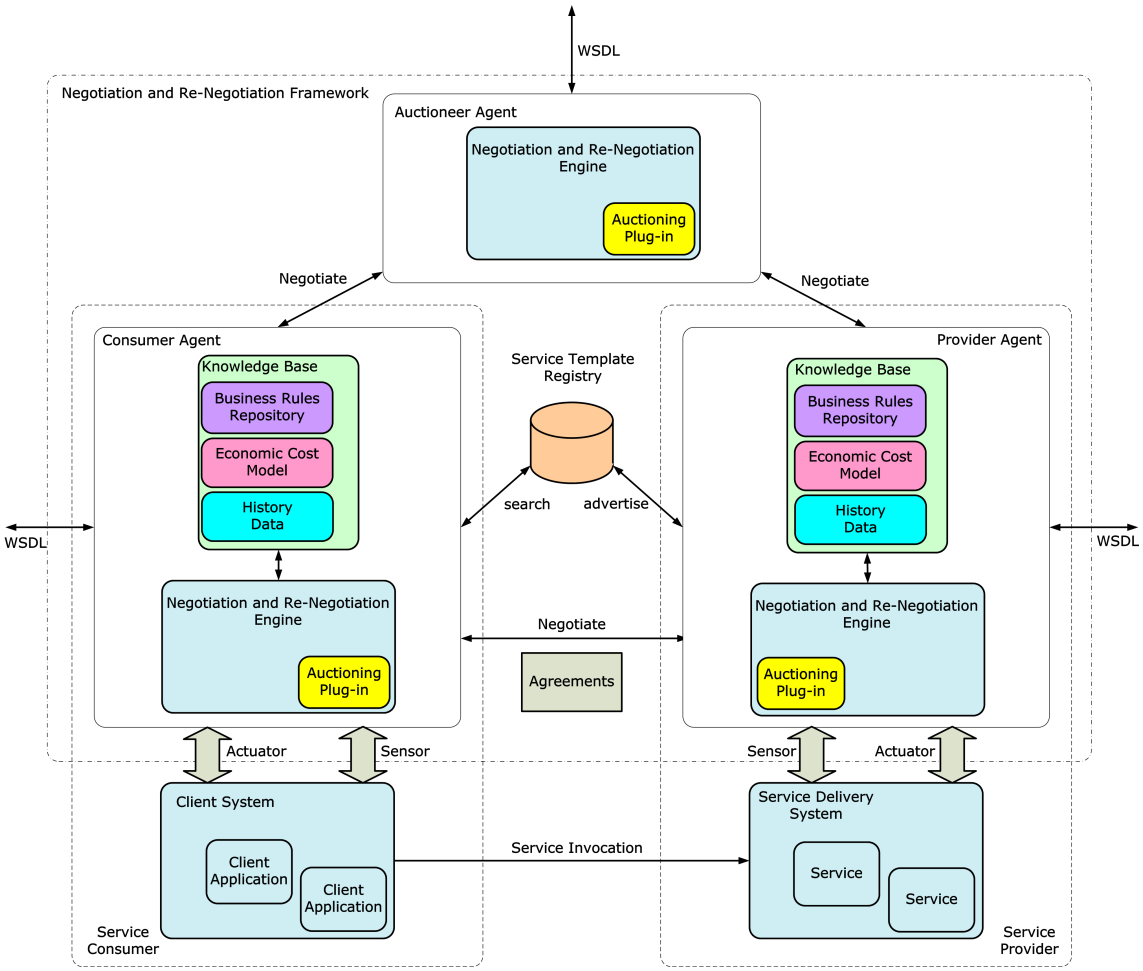


Figure 6.1: *Framework Overview*

ists for both service provider and -consumer. Please note that the framework supports

also many-to-many associations between consumer and provider. This is necessary for auctioning and biding.

6.2 Components of the Framework

The **Negotiation and Re-Negotiation Engine** is one of the main components of our framework. This engine is responsible for both, negotiation and re-negotiation, that means for the whole life-cycle of a Service Level Agreement. The negotiation engine is designed as an autonomic manager. The engine uses the knowledge base for all its decisions during life cycle of each SLA. To extend bilateral contracting an *Auctioning Plug-in* is used to support also auctioning and biding. The plug-in can be easily replaced to support different types of auction models. The sensors are responsible for monitoring the agreed QoS of the running services and applications. Actors can alter already running services.

The **Knowledge Base** consists of three parts, the *Business Rules Repository* where the participants (consumer or provider) store their own business rules which implement the specific business strategy of the participant. The repository uses decision trees and tables. The second part of the knowledge base is the *economic cost model*. In this cost model the cost of each production factor like computational power, disk space and network bandwidth are stored. The cost model distinguishes between fixed cost and variable cost. This is necessary to give correct answers for a decision e.g. if a provider has free capacity of a resource and a customer already running a service is paying the fixed cost and the provider wants to decide at which price the free resource can be offered. Existing cloud cost models neglect traditional economic principles. Cost models that consider variable cost are better suited for our economy situation today. This distinction is also necessary to implement or derive more accurate business strategies.

The economic cost model as knowledge base in the framework has to support all traditional economic fundamentals and methods described in the standard economic literature [89]. Traditional economics covers operating production factors, the production, sales theory, and investment and finance. These issues are necessary to allow applying all well known traditional economic methods to the cost model. In [90] we developed a comprehensive cost model based on variable and fixed cost for common cloud computing environments applying traditional economic methods. Based on this model we show that business strategies for both, cloud providers as well as cloud consumers can be derived. On the one hand this model makes it possible to design new cloud computing environments and also optimize existing clouds; on the other hand this model can also be used to give a clear answer for building internal cloud environments to IT Managers.

The third part of the knowledge base is *history data*. The history data stores past information. That means the experience each participant has made in contracts and running services, e.g. how many successful contracts have been made between consumer and provider, statistical data about the QoS of each provider, etc. This information is very useful for decisions during negotiation and re-negotiation.

The **Service Template Registry** is a registry where providers can store their offered services. The consumers can retrieve this registry for finding the best matching service they need. The Service Template Registry is used during first time establishing agreements. That means standard offers from the provider are stored. During Re-Negotiation the registry is not used.

The **Consumer- and Provider Agents** are the interfaces between the framework and an application or a human person. The interface is for initiating and controlling services, for changing running services, maintaining the knowledge base and for the auctioneer controlling of and interacting with the auctioning workflows. We use WSDL (Web Service Description Language) [91] as a protocol for the interface,

The **Auctioneer Agent** is controlling and interacting the auctioning workflow stored in the *Auctioning Plug-in*.

The **Protocols** used in the framework adhere to existing standards: WSDL for service description, WS-Agreement for order and offer description in the service template registry and WS Agreement Negotiation for the contracting process. The WS-Agreement Negotiation model defines the negotiation as a separate process. During this process service consumer and provider exchange information dynamically with the goal of creating a valid agreement offer that subsequently leads to an agreement [92]. Generally negotiation takes place prior to service execution. Re-negotiation is a reaction of one of the parties to the actual performance of the service execution.

6.3 Framework Patterns

We have defined framework patterns to classify typical business scenarios which can be handled by our newly developed framework.

Generally, we distinguish between two framework patterns. A conceptual overview of these two patterns is shown in figure 6.2 and 6.3.

- **Negotiating between Provider and Consumer**

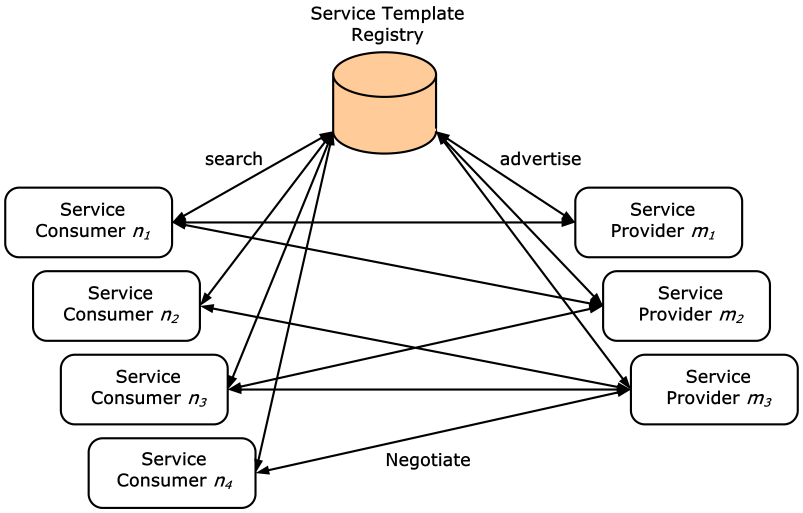


Figure 6.2: *Framework Scenario n:m Negotiating*

- **Negotiating between Provider and Consumer with Auctioneers**

The first pattern, the **n:m Negotiation Pattern**, is negotiating and re-negotiating directly between consumer and provider. Figure 6.4 depicts negotiation in more detail in which only consumers and providers participate. In this scenario $n_1 \dots n_4$ service consumers and $m_1 \dots m_3$ service providers are involved. The *Service Template Registry* is used by all service providers and consumers to advertise SLA-templates respectively to retrieve it. Between consumer and provider is an $n : m$ relationship, that means each consumer can have a relationship to any of the provider.

The second framework pattern (figure 6.5) is the **Auctioning Pattern**, where one or more *Auctioneers* are involved. The negotiating workflow (i.e. the auctioning type) is stored in the *Auctioning Plug-in* of the Auctioneer, Consumer and Provider Agent.

It is important to note that the framework supports each kind of mixture of the two negotiation types.

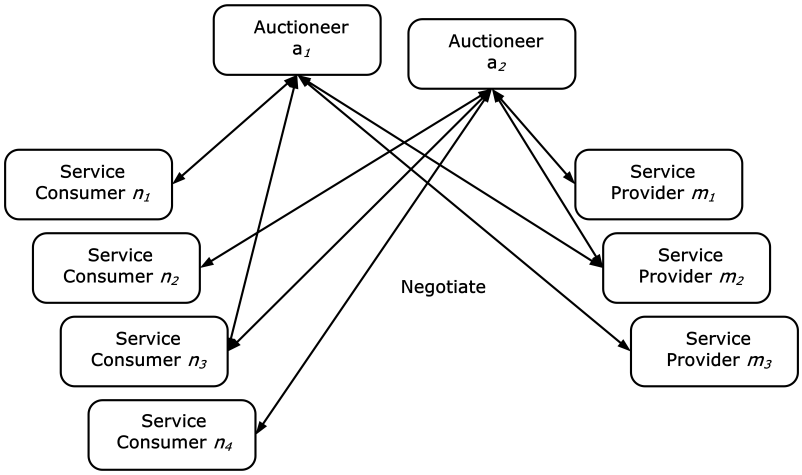


Figure 6.3: *Framework Scenario n:m Negotiating with Auctioning*

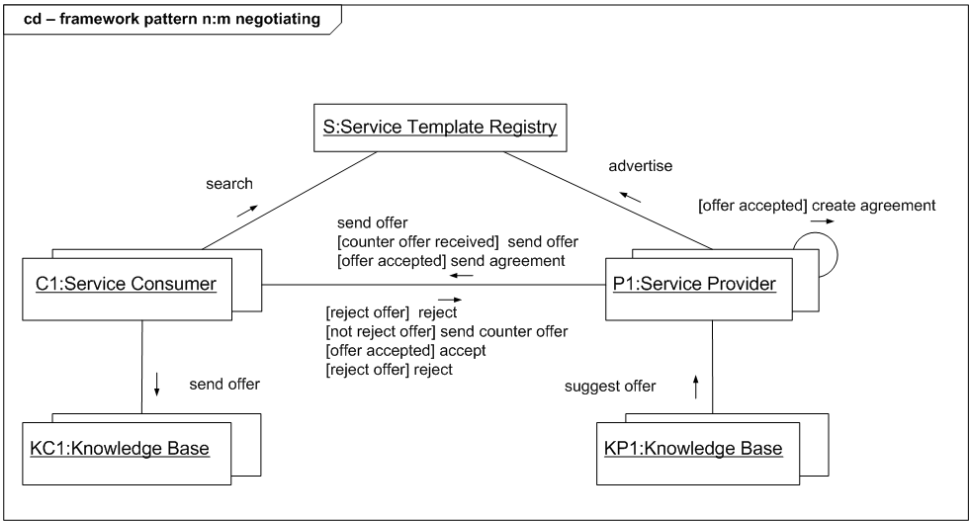


Figure 6.4: *n:m Negotiation Pattern*

Figure 6.6 shows a possible negotiation flow of the first type of negotiating (see figure 6.4). In this negotiation process four roles are involved. The service template registry acts as a mediator between providers and consumers. It is responsible for finding a matching provider for the consumer. The provider creates offers and the consumer searches for them. Both, consumer and provider, use a proprietary knowledge base for analyzing and creating offers. In figure 6.6 the communication between the knowledge base $KP1$ and provider $P1$ is simplified.

1. Provider $P1$ advertises his offer on service template registry S.

2. The consumer $C1$ looks at this registry S for matching providers. The registry suggests provider $P1$.

3. This provider creates the offers $O1$ and $O2$ based on knowledge base $KC1$ and sends these offers to the consumer.

4. As the consumers knowledge base advises to reject $O1$ the consumer sends a reject message to the provider. For offer $O2$ the consumer sends a counter-offer.

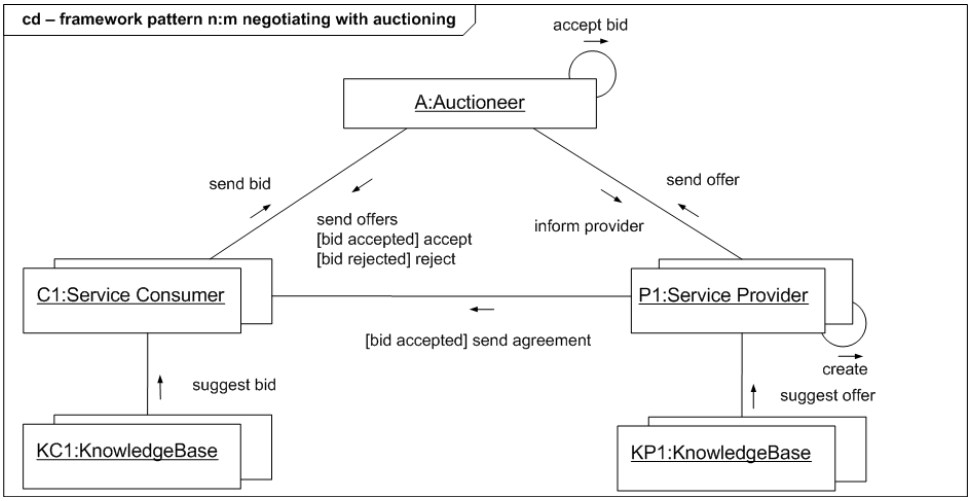


Figure 6.5: *Auctioning Pattern*

5. The provider analyses the counter-offer and sends a new offer based on the consumer's counter-offer.

6. The consumer checks this new offer and rejects it which triggers the provider to create and send a new offer based on the customer's counter-offer.

7. As the consumer's knowledge base considers this offer as acceptable the consumer sends an accept message to the provider who replies with an agreement.

Figure 6.7 depicts the sequence diagram of the second type of negotiating scenario, auctioning (see figure 6.5). In the auctioning based negotiation process four roles are involved. The auctioneer is responsible for the whole process including defining and controlling the auctioning rules, collecting all offers and bids, executing clear actions, etc. The provider creates offers and the consumer searches for them. Both, consumer and provider use a knowledge base for analyzing and creating offers.

1. **Offer Phase**: First of all the provider $P1$ creates an offer using his knowledge base and sends it to the auctioneer A.

2. **Bid Phase**: Auctioneer A collects all bids from the consumers $C1$ and $C2$. The consumers create their bids based on their knowledge bases.

3. **Quote Phase**: The auctioneer A accepts a bid (the quote) applying auction rules (e.g. Vickrey auction). After a quote is fixed, the provider is informed which creates a template for the agreement based on the offer.

4. **Clearing Phase**: The winning consumer $C1$ receives an accept message from auctioneer A and the agreement template from provider P, consumer $C2$ receives a reject message.

WS-Agreement is bundle of protocols used for SLAs and negotiation. WSLA is older than WS-Agreement and provides an XML-Schema used for describing WSLAs.

Figure 6.8 presents a code snipped of a WSLA document, which describes this situation. It shows a service level objective for hard disc space which is part of the obligation section. In the example provider $P1$ is the obligated party for the time period defined in the validity block. The expression section contains the content of the obligation. The obligated party must provide $HardDiscSpace$ from 30 to 90, whereas $HardDiscSpace$ is a SLA parameter defined with an appropriate metric in a separate section of the document.

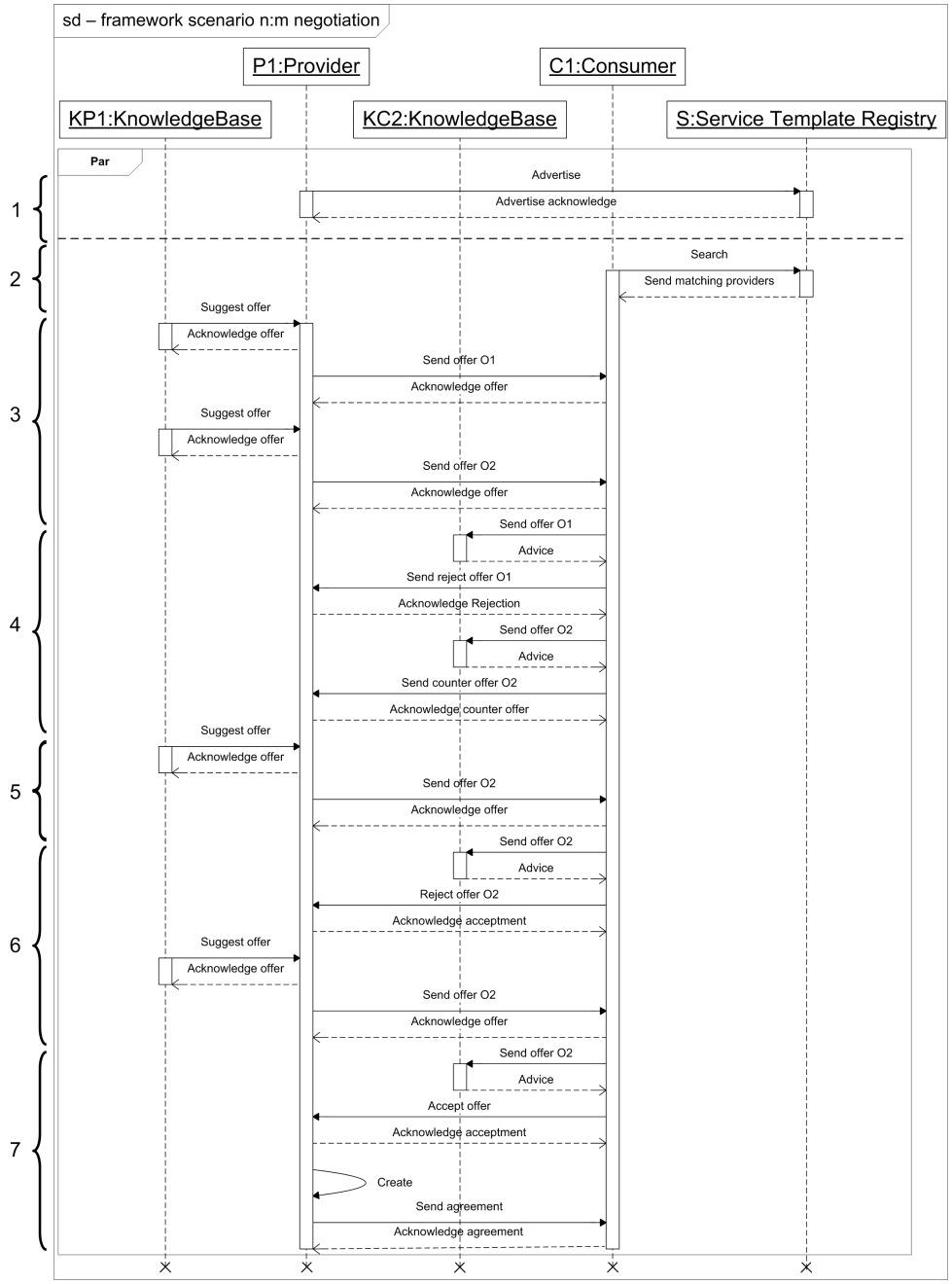


Figure 6.6: *This picture shows a running example of a possible negotiation flow.*

6.4 Consumer and Provider State Diagram

To show the different states of each participant we present a state diagram of both, provider and consumer in our framework.

Please note in our framework consumer and provider have the same state diagram. That means they execute the same states and event triggers. All states and triggers depicted in figure 6.9 are described as follows:

- **Running**
  In this state normal or idle operation is performed, e.g. running services after successful contracting.

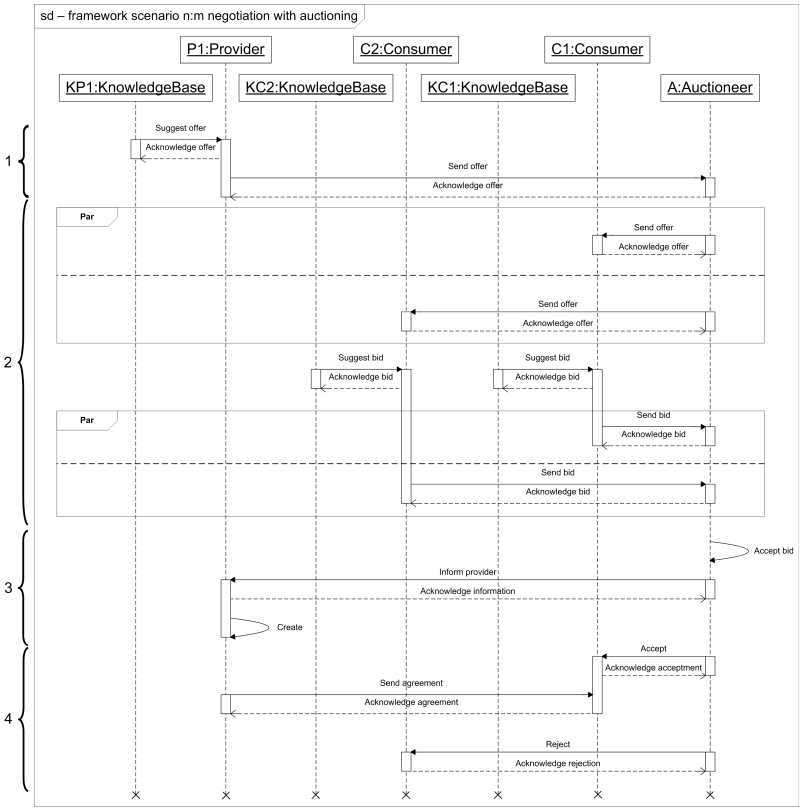


Figure 6.7: *The figure illustrates a sequence diagram for negotiation with auctioning*

```
<ServiceLevelObjective name="HDSpace">
    <Obliged>P1</Obliged>
    <Validity>
        <Start>2012-05-15T09:30:47Z</Start>
        <End>2015-05-15T09:30:47Z</End>
    </Validity>
    <Expression>
        <And>
            <Expression>
                <Predicate xsi:type="GreaterEqual">
                    <SLAParameter>HardDiscSpace</SLAParameter>
                    <Value>30</Value>
                </Predicate>
            </Expression>
            <Expression>
                <Predicate xsi:type="LessEqual">
                    <SLAParameter>HardDiscSpace</SLAParameter>
                    <Value>90</Value>
                </Predicate>
            </Expression>
        </And>
    </Expression>
    <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figure 6.8: *The figure illustrates a code snippet of a WSLA document*

- **SLA match search**
  The SLA match search is the state where both, provider and consumer try to match

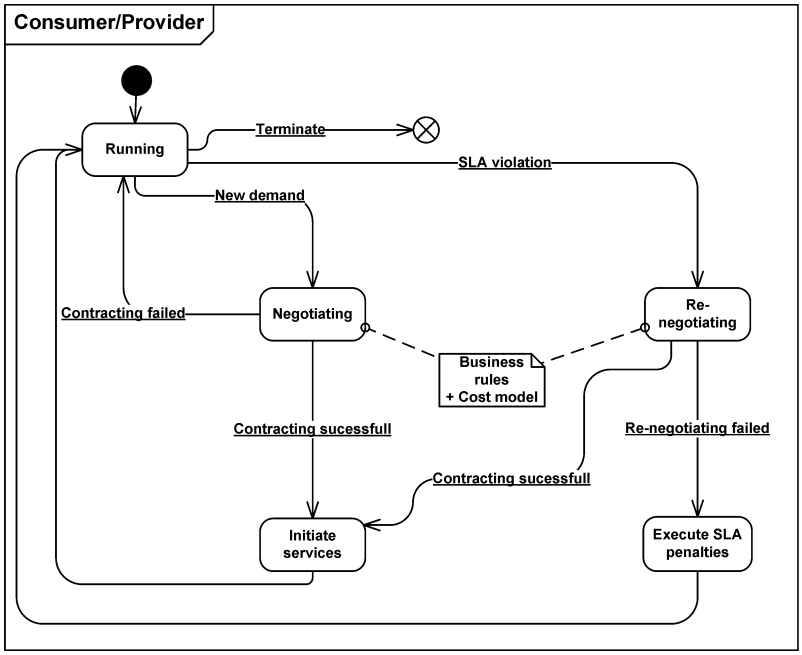


Figure 6.9: *State diagram for consumers and providers*

their SLA parameters. This is the initial state before the negotiating state can be invoked. In our framework this is a rough selection based on predefined matching rules.

- **Negotiating**
  After the initial state the negotiating state can be reached. The decisions necessary for negotiating are based on stored business rules and the embedded cost model.

- **Initiate services**
  After successful negotiating the services are initiated or scheduled if start time is different.

- **Re-negotiating**
  The re-negotiating state is only reached if a SLA violation happens or the consumer wants to re-negotiate because another provider offers a more attractive service. In this state both, consumer and provider, have the chance to avoid executing the contracted SLA penalties. As in the negotiation state the decisions necessary for negotiating are based on stored business rules and the embedded cost model.

- **Execute SLA penalties**
  This state is reached if the re-negotiating state failed for successful contracting.

7 Use Case Scenarios

This section describes several simulation scenarios by introducing basic implementation approaches and overviews.

7.1 Typical Business Scenarios

Services can be seen as virtual goods which are supplied by providers and demanded by consumers. So the fundamental market mechanisms for consumer goods can be applied to services, too. Basically, the market controls price and quantity based on the demand and supply [49]. A typical market model is depicted in figure 7.1. The numbers 1, 2 and 3 represent three different situations to which we will refer in the following paragraph.

Fixed priced services, without considering market mechanisms, lead to situations in which demand and supply differ as shown in figure 7.1: Setting a high price as shown in situation 1 leads to a supply exceeding the demand. We call this situation over-provisioning. A low price, as shown in situation 3, leads to demand exceeding the supply which we call under-provisioning. Both situations are inefficient as there is a gap between demand and supply. Thus, there is a need for a service-market in order to find the equilibrium where the demand matches the supply as shown in situation 2 in figure 7.1. Establishing a market prevents permanent under-provisioning and over-provisioning. Consequently, the market can be considered as a mechanism for finding an adequate price depending on current demand and supply. In order to establish a service market, virtual platforms where consumers and providers can publish their demands and supplies are necessary.

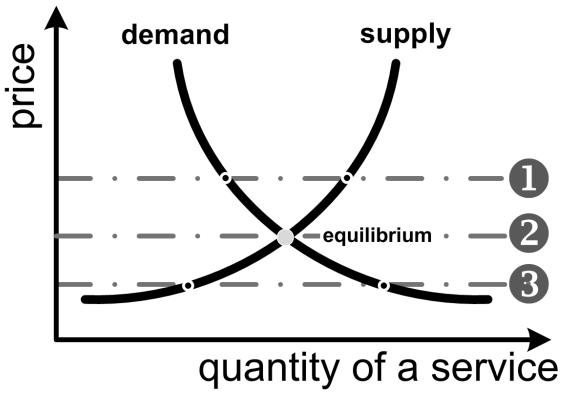


Figure 7.1: *Basic market mechanism*

Market Situations

An instance of a service market is shown in figure 7.2. The providers are enterprises offering services which are used by consumers or intermediates. Intermediates use the services in order to resell them or to offer new services composed out of other services.

The market mechanism explained in the previous section works, if several providers and consumers are attending the market. By considering intermediates as providers we can distinguish between the following four scenarios.

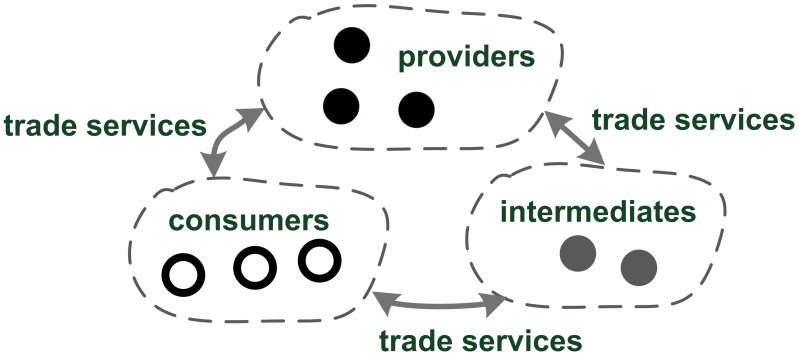


Figure 7.2: *Market instance*

- **n:m Scenario**

  In this scenario n consumers buy services from m potential providers. Neither a consumer nor a provider has enough power to influence the price-quantity relationship: If a provider increases his price, consumers will buy from other providers and vice versa. So the equilibrium is the result of the demand-supply relationship on the market. Such a situation is an example of an oligopoly [49].

- **1:m Scenario**

  In such a scenario, one consumer can buy a service from m providers. As there is no other consumer on the market, the consumer has a high influence on the price. This is because the providers want to maximize their profit. From a provider's point of view, the only way to maximize the profit is to serve the consumer. As the consumer can choose between different providers, the offer will be of low profit for the consumer. However, not selling the service results in a loss of the provider. This situation represents a monopole on the consumer side [49].

- **n:1 Scenario**

  Contrary to the 1:m situation, n potential consumers on the market want to buy a service from 1 seller. Thus, the provider can determine the price as the consumers have no chance to switch to other providers. The provider will choose a price which maximizes his profit. This situation represents a monopole on the provider side [49].

- **1:1 Scenario**

  In a 1:1 situation, the consumer as well as the provider have no other negotiation partner. As both, the consumer as well as the provider, have a monopole, the price will be a result of negotiation [49].

As [49] describes, only oligopolies enable the demand-supply driven price mechanism on markets. Oligopolies represent situations in which consumers can choose between providers and vice versa. To enable oligopolies, a service market is necessary where the market participants can communicate with each other. The bazaar style negotiation introduced in this paper is one way to enable market mechanisms.

7.2 Framework Architecture Implementation

Before we introduce an initial implementation, we want to detail the phases of negotiation. The implementation in this section is an initial prototype with basic functionality which should show the basic agent based paradigm. A implementation considering the basic negotiation paradigm is shown in the next chapter.

7.2.1 Phases of Negotiation

The framework developed in a previous chapter supports the discovery and negotiation of SLAs.

We detailed this phase into 5 sub phases depicted in figure 7.3. In the

- Preparation phase, the business rules for the business rules repository are discovered, formulated and stored. Such rule can e.g. determine which service parameters should be exceeded in case no provider can be found for a required SLA.
- In the Initiation phase, parameters for the negotiation are set, such as priorities of SLOs. These priorities can e.g. represent the order in which service parameters are exceeded.
- The Discovery phase encompasses the querying of repositories for adequate negotiation partners.
- The Negotiation phase itself depends on the negotiation style chosen.
- Finally, the results have to be validated. The process can be restarted at any point in order to get new results by changing parameters.

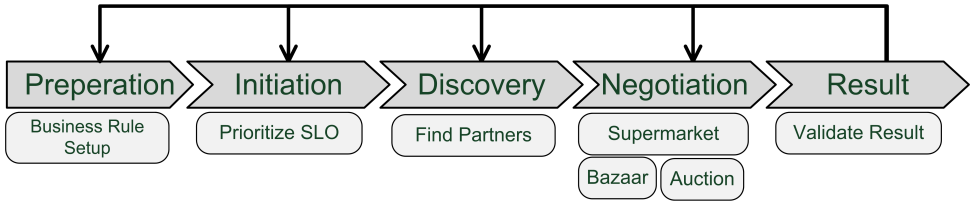


Figure 7.3: *Phases of discovery and negotiation*

Using the framework and its negotiation engine, several negotiation styles can be used such as the Bazaar style, which we have introduced in a previous section.

7.2.2 Agent based Implementation

Agents are software components which are able to solve problems in an autonomous way. Platforms in which several agents communicate and negotiate are called multi-agent systems and are used to solve problems in a distributed way [93]. The communication of agents has to be based on standardized languages so that interoperability of communication is ensured [94].

The goal of this thesis is to introduce an initial implementation of the conceptual framework described in 6 which is the basis for a service market place. Two main characteristics of the framework are the facts (i) that agents are used (ii) and that the agents are communicating directly with each other.

Instead of starting implementing the framework from scratch, we made a survey on already existing frameworks supporting agent based systems.

The most scientific papers describe negotiation frameworks, protocols or systems without introducing a concrete implementation such as [95] [96] or [97].

To the best of the authors' knowledge, Jade is the only comprehensive framework supporting agents as well as a complete communication infrastructure for the agents. Moreover, Jade is an implementation based on the FIPA standard which will be explained in the following subsection [98].

In several papers such as in [99] and [98] the Jade framework was already used for simple negotiations among agents. However, none of these papers considered service negotiation

in a bazaar style manner for web services. A knowledge base as described above was missing, too.

In the following sections we want to introduce the FIPA standard as well as the Jade framework in order to explain how Jade can be used to implement our conceptual negotiation framework. These sections should just give a rough overview.

7.2.2.1 FIPA Standard

FIPA is a standard maintained by the IEEE. Its goal is to standardize the communication of heterogeneous agents as well as infrastructures containing agents. In the following enumeration we want to highlight two critical artefacts of the standard which are important for our implementation. A detailed description of the standard would exceed the range of this paper.

- One key part of the FIPA standard is the standardization of the **communication messages** between the agents. The so-called ACL (Agent Communication Language) is a language specified by the standard. Within the ACL, types of messages, called communicative acts, are defined such as Request, Agree or Propose.

  The payload of a message can be delivered in an arbitrary format but FIPA suggests to use the FIPA-CL (Content Language). Furthermore, the FIPA describes several communication flows for e.g. English auctions.

- The FIPA standardizes the **agent management**, too. A reference model was published in which components for agent management were described resulting into a sketch of an agent based system architecture. We will refer to this architecture in the following section.

[100]

7.2.2.2 Jade

As Jade has implemented the FIPA standard, its core architecture is equal to the FIPA reference model. Figure 7.4 shows a high level view of base components of the Jade framework to which we will refer in this paper.

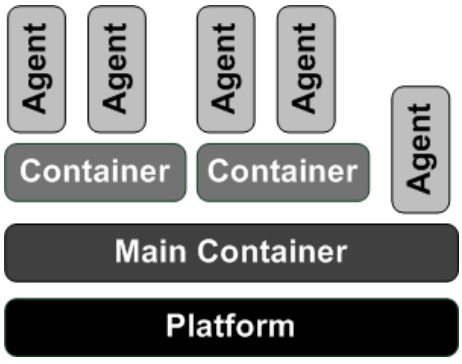


Figure 7.4: *Jade hierarchy*

- **Main container**

  A Jade platform consists of containers containing **agents**. One of the containers is the main container to which all the other containers have to connect. The main container is, among other things, responsible for managing the message flow between agents in different containers. Agents can be managed by the main container or by

simple containers. This is because the main container is a simple container with some additional components such as a so-called directory facilitator.

All the agents within a platform can communicate with each other, independent of which container manages the agent.

- **Container**

  Containers can be distributed over the network and provide basic services such as message transport for its agents.

- **Directory facilitator**

  The Directory facilitator is maintained by the main container. The directory facilitator is a yellow page service where agents can register their services and other agents can search for these services.

Figure 7.5 shows a possible instance in a Jade base negotiation system. There are three platforms whereby platform 1 has 3 containers. The agents communicate based on ACL messages and use the directory facilitator for searching and registering, in our case, services.

With this figure, we tried to introduce a mapping between the Jade framework and our conceptual framework described in section 6. We want to highlight which components of the Jade framework are already provided by the Jade framework and can be used for implementation. Therefore we coloured the components in 7.5 with the same color used in figure 6.1.

So the light blue marked consumer and provider agents in figure 6.1 will be implemented as Jade Agents.

The Jade directory facilitator in figure 7.5 corresponds to the service template registry in figure 6.1.

Of course, we will use the containers for the agent management and message transport.

The blue marked components in figure 6.1 as well as the knowledge base are not supported by Jade directly. Therefore we are forced to implement the knowledge base and the grey marked components by ourself. In the scope of this paper we will focus on an initial implementation of business rule repository of the knowledge base within the Jade framework. The other missing components are part of our further research.

The implemented framework has to fulfill the requirements described in section 6.1 which can be seen as high level functional requirements. Especially the non-functional requirements such as scalability and performance are essential in negotiation systems.

One way to audit these non-functional requirements is to do performance and scalability tests. Such analysis would exceed the scope of this paper and will be part of our future research.

Reviewing the Jade framework, it seems like the main container might be the system's performance bottleneck as it coordinates the messaging between the other containers. However, [100] states that Jade uses special cache algorithms and decentralizes platform operations so that it is well scalable.

Before we presenting an initial implementation, we take a closer look on the ontologies used for message exchange between agents.

7.2.3 Ontology for Agent Messages

One key issue of agent based systems is the communication between heterogeneous agents. Therefore, it is obvious, that they need a common language, a way of how agents exchange messages.

As [100] emphasizes, there are generally three ways of communication in agent based systems.

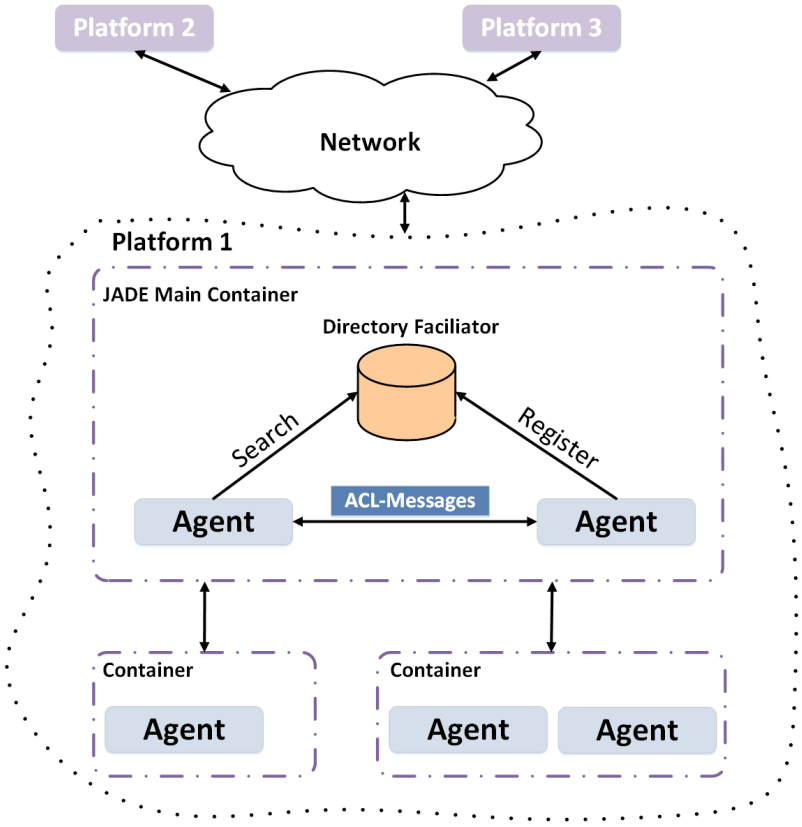


Figure 7.5: *Jade framework*

- **Strings**

  The most easiest way of communication between agents is to exchange messages based on strings. Strings are not bound to any programming language or operating systems which allows communication between heterogeneous systems. However, there are a couple of problems with strings especially for reading and interpreting the string correctly. With increasing complexity and size of messages problems become bigger.

- **Serialized objects**

  A more appropriate way of exchanging messages is the usage of serialized objects. So the receiver simply has to deserialize the content of the message instead of string parsing. Nevertheless, this mechanism has problems regarding interoperability: If the sender serializes a Java object, a non-Java based receiver will not be able to deserialize the object. Therefore this way of message exchange reduces interoperability. Addionally, the data sent is not readable by humans.

- **Ontologies**

  An ontology in the scope of Jade allows to define a vocabulary as well as a semantic. Using ontologies as basis for exchange, we can guarantee that the receiver will interpret the message correctly as the participants knowing the ontology have the same meaning to the symbols received.

  In Jade, the ontology describes the elements of the content of an ACL message.

For the agents in our framework, which form a service market place, we need an ontology supporting the negotiation of services. Currently, there are already standards which support negotiation of services like WSLA (Web Service Level Agreement).

We tried to create an Jade-ontology inspired by the base concepts of WSLA as it supports the exchange of service parameter intervals needed in a bazaar style negotiation.

The most important components of our ontology are depicted in figure 7.6. The WSLA represents an SLA. It contains *Service Parameters* which have a range.

One so-called predicate (will be explained bellow) is depicted in figure 7.6. The *Provide* predicate has a SLA and is used to ask if a provider/consumer offers/requires a certain SLA.

As you can see, figure 7.6 is divided into two sections grouping elements into concepts and predicates.

- **Concept**

  In the Jade Ontology, concepts represent things of the real world. This can either be a car or, like in our case, SLA.

- **Predicates**

  An agent may ask another agent if a certain proposition is true. In our case, Provides is such a predicate. A buyer agent may ask a seller agent if it is true that he offers a specific SLA.

- **Agent Actions**

  This section is not shown in figure 7.6. Agent Actions are a form of concepts which represent actions that can be performed by an agent.

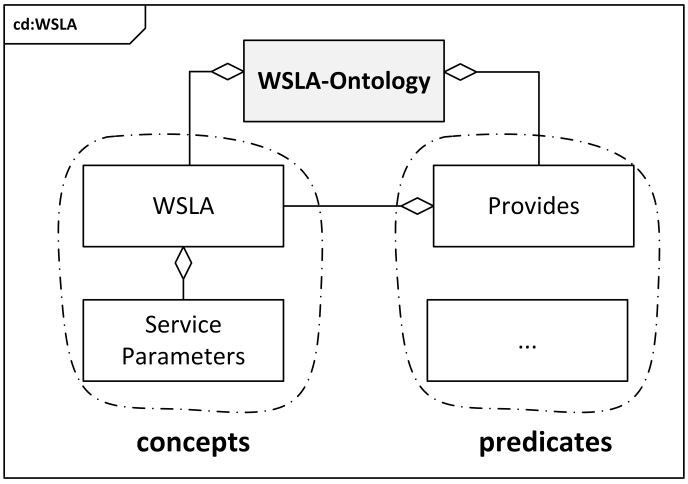


Figure 7.6: *Jade ontology for agent messages*

In Jade, the ontology elements are realized as Java classes. So we created e.g. a class named WSLA containing a list of service parameters. An instance of this class can be sent via ACL to another agent. If the agent knows the ontology, he is able to read the message content and interpret it correctly.

The ontology shown is used in our initial implementation. Of course, the introduced ontology is far away form being complete. However, it is comprehensive enough to execute our scenarios in the following sections.

7.2.4 Market Scenario

As described before, we can use several components of the Jade framwork. However, some components are not supported by Jade. In this thesis, our initial implementation based on Jade should achieve the following goals:

- Jade provides the basic infrastructure for the agents which should be used. We want to create a buyer agent requiring services and a seller agent, called provider, offering services using the ontology explained before.
- Both types of agents should have a knowledge base as described in chapter 6. In this thesis, we will focus on the business rule repository of the knowledge base.
- Jade provides a framework for agents. However, there is a need for an application of a buyer, also called consumer, starting the agents. Moreover the agents must report their result to somebody. These requirements are without the scope of the Jade framework and are highlighted in this section.

To clarify our goals, we developed a market scenario which should be supported by our initial implementation.

7.2.4.1 Scenario Description

Assume that a consumer has a video or pictures. As shown in figure 7.7, the consumer wants a workflow which compresses the pictures or videos and stores them on e.g. a cloud. Therefore the consumer looks for one or two providers offering services of the required workflow.

Both services have several parameters. For simplicity reasons, we consider one parameter per service in this paper.

As described in previous sections, agents should be able to find providers and negotiate SLAs with them without human interaction. Thus, the search for matching providers for each service of the workflow is done by agents. Regrading the workflow illustrated in figure 7.7, one agent searches and negotiates with providers for a compressing service and a second agent looks for a provider offering storage. Both agents store their results into central repository, the blackboard. All the agents' results are stored on this blackboard, which is the decision basis for the consumer. A short introduction of the blackboard is provided in a following section.

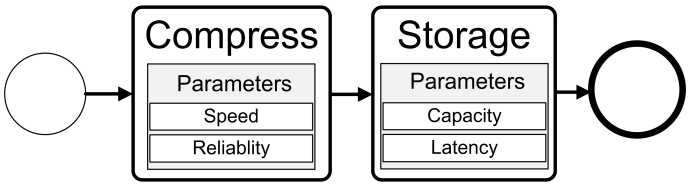


Figure 7.7: *Sample workflow*

7.2.4.2 Architecture Overview

In the following section, an overview of our implementation architecture, based on the market scenario, is described. We look at this implementation from a consumer's point of view, but the implementation can be used for the provider, too.

Figure 7.8 illustrates the implementation whereas the numbers in the figure correspond to three steps. Continuing the example described before, a consumer searches for two services. Therefore he starts an agent for each service and passes the requirements to them. This is illustrated in figure 7.8 with 1. Agent 1 looks for a provider offering the compression whereas agent 2 looks for a service offering the storage.

As the figure illustrates, the agent use rules. So some part of the agent's autonomous behaviour is implemented in business rules and evaluated by rule systems. We used the rule based approach for two main reasons [101]:

- Rule based systems have benefits if decision making is complex. Usually complex decisions include a lot of conditions determining if a rule is executed or not. Within rule based systems such condition-event relationships can be defined declarative via business rules.
- Rule based systems make it possible to add or remove rules from the knowledge base which increases flexibility.

As figure 7.8 depicts, agent 2 contacts and communicates with the corresponding providers. This situation is marked with 2.

Before the agent was able to contact the providers, he used Jades directory facilitator for finding the providers. During negotiation the agents can access and write their results to the blackboard which we will describe in the following section. In figure 7.8, the blackboard is the database shown in the cloud. This situation is marked with 3.

As both, the agents as well as the client have to write and read from the blackboard, its accessibility is critical. Generally, there are several implementation possibilities for the blackboard.

- Due to the fact that the data retrieved by the agents has to be made persistent, our initial approach was to create a simple database such as H2. However, this would force the agents to gather the necessary database drivers for accessing the database. Moreover, this approach faces the risk that the agents can write data into the database without validating the data before.
- Therefore, we decided to use a REST (Representational State Transfer) web service for accessing the database. This solves the problems stated before: The agents just need to access the web service. So they do not have to care about database drivers. Additionally, this simplifies the access to the database of heterogeneous agents. Further, validation of the data can be done in the web service.

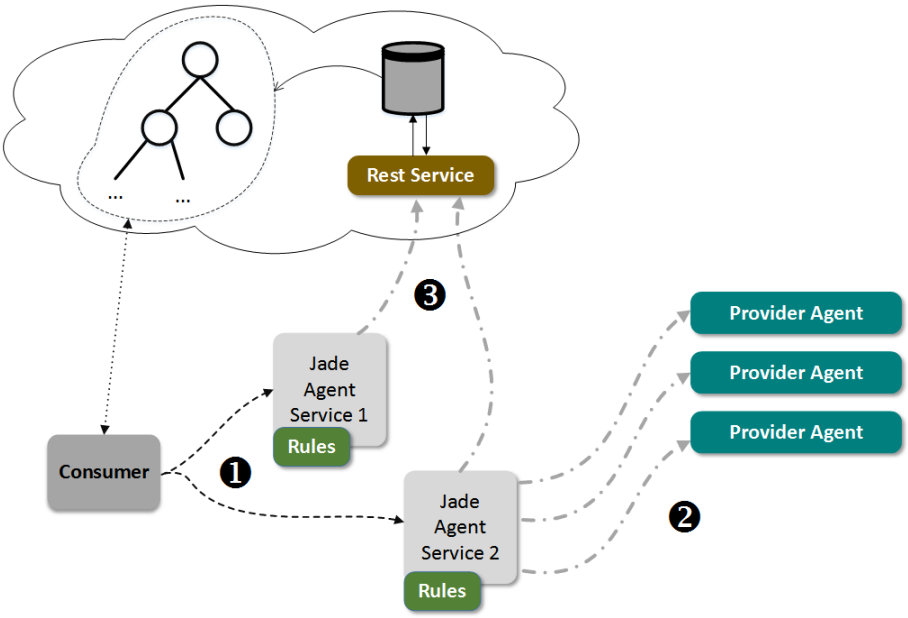


Figure 7.8: *Jade based implementation*

After all agents have stored the negotiation data into the database, the client has to access it in order to make a decision. The database is used to implement the blackboard approach.

7.2.4.3 Blackboard

The blackboard approach origins from artificial intelligence. It is defined as a process where different expert knowledge is added to the solution in a stepwise manner. In our situation, the agents add their negotiation results of services for the desired workflow to the blackboard.

As proposed in [102], the negotiation results, which will be the offers from the providers, can be represented as a tree as depicted in figure 7.9. For simplicity reasons, we neglected the price in this figure. As each service of a workflow has parameters, each service can be depicted as tree. The tree in figure 7.9 shows a possible result of the compression service. Obviously, the compression service has two parameters: reliability and speed. Each path from start to the leaf of the tree is one found service. One such service is marked with a grey line. This service has a reliability of 30 and a speed of 20. As there are three different paths in the tree, we can choose between three offers.

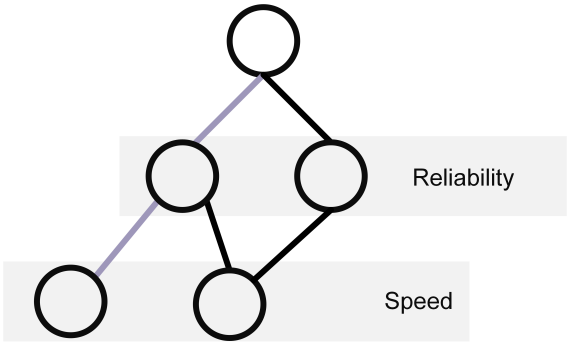


Figure 7.9: *Decision tree representing negotiation results*

Utilizing the tree, we can map the problem of finding and adequate service to finding the shortest path from root to a leaf. Based on the information given, several graph algorithms and heuristics like A* or Dijkstra can be used. [103] has introduced several utility and standardization methods to find a solution with the highest utility based on such trees.

7.2.5 Bazaar Style Negotiation Initiation Types

We have not yet explained how the negotiation itself works in the implemented framework. This is a non-trivial issue as there are a lot of possible forms of bazaar style negotiation like e.g. the different forms of auctions. Describing all these forms of bazaar style negotiation would exceed the range of this paper.

A precondition for negotiation between a consumer and a provider is the definition of the messages exchanged. In the supermarket approach, the consumer searches for an offer and sends a purchase order to the service provider if the offer is appropriate for him. In a bazaar style negotiation, there are much more possibilities. In this section, we focus on the negotiation initiation phase were we distinguish between two ways which we call consumer-first approach and provider-first approach. These two types will be explained below.

A sensible parameter in an offer is the price. Unlike a typical service parameter, the height of the price is a summary of other service parameters influenced by the providers business and pricing strategy. The compression service explained above has the parameters reliability and compression rate. These two parameters are independent which means the provider can offer a slow compression with a high reliability and vice versa. If the user wants to reduce the price of his SLA, he will reduce at least one of the parameters.

In the initial phase, the service parameters are ranges. As the price is influenced by other service parameters which have not yet determined, it is not useful to use a price interval within a SLA. However, it might be useful to determine an upper price/lower price range from a consumer/provider point of view. In the following, two service negotiation initiation types in a bazaar style market are introduced.

- **Consumer-first approach**

  The message flow in a consumer-first approach is the following. The consumer sends his desired SLA including ranges for his service parameters to the provider. The provider executes a matching process by checking the SLA and terminating the negotiation with a refuse message if he cannot supply the SLA. In case of a refuse, at least one interval of the consumers SLA is not overlapping with corresponding interval of the provider. The negotiation terminates too, if the price limits are not matching.

  If the consumer's desired SLA can be supplied, the provider sends his SLA consisting of service parameter ranges offer including his ranges to the consumer. The consumer looks at the providers SLA and sends a **concrete offer** to the provider. A concrete offer contains a SLA containing concrete values for the service parameters and a price the consumer wants to pay. So in this interaction, the consumer knows the provider's ranges and can make a concrete offer. As the consumer makes the first offer, we call this interaction the consumer-first approach.

- **Provider-first approach**

  Unlike in the consumer-first approach, the provider does not send his ranges to the provider after the matching process succeeds. Instead, the provider sends a concrete offer including a price. Therefore we call this approach the provider-first approach. In this case, the provider does not have to reveal his service ranges as the provider is generating the offers.

Of course, there will be a lot of different other forms how negotiation can be initiated. However, these are two basic approaches. In both cases, the negotiation continues with e.g. a sequence of offers and counter-offers until both parties agree or one of the parties terminates the negotiation. The negotiation itself can be done in various ways. Several of them are defined in the FIPA-standard such as English auctions.

7.2.6 Implementation

As an initial implementation, we implemented the scenario illustrated in figure 7.8 in Java. We used Maven as build tool, Jersey for the web service, H2 as database and the Swing API for the GUI of the buyer.

7.2.6.1 A First Simulation with the Negotiation Framework

Precondition for starting a negotiation is to find at least one matching provider. Thus, the consumer is interested in answers to the following questions:

- What is the probability of finding a provider offering a SLA matching my request?
- Which service parameters of the SLA, and to what extent, have to be adapted in order to find at least one matching provider?
- How many providers should I contact in order to be practically sure to find at least one matching provider?

In 4 we presented answers to these questions by defining an analytical model forecasting the probability of finding appropriate providers for a requested SLA. A twofold approach by defining a theoretical model and proving its applicability by practical simulation was followed. Further, an algorithm optimizing a consumer's SLA in order to practically ensure finding at least one appropriate provider was presented. To omit redundancy, we focused our analysis on the consumer's point of view. However, all answers given can be applied to the provider as well.

As shown in figure 4.1, the probability of finding a provider offering a service parameter which matches the consumer's request is equal to the probability that these two intervals are overlapping.

As already explained, for forecasting the probability of finding a matching partner (consumer or provider) we need both, the requested SLO parameter ranges of the consumer and the offered service parameter ranges of the provider. Unfortunately, the service parameter ranges of the providers can change quickly and no repository containing all this data in real-time is available in practice. Therefore we were forced to make assumptions about the provider ranges: In our simulation, which is described in 4, the length as well as the position of the *provider intervals* are generated randomly. Thus, the consumer's SLO parameter ranges are covered by the providers with the same probability, independently of the ranges' position and the SLO parameter type.

The simulation approach described in 4 was simulated in a simple Java program, but was not simulated in a framework with agents and Jade. Therefore we rebuilt such a scenario and checked, if the results complied with the results of the forecasting framework introduced in 4.

7.2.6.2 Simulation Flow

In our simulation, we search for a compression service containing two services of length 20: The reliability should be between 10 and 30, whereas the speed should be between 50 and 70. In both cases, the market minimum is 0 and the market maximum is 100. The exact units are neglected.

We did the implementation in our Jade based infrastructure in which agents communicate by using the ontology described in a previous section. We developed provider agents, offering the compression and storage service with random generated service parameters as described in 4.

We also developed a consumer agent. First of all, the consumer agent searches in the directory facilitator for providers offering the required service. The directory facilitator returns the providers who generally offer the service, but it does not check if the intervals are matching. So the consumer agents contact the providers returned by the directory facilitator in order to check if the service parameter ranges of the required service are matching.

In our simulation, the consumer agents contact one provider after the other until a matching provider is found. This experiment is done 100 times to get results comparable to those in 4.

Figure 7.10 shows a screenshot of our implementation. In this sequence diagram, the agent *buyer 1* has already contacted the directory facilitator. So the agent *buyer 1* contacts the first provider returned by the directory facilitator. The provider does not offer a SLA with the required service parameters. Therefore he sends a refuse message which triggers the buyer to contact the next provider. This provider can provide the SLA and sends an agree message.

In our simulation, 12% of the first asked providers were matching as table 7.1 shows. The table consists out of the simulation part and the prediction part. The prediction part shows the expect results using the prediction model presented in 4. The bold numbers in

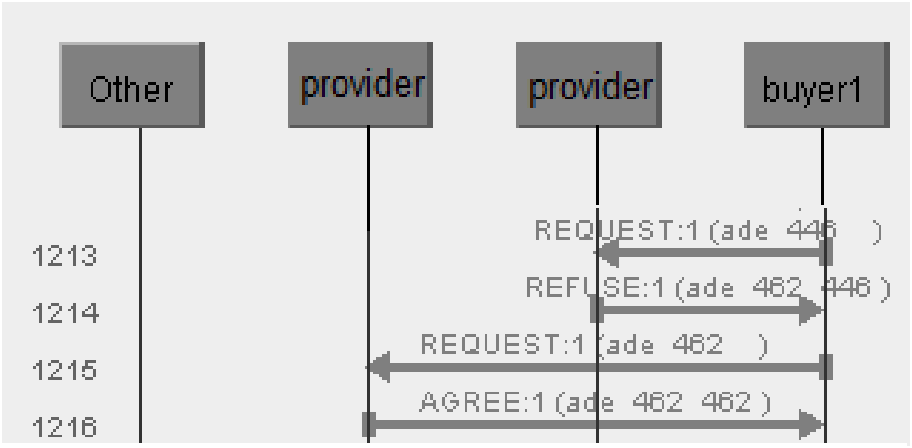


Figure 7.10: *Provider contracting*

Prediction

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.20	0.36	0.49	0.59	0.68	0.74	0.79	0.83	0.86	0.89
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0.92	0.93	0.95	0.96	0.97	0.97	0.97	0.98	0.99	0.99

Simulation Result

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.12	0.29	0.43	0.52	0.61	0.64	0.69	0.78	0.82	0.84
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0.89	0.93	0.96	0.98	0.98	0.99	0.99	1	1	1

Table 7.1: *Predicted results and simulated results*

row 1 and 3 like 1, 2 is the number of the provider contacted, the non-bold numbers in the rows 2 and 4 are the percentages of finding at least one matching provider. Thus, the data can be read as follows: contacting 1 provider, the probability that this provider is matching is about 20%. Contacting 2 providers, the probability that at least one of them is matching is about 36%.

These numbers are based on equation (7.1) which was developed in 4. It returns the probability that a provider matches (p), given the consumers interval length which is denoted with x.

For a consumer interval of length 20, the probability of finding a matching provider is about 0.449. If we search for a SLA consisting of two services like the compression service, we can simply connect their probabilities as explained in 4. This leads to a probability of approximately 0.20. The result is depicted in table 7.1 in row 2, column 1. The other numbers are calculated based on binomial distribution. A detailed explanation of this probability calculation can be found in section 4.

$$\begin{aligned} p_x &= 0.00688667 \cdot x + 0.31133315 \\ p_{20} &= 0.00688667 \cdot 20 + 0.31133315 \approx 0.449 \\ p_{20,20} &= 0.449 \cdot 0.449 = \approx 0.20 \end{aligned} \tag{7.1}$$

In figure 7.11, our simulation result is depicted as the lowest one of the grey lines (simulations), and the prediction is depicted as black line. As you can see, the average of the simulations seems to be approximately the predicted black curve.

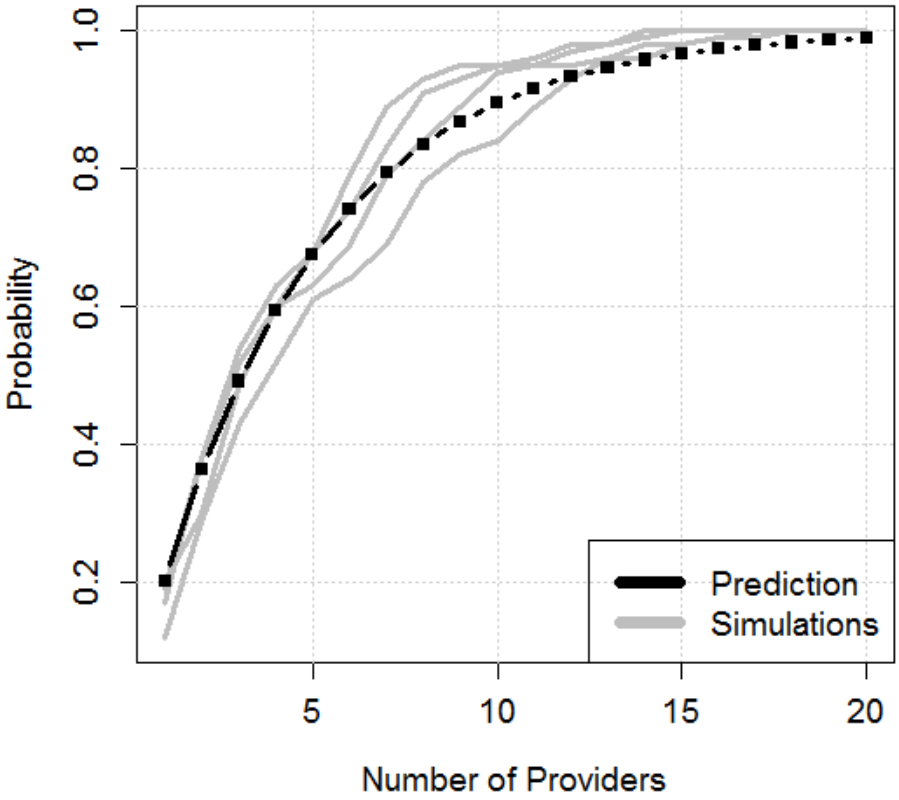


Figure 7.11: *Simulation and prediction of finding matching providers*

7.2.6.3 Utilizing Business Rules in Agent

In a further scenario, we extended the agents by using business rules in the agent. Therefore we used Drools.

Rules can be used for e.g. the following scenario: A consumer requires a service and contacts all providers returned by the directory facilitator. Unfortunately, non of the providers offers a matching result. With business rules we could implement a behaviour to extend the intervals of a SLA in order to find at least one matching provider.

We implemented such a scenario with the above introduced framework as depicted in algorithm 4.

We explain the implementation augmented with the business rules utilizing the process depicted in figure 7.3.

- **Preparation and Initiation**

  For an algorithm exceeding intervals there are basically two main questions:

  - Which intervals should be exceeded?
  - To what extent should the interval(s) be exceed?

  Another issue is to determine where to store the parameters answering these questions like e.g. the size an interval is exceeded. This can either be in a database or in the business rules.

Algorithm 4: Extending intervals

```
Data: Providers returned by the directory facilitator
boolean providerFound=false;
while !providerFound do
    for each provider P do
        if P matches then
            add to blackboard;
            if !providerFound then
                providerFound=true;
    if !providerFound then
        executed business rules;
```

We decided to make the following distinction:

- The rules decide which service parameters intervals should be exceeded because this can be a complex decision including several conditions making it appropriate for a rule based solution.
- The size an interval is exceeded is stored in a database as we decided for our example that this size will be constant for each service parameter.

A code snipped of our rules is shown in listing 7.1. The rule can be read as *IF service parameter Reliability is already exceeded and if service parameter Storage is not exceeded yet, exceed it.* So the business rule triggers the interval excess but it does not define how much. This is stored in a database.

Listing 7.1: *Simple business rules*

```
rule "Exceedation Rule 1"
    when
        sp : ServiceParameter
         (IsExceeded == true &&
         ID==ServiceParameterID.Reliablity)
        sp2 : ServiceParameter
         (IsExceeded == false &&
         ID==ServiceParameterID.Storage)
    then
        sp2.exceed();
end
```

As the rule implicitly contains the priority, we merged the two phases Preparation and Initiation.

- **Discovery**

  If the client triggers the negotiation, agents are started for each service of the required workflow. Each agent contacts the directory facilitator to find matching providers.

- **Negotiation and Result**

  Afterwards, the buyer agents can contact the provider. In our case, they ask provider agents if they offer a matching service. The provider returns an accept message if he can offer the required SLA. In all other cases, a refuse notification is returned. The results are persisted via a web service by the agent.

Figure 7.12 shows a screenshot of our implementation. It shows the client GUI where the client can set his desired SLA. After the agents have done negotiation, the result is shown as a tree which we described above. Based on algorithms and heuristics, the most appropriate service can be found.

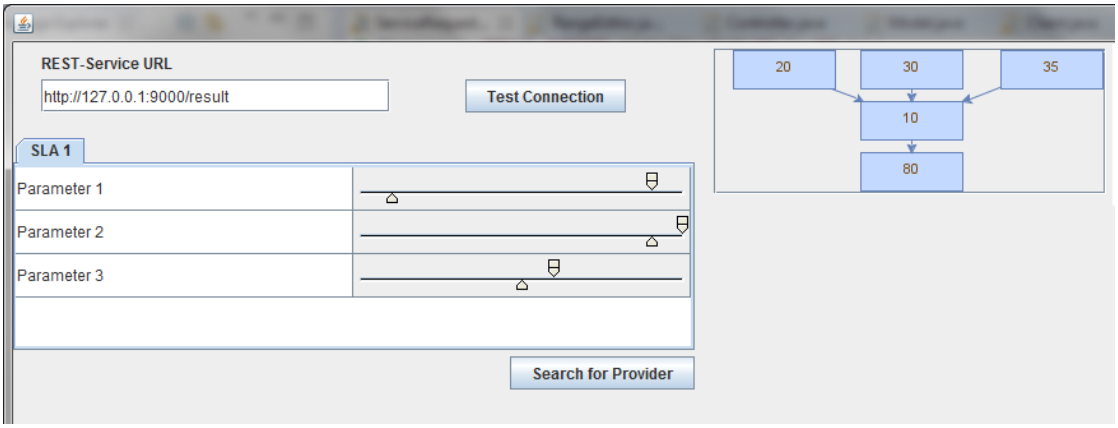


Figure 7.12: *Screenshot of client GUI*

8 Simulation and Analysis in CloudSim

Before starting describing the CloudSim scenario, we want to clarify some scenario characteristics. We assume, that all consumers need virtual machines for the exact same time. So offers for virtual machines differ in price but not in time of use.

The resources of a host are called physical resources whereas the resources of the virtual machine are called virtual resources.

In this thesis we use the terms bidder and consumer synonymously. A consumer can be represented by a broker.

8.1 CloudSim Framework

The CloudSim framework introduced in [54] was developed to model clouds. Based on these models, several allocation mechanisms, which we will explain in the following paragraph, can be simulated. Figure 8.2 shows some of the basic components used in CloudSim.

In CloudSim you can model so-called datacenters. A datacenter consists of one or more hosts whereby each host has several resources such as processing power, storage and ram. In this paper, we will only consider these three resources. The processing power is provided by processors and measured in millions of instructions per second (MIPS). Ram as well as storage are measured in mega bytes (MB).

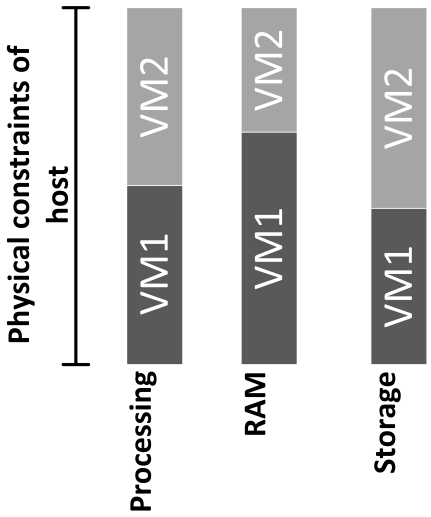


Figure 8.1: *Two virtual machines on a host*

In this section, one datacenter represents one cloud provider.

Virtual machines run on hosts and are characterised by the same properties as the hosts: processing power, storage and ram. A host can run several virtual machines as illustrated in figure 8.1. In this figure a host runs two virtual machines and allocates its physical resources to them. Of course, a host cannot allocate more virtual resources to virtual machines as physical resources are available.

A virtual machine is owned by a consumer. The consumer is represented by a broker which is responsible for finding adequate virtual machines.

8.1.1 Allocation Mechanism

CloudSim offers several allocation policies which can be extended or replaced. Some of these allocation mechanisms are explained below. The numbers in figure 8.2 correspond to the numbers in the following enumeration.

1. The VM Allocation Policiy is responsible for assigning virtual machines to hosts. The default policy is the First-Come-First-Serve policy [54]. So if a datacenter has to run a new virtual machine, one of its host after the other is evaluated until a host is found which has the physical capabilities to run the virtual machine. In all our following examples we will use this policy.

2. A virtual machine can specify the processing power as well as the number of processors. The Processor Allocation Policy is responsible for the mapping physical processors of the host to the processors in virtual machines. CloudSim offers two different policies [54].

   The space shared policy assigns each physical processor to one virtual processor. Using this policy, a host with two processors is not able to run a virtual machine with three or more virtual processors.

   Contrary to the space shared policy, CloudSim offers the time shared policy. Using this policy, each virtual machine gets a time slice of a processor. So one physical processor can be mapped to several virtual processors. Of course, they have to share the capacity.

3. The third allocation policy represents the focus of our research: the assignment of virtual machines to consumers. For this allocation, several economical aspects have to be considered. Interesting questions regarding this policy are: (i) *Q1*: Who gets the virtual machine if serveral consumers want to buy it? (ii) *Q2*: How is the price determined?

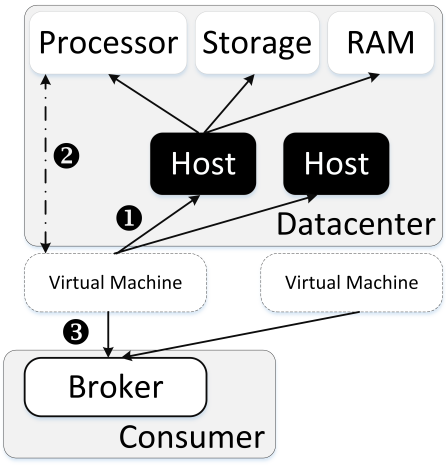


Figure 8.2: *Simplified CloudSim structure*

8.1.2 Simulation Flow

CloudSim's simulation flow is based on events [54]. Events can be considered as messages which have a well-defined structure. Basically, an event consists of source, destination, tag and data. Of course there are more fields such as delay. In this thesis, these fields are neglected.

The source represents the sending entity whereas the destination represents the receiving entity. Entities is a general term used for all components which are able to send and receive messages such as datacenters or brokers. Events can be sent to other entities as well as to the sending entity itself. The tag can be interpreted as the message type which indicates what action the receiving entity should do. The data field can contain any data which the receiving entity can use.

The simulation flow is organized in phases. Before simulation starts, each entity can submit events to the simulation. After all events have been collected, the events are processed. By processing an event, new events can be created. Figure 8.3 shows such a scenario. The events which to be processed are stored in a so-called deferred queue. By processing *Event A* three new events are created: *Event B*, *Event C* and *Event D*. Theses events are temporarily stored in the future queue as soon as all events from the deferred queue have been processed, the events from the future queue are copied to the deferred queue and processed. If both queues are empty, the simulation is finished. For a detailed description of the simulation flow, read [54].

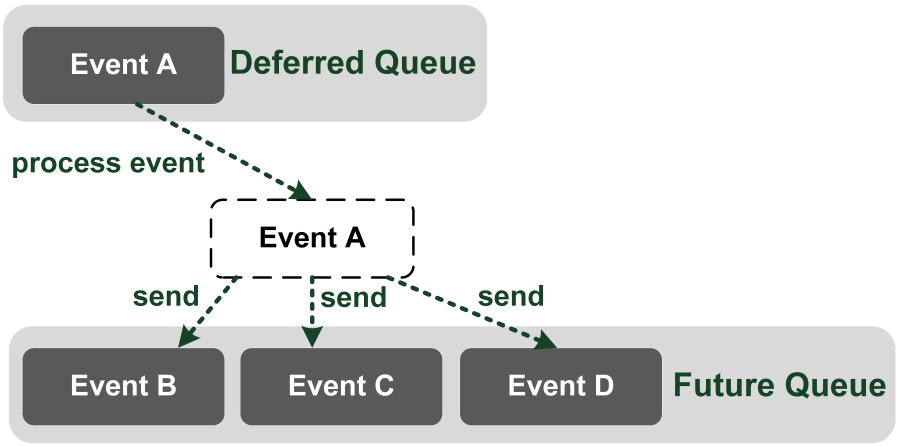


Figure 8.3: *CloudSim simulation flow*

8.2 Virtual Machine Auction

[104] have already implemented a double auction using SimCloud. Our goal is to implement a bilateral negotiation based on the bazaar style which we described in our previous research in [105] and compare it with typical auctions. In order to make our bilateral negotiation comparable with auctions, we have to implement our own auction using similar decision models.

First, we want to introduce our simple auction using the English auction style.

8.2.1 Roles

Generally, our simple auction consists of three different roles.

- **Consumer**

The Consumer wants to buy a virtual machine. The goal of the consumer is to get the required virtual machine at a low price.

- **Datacenter**

  Datacenters have hosts and want to sell virtual machines to consumers at a high price. In the following sections, they organize an auction in order to sell their virtual machines.

- **Matchmaker**

  The matchmakers are responsible for bringing consumers and providers together. Thus, datacenters which want to sell virtual machines have to register their virtual machines at the matchmaker. Consumers, who want to buy virtual machines, have to ask the matchmaker for datacenters.

8.2.2 Auction Procedure

Our simple auction model is similar to a typical English auction. In an English auction, every attendant can submit a bit which increases the previous bit. Such situation is illustrated in figure 8.4. Here three bidders want to buy one resource. *Bidder A* does not want to spend more than 59$, the other two bidders do not want to spend more than 60$.

Assume that *Bidder A* has submitted the last bid of 59$. We call this bid the leading bid. If the other two bidders want to buy the resource, they have to bid 60$, otherwise *Bidder A* wins. Both bidders want to increase their bid. However, in real-time auctions, only the first bidder's bid is accepted. In our scenario, *Bidder B* was the first one and has a leading bid of 60$. The second bidder, *Bidder C*, does not want to spend more than 60$ so that *Bidder B* wins the auction.

Our simulation is executed in phases, so there is no real time difference between two bidders. In each bidding phase, we collect all bids of all attending consumers. This is shown in the lower part of figure 8.4. Here *Bidder A* has the leading bid, the other two bidders submit 60$ as a bid. From the increasing bids of the same value, we choose randomly one. So both, *Bidder B* and *Bidder C* have the chance of getting the leading bid.

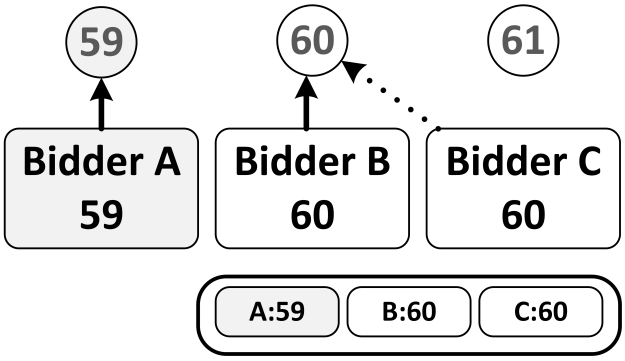


Figure 8.4: *English auction and simple auction*

Figure 8.5 shows a more detailed description of our implemented auction in CloudSim. In this example, two consumers, represented by their brokers, a matchmaker and a datacenter are involved. The grey boxes represent the events which are executed in the same phase as described in section 8.1.2. The message names are the event tags.

First of all, the datacenter registers its virtual machine it wants to sell. The consumers ask the matchmaker for datacenters which offer a virtual machine with their required

characteristics. Both brokers require the same virtual machine which is offered by the datacenter. Thus, the matchmaker returns to both brokers the address of the datacenter.

Afterwards, both brokers submit bids to the datacenter. These bids have to comply with a datacenter's minimum bid. The datacenter evaluates the bids and returns a bid exceeded message to the broker who does not provide the best bid. The broker whose bid is the best one gets a leading bid message. If two bids are equal, the leading bid is assigned randomly to a broker. If a broker receives a bid exceeded notification indicating that his bid has been exceeded, he has the option to exceed his bid or to leave the auction. The excess can be governed by an increment. If only one broker is left, the auction is finished and the broker left creates the virtual machine (not shown in the figure).

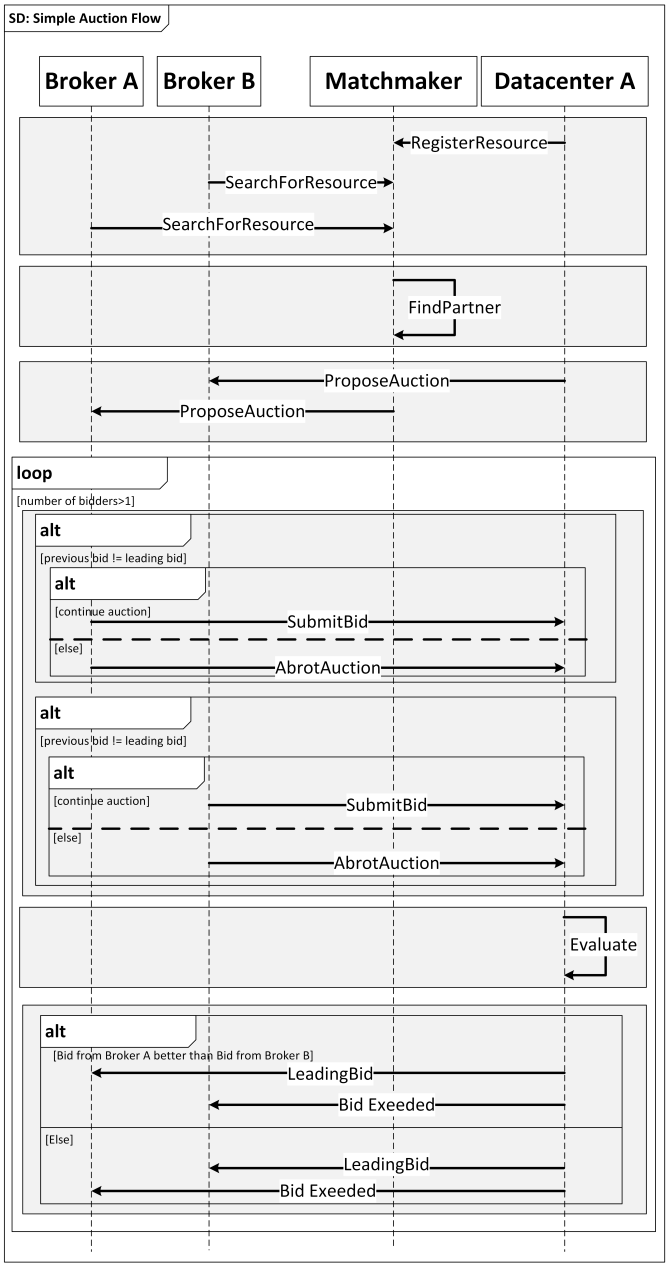


Figure 8.5: *Simple auction message exchange*

As the sequence diagram shows, the matchmaker is not contacted during auction itself. So the datacenters are responsible for the auction and can implement different evaluation

algorithms or message sequences.

The bids submitted within the auction are mandatory.

In the following sections we describe an auction scenario.

8.3 A Simple Auction with Homogeneous Virtual Machines

In this section we want to describe a scenario in which datacenters offer virtual machines to consumers. All the virtual machines have exact the same characteristics so that we talk about homogeneous virtual machines. The goal of a consumer is to get exactly one virtual machine as cheap as possible.

For each virtual machine, the datacenter starts a separate auction where any consumer can attend. Of course, consumers need several parameters governing their behaviour in an auction.

- **Maximum Price**

  Consumers need to have a maximum price which they want to pay for a virtual machine. A price exceeding this limit would result in a negative profit for the consumer. So a consumer prefers not to have a virtual machine before he pays more than the maximum price. We call this *Maximum Price Policy*.

- **Selection Strategy**

  The *Maximum Price Policy* forces the termination of all auctions which have already exceeded a consumer's price limit.

  However, consumers need a further strategy regarding decision making about continuing and terminating auctions. These strategies will be explained in the following subsection.

- **Bid Increasing Strategy**

  This strategy determines the extend to which the consumer increases his bids. In our simulation, all auctions have a fixed increment so that no bid increasing strategy can be used.

In the following subsection we will introduce selection strategies.

8.3.1 Selection Strategies

We have already introduced the *Maximum Price Policy* which all our consumers use. Selection strategies describe which auctions are quitted and which ones are continued.

From an isolated point of view, such strategies are not necessary: All auctions have to result in the same price, as all providers offer the same virtual machine. If the consumers are rational, they bid the same price in all auctions . No one is willing to pay more for the same virtual machine.

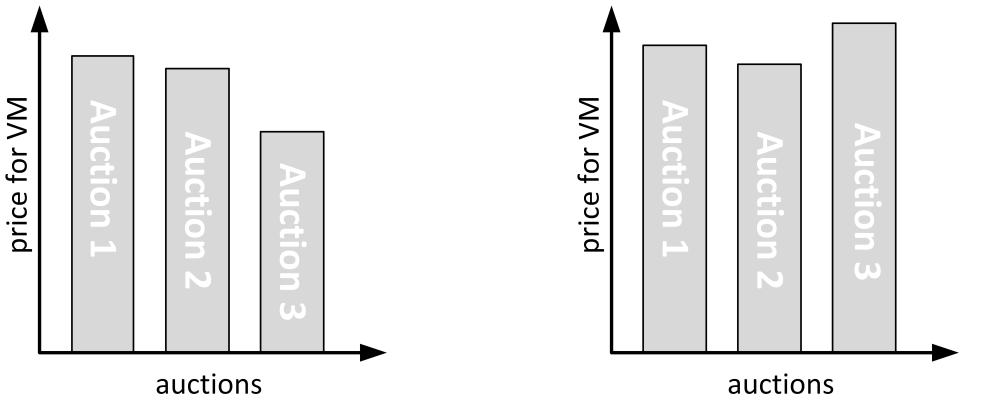
However, if less virtual machines are available than required, consumers have to fear that they may not get a single virtual machine. Moreover, consumers may have different price limits which contort the uniformity of prices.

In this section, we want to introduce two selection strategies consumers may use.

- **Secure Strategy**

  To clarify this strategy, consider the following situation as depicted in figure 8.6a. Three auctions are running whereby the price of the first auction is higher than the price of the second and third one. In such a situation, the consumer may decide to submit a bid to all three auctions. Of course, the consumer will not bid for auctions which have already exceeded the consumer's price limit. This strategy is called the

Secure Strategy. Using this strategy, the probability that the consumer does not win any auction is minimized. However, submitting bids to several auctions may result in unexpected outcomes as the consumer may win more than one virtual machine and consequently has to pay more. From the pricing point of view, this strategy increases



(a) *Auction prices - start situation* **(b)** *Auction prices - possible end situation*

Figure 8.6: *Auction prices scenarios*

the probability that a consumer wins expensive auctions. Again, consider the situation in figure 8.6a. Assume that this situation is the start situation and that final situation is analogous (which means the price ordering of virtual machines remains the same). Assume further, by bidding for all auctions, the consumer has won *Auction 1* as no one else wanted to pay so much for a virtual machine. Winning *Auction 1*, the bidder terminates all other auctions. As you can see, *Auction 1* is the most expensive one. So the result is suboptimal. The problem is that the consumer does not know anything about other consumers. The alternative to bidding for *Auction 1* is terminating *Auction 1* and continuing the other two auctions. However, this alternative may be suboptimal, too: If there is a hard competition for e.g. *Auction 3* as shown in figure 8.6b, *Auction 1* is not the most expensive one. In situations such as shown in figure 8.6b terminating *Auction 1* and continuing *Auction 3* would be a bad decision.

If more than one consumer use the secure strategy, a lower price limit is created if all auctions have the same minimum bid and a fixed increment. Consider two consumers using this strategy as shown in figure 8.7. Each of the consumers will increase all bids of all auctions until one of the consumers reaches his price limit. Therefore no auction will stop until at least two consumers are attending using this strategy. The minimum price limit will be the second highest maximum price of a consumer using the secure strategy. A consumer who has less money than this boundary has no chance to win a virtual machine leading to a bad allocation of virtual machines. In figure 8.7 the lower price limit is the price limit of consumer A.

If auctions have different minimum bids and a fixed increment, this effect does not occur. If auctions start with different minimum bids, consumers with a high maximum price may win an expensive virtual machine and so terminate all other auctions. Such a situation will be explained in more detail in a following section.

If consumers use the secure strategy and more virtual machines are offered than needed, the probability that a consumer gets more virtual machines is very high.

- **Risky Strategy**

  If a situation as described in figure 8.6a occurs, a consumer could decide not to bid

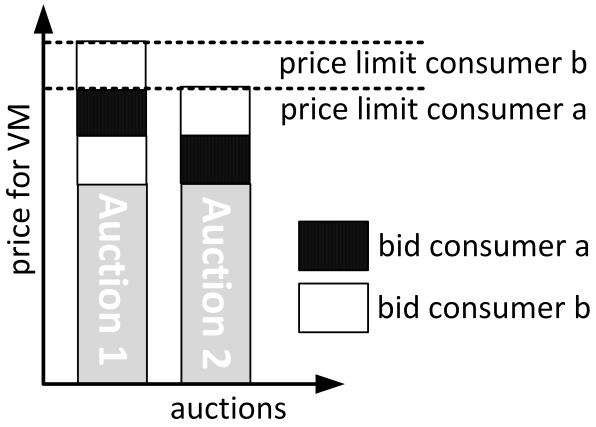


Figure 8.7: *Consumers using secure strategy*

for *Auction 1* any more. If the consumer terminates *Auction 1*, he can prevent the situation which occurs using the secure strategy: By bidding and winning the most expensive auction (*Auction 1*) the consumer may miss an auction which results in a lower price.

However, as the situation in figure 8.6b shows, bidding for the cheapest auction does not necessarily result in a cheaper price. Additionally, this strategy reduces the probability that the consumer gets any virtual machine.

The auction mechanism has several disadvantages. First of all, the consumer's demand will differ from the offers made by providers. Moreover, the auction's assignment strategy is bad: Even if enough virtual machines are available, there will be consumers who do not get a single machine or more than one machine. To proof this claim, we executed a simulation.

8.3.2 Simulation Scenario Using Secure Strategy

In this scenario we have 10 consumers who all require a virtual machine with the characteristics shown in table 8.1. The 17500 MIPS are assumed per processor. This virtual machine should represent a typical end user machine.

Additionally, we create datacenters which offer 10 of theses virtual machines. All offered as well as required virtual machines have the same characteristics.

Table 8.1: *Homogeneous virtual machine*

Ram	Storage	Processors	Processing
4096MB	512000MB	2	17500 MIPS

In our scenario we use 5 datacenters whereby each datacenter has two hosts. Each host has enough processing capability to run one virtual machine described in table 8.1. This data is summarized in figure 8.2.

Table 8.2: *Scenario setup*

Datacenters	Hosts	Demanded VMs	Offered VMs
5	10	10	10

The setup of consumers and datacenters is the following:

- The datacenters use the auction described above as allocation mechanism.
- The consumers use the secure strategy as selection strategy.
- The bidder's price limit is a random number between $50 - 99$.
- The increment for all auctions is 0.5. This means, if a consumer wants to increase a bid, he has to submit a bid exceeding the previous bid with 0.5.
- The auction's minimum bid is a random number between 0 and 49.
- A consumer does not know anything about other consumers.

Table 8.3: *Assignment Table*

Auction	Winner	Price
Auction 0	**Bidder B**	67
Auction 1	**Bidder D**	92.5
Auction 2	**Bidder B**	66
Auction 3	**Bidder E**	89
Auction 4	**Bidder C**	82.5
Auction 5	**Bidder H**	92
Auction 6	**Bidder F**	70.5
Auction 7	**Bidder A**	73
Auction 8	**Bidder G**	90.5
Auction 9	**Bidder B**	66
	Average Price	**78.9**

We executed the simulation with these parameters a 1000 times. One result is shown in table 8.3. Here three virtual machines were assigned to a single bidder (bidder B). All the other virtual machines were assigned to other bidders. The result looks interesting, as bidder B got the three cheapest virtual machines.

The prices from table 8.3 are visualized in figure 8.8. The average price is 78.9.

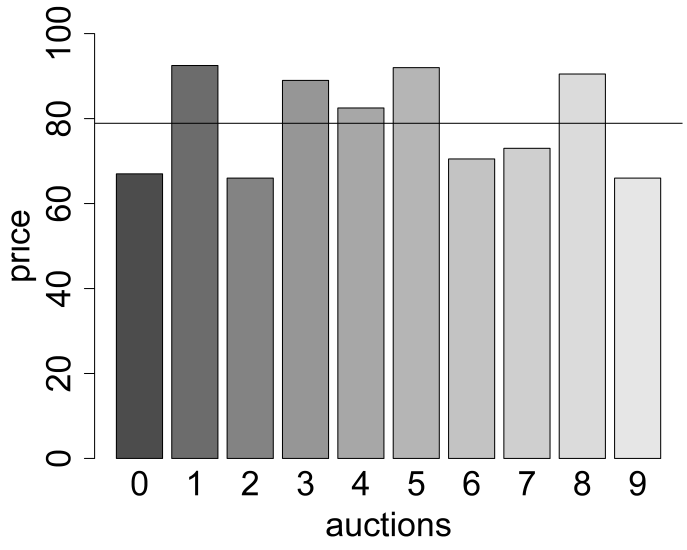


Figure 8.8: *Prices of 10 auctions*

In figure 8.9 we see how bidder B has won the three instances. This figure illustrates a situation at the end of the simulation process. As you can see, only four auctions are open. In three out of the four auctions, bidder B is the bid leader. Of course, a situation in which three submitted bids are chosen as leading bids is unlikely to face, however, it can occur. In *Auction*6 bidder F is the last remaining bidder. Thus, he wins this auction. As

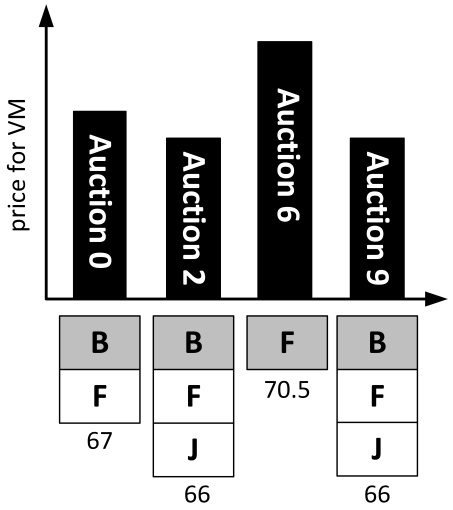


Figure 8.9: *Instance Prices of 10 auctions*

bidder F has not any other leading bids, he can terminate all the other auctions. Bidder J has already reached his limit so he does not submit any further bid. As bidder F and J terminate the auctions, bidder B is left and gets the last three virtual machines. Bidder B was lucky, because these last machines are pretty cheap. However, bidder B did not want to have so many machines.

In all our simulations, the cheapest virtual machine is the last one which is assigned to consumers. Without knowing anything about other consumers you never know if you get a single virtual machine. Moreover, it can occur, that you get more than one virtual machine. Executing this scenario with 10 datacenters and 10 brokers a 1000 times results in an average price of 74.61. All our results are plotted in figure 8.10. The abscissa shows the number of different consumers who won an auction. 10 means that all bidders won an instance, whereas 3 means that the 10 instances were assigned to only 3 consumers. The ordinate shows the price. We have added a regression trend line to the plot. It indicates that constellations, in which instances can be assigned to a lot of different consumers, lower the price.

8.4 Bazaar Style Negotiation

We implemented the basis of the bazaar style negotiation in CloudSim. As already mentioned in a previous section, WS-Agreement already proposes a protocol for bilateral negotiation which can be used for bazaar style negotiation. The negotiation messages in WS-Agreement are XML-based. Of course, in the event based CloudSim framework we do not exchange XML-messages. Nevertheless, we represent simplified XML-messages in CloudSim as events. Additionally, we used the basic ideas of the WS-Agreement Negotiation protocol. Several mechanisms which we used are stated below.

- Template offering
- Offer- Counter-offer sequence

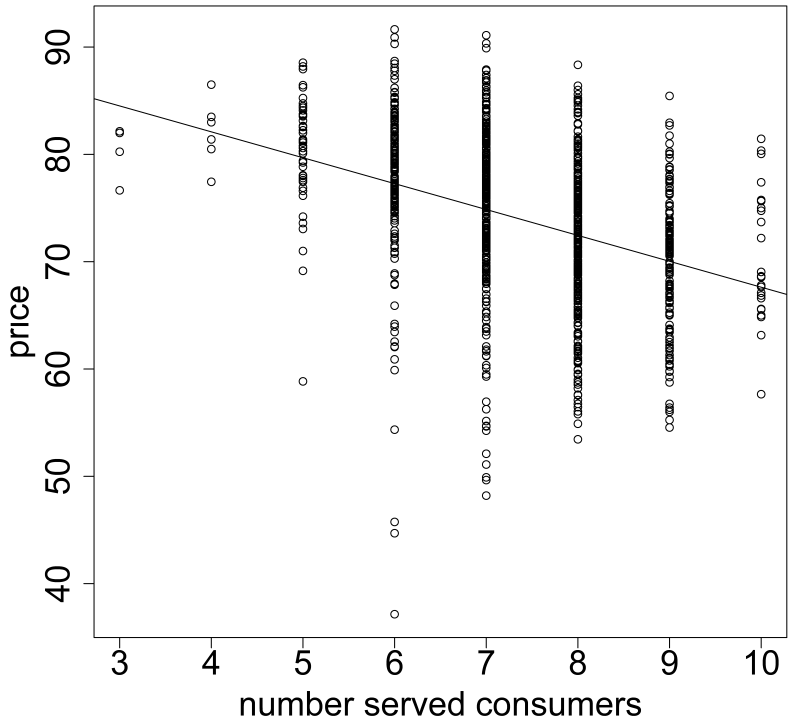


Figure 8.10: *Simulation results with regression trend line*

- Value constraints in offers

Figure 8.11 shows a simplified sequence diagram of a bazaar negotiation. As you can see, three roles are involved in negotiation. The repository acts as a matchmaker like in the auction scenario. Moreover, it has to provide an advertise mechanism as described within the negotiation framework description. The consumers are represented by brokers. At the start, consumers ask the repository for matching providers. The providers returned by the repository are contacted who send their templates back. As soon as the templates have been received, the interactions for both, consumer and provider, are identical: Each offer, which can be the template too, is acknowledged with an accept, reject or counter-offer message until the two negotiation parties agree to an SLA. It is also possible, that no matching agreement is found.

8.5 Initial Implementation in CloudSim

In this chapter we want to present an initial implementation of a negotiation in CloudSim. The used negotiation style is based on the offer and counter-offer mechanism described in the WS-Agreement Negotiation protocol [15].

In a typical negotiation scenario, two negotiation partners such as provider and consumer exchange messages in a rotative way. A simple example for the mechanism is provided in figure 8.12 which is a simplified illustration of a figure published in [15]. In the illustration, a provider starts a negotiation by sending an offer to the consumer ($n = 0$). Afterwards the consumer responds to the offer with two counter-offers ($n = 1$). The provider responds to the counter-offers with two new counter-offers ($n = 2$).

For clarity reasons we use the term *proposed offer* for offers which are received by a consumer or a provider. The offers which are sent are named *counter-offers*. So in $n = 1$, the consumer receives a proposed offer and creates two new counter-offers. In $n = 2$ the provider receives two proposed offers and creates two counter-offers.

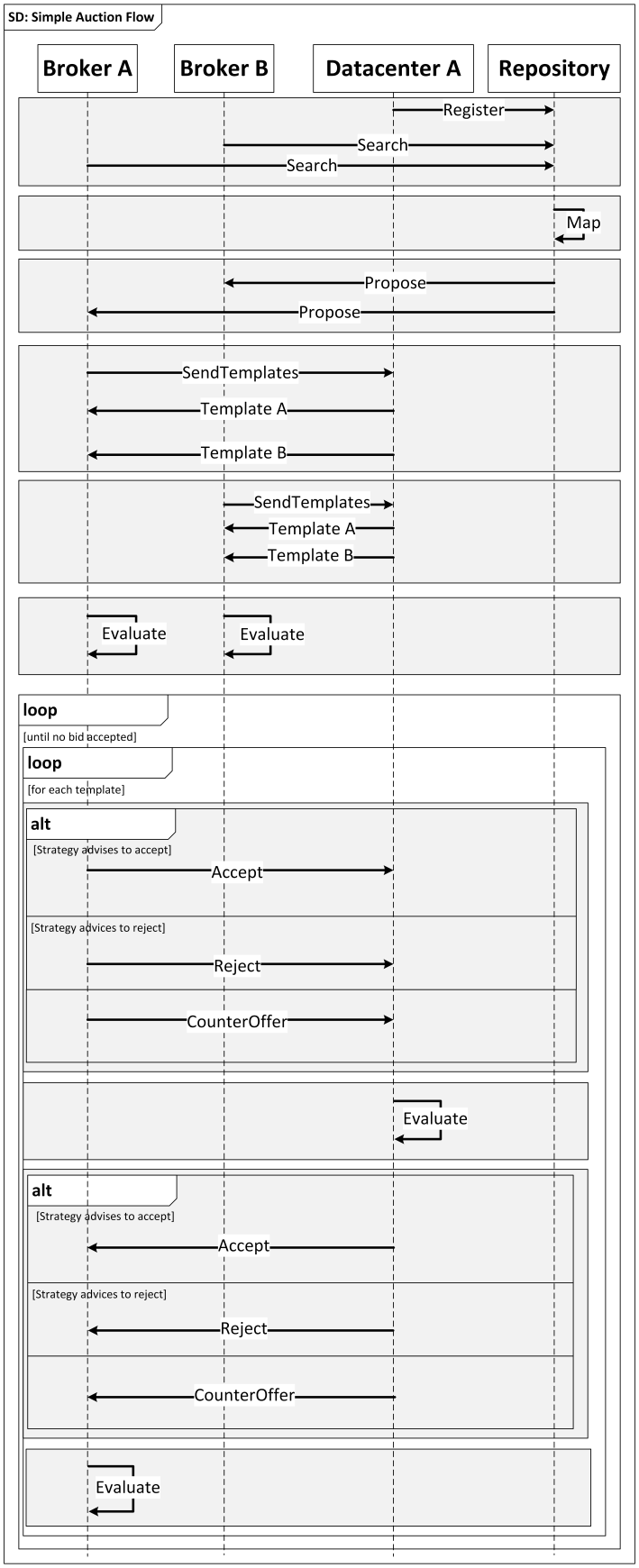


Figure 8.11: *Bazaar style negotiation in Cloud Sim(overview)*

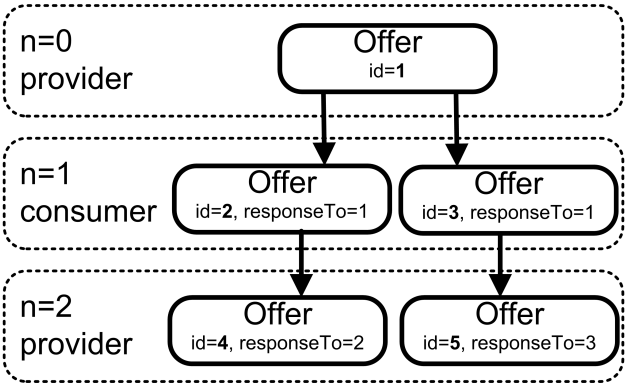


Figure 8.12: *Offer- Counter-offer example*

Of course, we have not implemented the all WS-Agreement protocols. This is because WS-Agreement protocols standardize negotiation over the Internet. For example, it standardizes XML-messages which are exchanged between two remote negotiation partners. However, CloudSim is a Java based simulation environment where events can be considered as substitutes for XML-messages. Each event is represented by a Java object.

If a negotiation partner receives a proposed offer, he has to make several decisions in order to respond to the offer. Precondition for making any decision is to evaluate the offer in terms of utility. The utility evaluation is described in the following section.

8.5.1 Utility Evaluation

A typical method for evaluating the utility of an item such as an offer for virtual machines is to use utility functions. In this paper, utility functions represent the utility of an offer for a provider or a consumer. Therefore, offers with higher utility values are favoured by consumers and providers over offers with lower utility values. Providers as well as consumers will have different utility functions based on their needs, experiences and resources.

In this thesis we want to give a short introduction about how utility functions can be used to evaluate a virtual machine. We distinguish between consumer and provider utility functions.

8.5.1.1 Provider Utility

The main goal of a provider is to maximize profit. This goal can be reached by selling VMs at high prices.

In our implementation we used the gross margin as utility function for the provider. The gross margin is the difference between turnover and variable cost. So offers, which result in a higher gross margin are preferred over offers with a lower gross margin. The following equitation shows the used provider function:

$$U(Provider) = \begin{cases} Price - Ram \cdot Price_{Ram} \\ -Storage \cdot Price_{Storage} \\ -Processing \cdot Price_{Processing} \\ \\ -\infty, Storage > Max_{Storage}, \\ Ram > Max_{Ram}, \\ Processing > Max_{Processing} \end{cases} \tag{8.1}$$

The variables $Price_{Ram}$, $Price_{Storage}$, $Price_{Processing}$ represent variable cost. So a proposed offer for VM has high utility for the provider if either the price is high or few resources are required. The provider has limited resources which are represented by $Max_{Storage}$, Max_{Ram}, $Max_{Processing}$. Offers violating these ranges are valued with a $-\infty$ utility indicating that it cannot be supplied.

This utility function is a simplification which we implemented in our simulation. In our further research, we want to enrich this utility functions with further concepts of the previous described cloud cost model. Furthermore, the utility can be extended by limiting the maximum amount of resources delivered to one consumer in order to enforce diversity.

8.5.1.2 Consumer Utility

As already explained in previous chapters, the consumers usually do not require a VM with fixed values. Instead, they have ranges or minimum values for the characteristics of the required VM.

In order to calculate the consumer utility, we consider a utility function for each parameter: storage, Ram, processing and price. In our implementation, the consumer uses a sqrt-function for storage and a log-function Ram and processing as shown in figure 8.13. These functions consider saturation: the utility increase decreases with each further MB for storage and Ram or MIPS for processing power.

This saturation reflects real consumer behaviour. A consumer requiring 8 GB Ram will get a great utility increment if it gets a further GB Ram. If 100 GB Ram are offered to the same consumer then the utility increment of a further GB RAM is small.

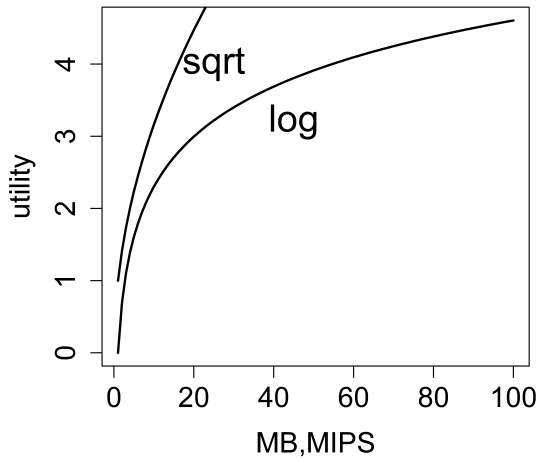


Figure 8.13: *Log and sqrt utility function*

The consumer has minimum requirements of VM characteristics which are considered in the utility function whereby x is a placeholder for storage, Ram and processing power.

$$U(Consumer_x) = \begin{cases} log(x), sqrt(x) & x \geq Min_x \\ -\infty, & x < Min_x \end{cases} \tag{8.2}$$

The utility for the price is calculated by a linear function as shown in the following

equitation.

$$U(Consumer_x) = \begin{cases} Max_{price} - price & x \leq Max_{price} \\ -\infty, & x > Max_{price} \end{cases} \tag{8.3}$$

As you can see in utility function 8.2, the utility value increases with increasing resources whereby the utility value of the price function in equation 8.3 decreases with an increasing price.

Using the equations, we can calculate the utility of price, storage, Ram and processing power. In order to evaluate a virtual machine, we need a final utility value. We summarize the utilities of storage, Ram, processing power and price to a total utility by standardizing them and calculating the weighted sum. The weight indicates the importance of the VM characteristic on the total utility.

After the utility of an offer is determined, a negotiation partner is able to decide how to respond to the offer. Typically, a response results in an accept message, a reject message or a counteroffer.

In the following section, we describe the way of how the counteroffer is created.

8.5.2 Counteroffer Creation

If a proposed offer is received which is neither rejected nor accepted, a counteroffer has to be created. Typical influence factors for counteroffer generation are depicted in figure 8.14.

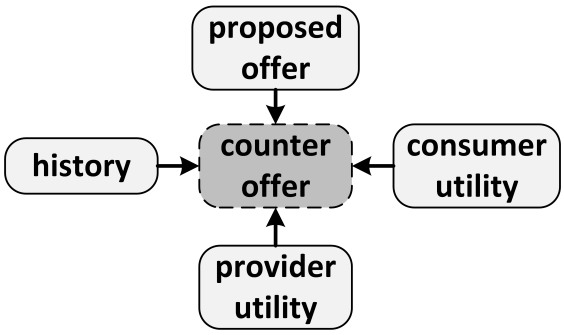


Figure 8.14: *Counteroffer generation influencing factors*

- **Proposed Offer**

  A counteroffer should consider the proposed offer. The counteroffer should be similar to the proposed offer. This is, because the proposed offer is usually an offer which is acceptable for the negotiation partner.

  Ignoring proposed offers leads to an interchange of random offers without perusing promising offer sequences.

- **Consumer Utility/Provider Utility**

  As already described, utilities are essential for evaluating offers. Counteroffers should have utility for the sending party as well as for the receiving party. So the counteroffer generator has to generate offers which increase utilities of both negotiation parties. Usually, this is a difficult task. However, due to different valuations of virtual machine characteristics, an improvement for both, consumer and provider, is possible as we will explain in the next subsection.

  In order to assess the utility function of the negotiation partners, models can be created.

- **History**

  The history can be used to find out which counteroffers may be successful and which utility values are reachable.

In our initial implementation we consider the proposed offer as well as the utility of the sending negotiation partner. The modelling of the other negotiation partners as well as the history database are parts of our further research.

8.5.3 Counteroffer Generation

A counteroffer should improve the utility of both, consumer and provider. Usually, a greedy counteroffer generation strategy will not be successful. An example of such a greedy counteroffer generation strategy is given in the following paragraph:

A consumer receives an offer for a VM. The proposed offer is modified by reducing the price and sent back as counteroffer. It is obvious, that the counteroffer is of lower utility for provider than the proposed offer. Counteroffers which are created by simply reducing price or increasing resource values for Ram, storage or processing power will not be accepted by the provider.

The goal of a counteroffer is to increase the utility of both, consumer and provider. As we have described in a previous section, changing exactly one VM characteristic of a proposed offer will lead to an utility increase of one negotiation partner while the other negotiation partner will lose utility. Thus, no so called win-win situations will occur.

We created two rules for the counteroffer generation:

R1: Modify exactly two VM characteristics of the proposed offer for creating counteroffers.

R2: One VM characteristic has to be improved, whereas the other characteristic has to be degraded.

By applying these two rules for counteroffer generation you can create offers which may be of higher utility for both negotiation partners. However, there is no guarantee for such a win-win situation which we will explain in the following subsection.

R1 is used in our simulation to ensure, that the counteroffer remains similar to the proposed offer. This rule is a simplification for the counteroffer generation.

R2 is the basis for finding negotiations results with improvements for both, consumer and provider. The reason for such a win-win situation is the different valuation of VM characteristics. To clarify the valuation differences, we provide the following example.

8.5.4 Example

We describe the VM characteristics using a vector like $(300GB, 4000\text{MIPS}, 5GB, 10\$)$. The first element represents the storage, the second element the processing power, the third element the Ram and the last item the price. In the following, we will neglect the units in the vector.

We consider that the offer $(300, 4000, 5, 10)$ was received by a consumer. The consumer wishes to have more Ram. So he creates a counteroffer based on the proposed offer: $(300, 4000, 7, 11)$. As you can see, the consumer increased the Ram but also the price. Nevertheless, the utility of the new offer is higher for the consumer: the consumer is willing to pay more than 1$ for 1GB Ram. However, the consumer offers the provider 1$ for 2GB Ram in the counteroffer.

The provider, who evaluates the offer, may have lower costs than 1$ for the two additional GB Ram. So the new offer is better than the old one for both negotiation parties. This substitution mechanism works for all VM characteristics. Thus, instead of increasing the

price, the consumer could try to decrease the processing power in order to get more Ram. It is also possible to decrease, for example Ram and storage, in order to decrease price. The process of improving VM characteristics and degrading another VM characteristic is called substitution process within this paper.

The substitution process which we implemented in our initial implementation is described in figure 8.15. The numbers on the abscissa represent the process steps. In the first step the utility of the proposed offer is calculated. Additionally, the VM characteristic which should be improved and the VM characteristic which should be degraded are selected. So we simply improve one parameter by deteriorating another parameter.

In the example depicted in figure 8.15 processing power is the VM characteristic which we want to improve. Storage will be the VM characteristic which we want to degrade. In the second step processing power is increased. The processing power is set to a value so that the utility of the counteroffer reaches the goal utility + Δ. The Δ represents an offset which will be used for degrading a VM characteristic in order to reach the goal utility. In the final step, we reduce the storage until the Δ is compensated. In our example, we set storage to a value until the goal utility is reached. The so generated counteroffer has higher utility for the counteroffer generator than the proposed counteroffer. Additionally, it may have higher utility for the negotiation partner.

There is a large number of offer modification mechanisms to reach the goal utility. The mechanism described above, represents a simple one in order to illustrate the principle.

We suggest to keep the Δ low. This is because we want to keep the counteroffer similar to the proposed offer. A big Δ leads to big improvement of processing power while storage is greatly reduced.

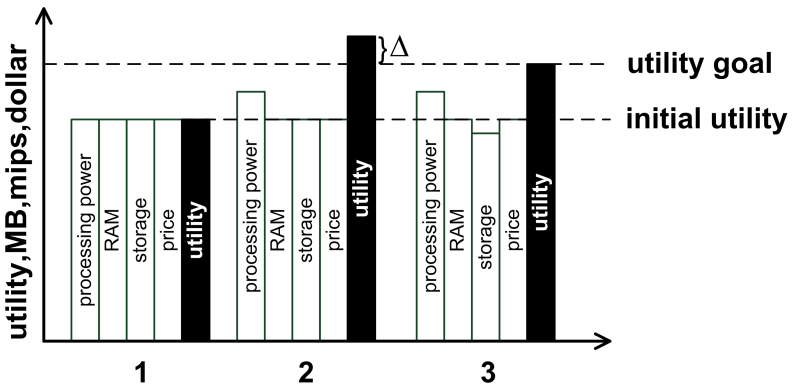


Figure 8.15: *Substitution process*

Usually, a negotiation partner described does not know anything about the utilities of the other negotiation partner. This makes counteroffer generation complex as you do not know which VM characteristics should be improved and which ones should be degraded. Moreover, you do not know the increment size for improving and degrading. One strategy is to create counteroffers for all combinations as shown in figure 8.16. This figure shows an initial offer and its counteroffers. The initial offer may be sent by a provider. The consumer uses this offer and creates counteroffers by improving as well as degrading all VM characteristics resulting into 12 different offers all of which have the same utility for the consumer.

In the first part of the figure, the processing power is reduced, in the second part, Ram is reduced, in the third part, storage is reduced and in the fourth part, price is increased.

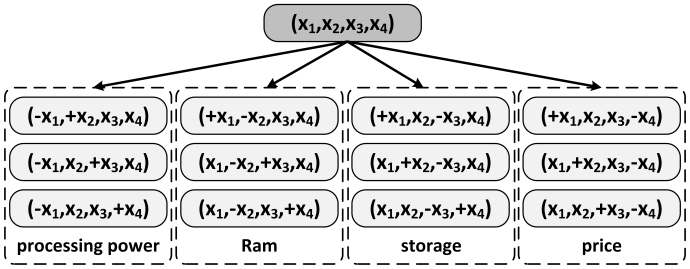


Figure 8.16: *Counteroffer tree*

8.5.5 Sample Scenario

In this section, we want to introduce a scenario using the implementation we described above. The results of the scenario should show, that negotiation results in higher utilities for both, consumer and provider than the supermarket approach. Additionally, the negotiation result is compared to Pareto optimal solutions.

A Pareto optimum solution is a situation in which neither the provider nor the consumer can get a higher utility without reducing the utility of the other negotiation partner.

For the scenario we used one provider and one consumer. Both use the corresponding utility functions described above. The consumer's minimum virtual machine characteristics are (200,30000,3,100). So if one of these ranges is violated, the VM has no utility for the consumer. Of course, the prices forms a maximum boundary.

The negotiation is started by the provider who sends one offer. This initial offer is called template. The provider rejects each counteroffer which has lower utility than the utility of the template.

In our initial scenario, the negotiation is finished as soon as one of the two negotiation partners accepts an offer. An offer is accepted by the consumer if it has a utility greater than 7500. The provider accepts an offer if its utility is greater equal than 13000. These boundaries don't mean that an offer has to have at least 7500 consumer utility and 13000 provider utility. Instead, if one of these utility limits is reached, the provider/consumer tries to make an agreement on that counteroffer. Nevertheless they are also willing to accept offer of lower utilities, if they get an acceptance request. The minimum agreement acceptant utilities are separated parameters and are turned off for the scenario. Consider that the provider and consumer use different utility functions. So their utility values are not comparable.

Generally, for each offer 12 counteroffers are created. So after 5 iterations, 20736 exist. So we are not able to depict a complete negotiation tree within this paper.

A pruned negotiation tree is illustrated in figure 8.17. It shows the path from the initial offer to the accepted offer. Additionally, the tree visualizes the negotiation generation mechanism: the template is used and the counteroffer modify exactly two parameters of the counteroffer. For the first counteroffer in $n = 1$ the template's price as well as processing power was modified. The second counteroffer is equal to the template without the processing power and the storage capacity.

The initial offer with the characteristics $(300GB, 4000\text{MIPS}, 5GB, 10\$)$ has a consumer utility of 6299 and a provider utility of 3000. The accepted offer $(470GB, 4800\text{MIPS}, 5.25GB, 13\$)$ has a consumer utility of 7531 and a provider utility of 4425. So the consumer get 1232 more utility and the provider gets 1425 more utility.

In a supermarket approach negotiation and consequently improvement does not exit. Thus, if the provider offers the template in a supermarket based allocation process, the consumer can take it or leave it.

The results of our scenario are summarized in table 8.4.

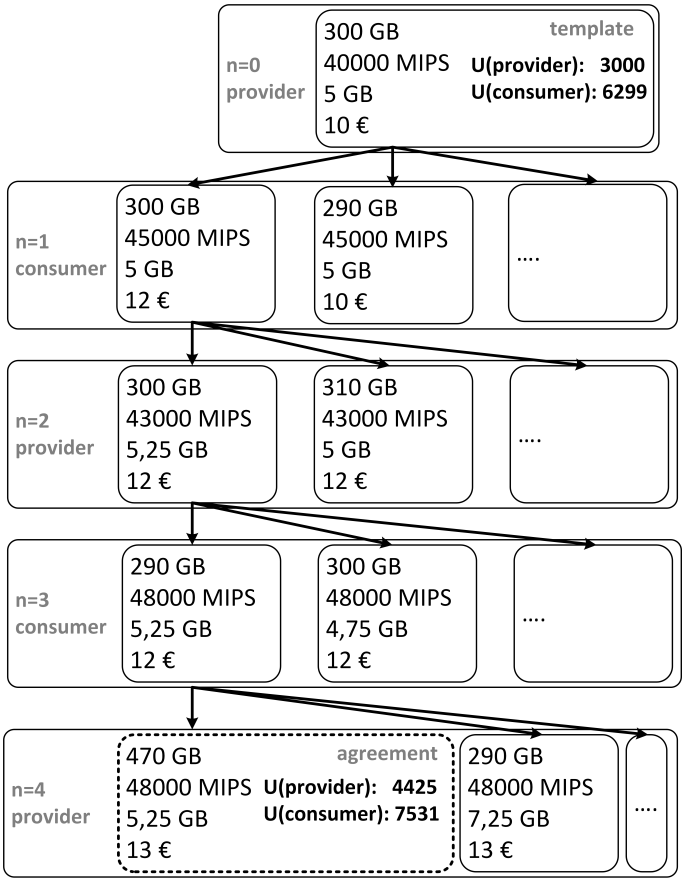


Figure 8.17: *Offer tree from template to agreement*

Table 8.4: *Result*

	Supermarket	**Bazaar**	**Pareto**
U(consumer)	3000	4425	≈ 30125
U(provider)	6299	7531	≈ 8749

The Pareto border representing the Pareto optimum is illustrated in figure 8.18 by the black line. In this figure, the initial offer as well as the agreement are depicted. We calculated the Pareto optimum by an iterative approach: we iterated overall possible service combinations and looked for Pareto optimal points. For this iteration we used increments for all service characteristics: Ram: 256 MB, Storage: 10 GB, Processing Power: 5000 Mips and Price: 1. So our results are approximations.

Some datapoints of the Pareto border are shown in table 8.5. As you can see, neither the supermarket results, nor the negotiation results are Pareto optimal. However, the negotiation result is nearer to a Pareto optimal solution.

Figure 8.18 has an interesting curve. The first part of the curve seems to be approximately linear while the second part of the curve seems not to be linear and has a strong negative slope. We tried to explain this course of the curve by using the characteristic vectors of Pareto efficient points. Using the Pareto optimal vector, we recalculated consumer utility without considering the price. The resulting points are called *resource points*. Most of them have a new consumer utility (y-value). Of course, their x-value remains the same. The points are shown in figure 8.19. The points do not form a smooth line. As you can see, some resource points have the same consumer utility. The reason for the non-smooth

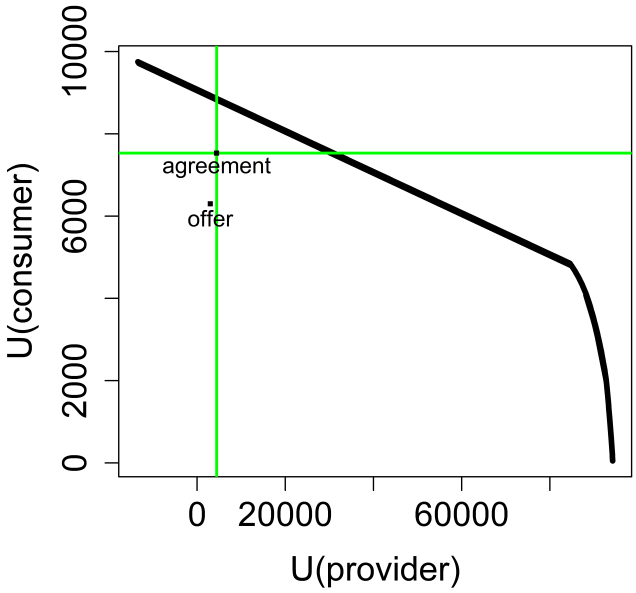


Figure 8.18: *Pareto border*

course is, that we used an iterative approach based on stepwise modification of the characteristics to get approximated Pareto efficient points as we have already described above. This increment size leads to non-smoothed courses of curves.

Using figure 8.19, we can conclude that that the offers which form the Pareto border remain stable for $U_{provider} <\approx 72000$. So the utility considering storage, Ram and processing power, without price is not decreased. However, the consumer utility considering price is decreasing as figure 8.18 shows. So it is obvious that for all points $U_{provider} <\approx 72000$, that the utility decrement is resulted by the price. The Pareto optimal data confirms that the VM characteristics processing power, Ram and storage remain approximately the same. As described above, the utility function for the price is linear which is the reason for the linear course of the first part of the curve.

At $U_{provider} \approx 72000$, the curve in the figure suddenly changes its course. In figure 8.18, until $U_{provider} \approx 72000$ the price has simply increased while all the other parameters remain almost unchanged. At $U_{provider} \approx 72000$ the price has been increased to the consumers limit. Thus, if we want to further increase the utility of the provider, we have to change the other VM characteristics. So Ram, storage and processing power are interchanged and reduced to get a provider utility of greater than ≈ 72000.

Table 8.5: *Selected Pareto optimal points (approximation)*

U(consumer)	U(provider)	VM characteristic
8824	3425	(995,58000,14.5,20)
8749	*4425*	(995,58000,14.5,21)
8724	5425	(995,58000,14.5,22)
7530	30775	(985,59000,15.75,48)
7531	30125	(990,59000,15,47)
7532	30050	(980,59000,15.25,47)

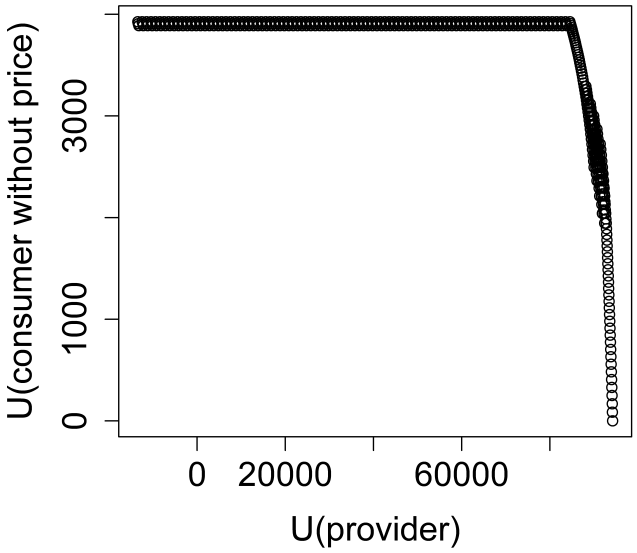


Figure 8.19: *Consumer utility without price*

8.5.6 Interpretation of Results

We have shown in the example before that the negotiation result is better than the negotiation template in terms of utility. This is a general effect of negotiation: the negotiation gives both, consumer as well as provider the chance to make improved offers.

If we are assuming that in a supermarket based market mechanism the offer is the equivalent to the template, the chance that this offer is Pareto efficient is small. This is because the provider doesn't know the consumers utility function. A negotiation based approach leads to results which may have a smaller or equal distance to the Pareto border than the supermarket offer.

A general comparison between negotiation, supermarket and auction is difficult. First of all, each consumer has different utility functions and each provider may have different datacenters as well as cost functions. Due to this individuality only simulation based approach can be used to make comparisons.

Especially comparing the negotiation based approach to auctions is a though task which we want to make in the future research. This is because auction strategies are individual and a lot of different types of auctions are existing.

8.5.7 Further research with CloudSim

In our current counteroffer generator we produce a lot of offers which may be redundant or inappropriate. In our further research we focus on the reduction of messages exchanged. Moreover, we want to utilize the history database as well as a model of the negotiation partner to further improve negotiation.

The modelization of the negotiation partner seems to be very challenging. We think that neural networks may be appropriate to assess the negotiations valuation of offers. This network gets the characteristics of the virtual machines as input and responds with a assessed utility value.

The presented simulation scenario can be extended too. We want to assess the behaviour

in a n:m negotiation. Further providers may buy resources from other providers in order to server additional consumers. These extension make the scenario more complex and more close to reality.

9 Further Research

In this thesis we described the way *how* autonomous service negotiation and re-negotiation can be done. We have introduced a generic framework and all relevant components. We have shown a simulation implementation and investigated typical market scenarios and business cases.

Our further research will focus on modelling a market by *using* the negotiation system as described in this thesis. Based on this market model we can execute scenarios and compare different strategies leading to optimal satisfaction of consumers and providers. We want to extend our simulation environment with more complex business rules. Furthermore, security aspects and their influence on business rules and service composition should be considered [106], [107].

Market Modelling Interactions on a market will be basically between providers and consumers. However, more complex constellations can occur: Consumers may form a group in order to reinforce their negotiation power. Providers can sell resources to other providers. Additionally, new actors such as intermediates may participate in market. All these forms have to be considered in order to model a realistic market.

Development of Strategies Depending on the market situation, strategies have to be developed in order to find optimal solutions. So there is for all participants involved a need for developing a strategy selection process as well as strategies which are suitable in specific market constellations . Extending service selection rules with utility functions will also be investigated.

10 Conclusion

In this thesis we introduced a novel negotiation framework. One part of this framework is the knowledge based Negotiation/Re-Negotiation engine. The knowledge base is built up on an economic cost model and a business rule repository as well as on a history database. Trading decisions require support by a system implementing the business strategy of the participating business partners by delivering all relevant business information, such as estimated fixed cost and variable cost. Such a cost model is the necessary requirement for automatic negotiation and re-negotiation of services. Our novel framework allows to implement the central business goals for both, service consumer and service providers. Using the concept of autonomic managers allows the framework to act as self governing system. To support all possible kinds of auction we designed a changeable auctioning plug-in as a part of the negotiation engine. For establishing SLA's a Service Template Registry is used where consumers retrieve templates and providers store their templates. For communication between the components we use standard protocols like WS-Agreement and WS-Agreement Negotiation.

In future research we will investigate the behaviour of different auctioning models. In further research we will focus on defining business strategies in the knowledge base of the autonomic system to allow for automatic, adaptive and dynamic negotiation and re-negotiation processes within an ICT marketplace.

Further, we presented a comprehensive analytical model forecasting the probability of finding matching providers for web service negotiations based on quality of service parameters. We simplified this problem by mapping it to the calculation of the probability that a given interval (*consumer interval*) and a randomly generated interval (*provider interval*) overlap. Using the simulation approach, we are able to forecast the probability of finding matching providers for any number of services and any *consumer interval* length. Moreover, we introduced a simple mechanism adapting the ranges in order to increase the probability of finding matching providers. Therefore, the consumer has to prioritize the service parameters. For justification we followed an approach checking our theoretical findings by simulation of practical examples.

This work is part of a research endeavour which aims at the development of a framework for automatic, adaptive and dynamic negotiation and re-negotiation processes establishing an ICT marketplace for web services.

List of Figures

List of Tables

Statements of Thesis

1. *"Autonomous systems are enabler for shifting CAPEX to OPEX."*
2. *"Bazaar-style negotiation supports liquidity of service market places."*
3. *"Business Rule driven autonomous negotiation systems help to materialize composite services."*
4. *"Cloud Cost Models distinguishing between variable and fixed cost leads to more flexibility and precision in service negotiation."*
5. *"Defining ranges for QoS parameters is the prerequisite for forecasting the probability of finding a matching provider."*
6. *"By Forecasting the probability of finding a matching provider, consumers could assess their chances of finding a provider offering the requested SLA."*
7. *"Bazaar-style service negotiation leads to higher SLA matching probability."*
8. *"You are what you do today, not what you say you will do tomorrow."*
9. *"Constant dripping wears the stone"*
10. *"There is no time like the present"*

CURRICULUM VITAE Werner Mach

General Informations

Date of birth	April 6, 1960 Amstetten (Lower Austria)
Citizenship	Austria
Status	Married to Sylvia Mach

Education

1966-1970 Primary School in Amstetten (Lower Austria)

1970-1974 Middle School in Steyr (Upper Austria)

1974-1979 School for higher technical education (HTBLA) Vienna (Austria)

1984-1989 Computer Science and Business Informatics at the University of Vienna

1989 Master degree Magister der Sozial- u. Wirtschaftswissenschaften (Mag.rer.soc.oec.) equivalent to Master of Business Information systems

2010 Doctorate degree Doktor technicae

since 2011 PhD study at the University of Vienna under the guidance of Univ. Prof. DI. Dr. Erich Schikuta

Languages

German native

English written and spoken

Professional Career

Studies in Computer Science and Business Informatics at the University of Vienna (UV), graduation with a Master degree while working as a software engineer.

Software development projects like database-applications, CAD and distributed database systems.

Head of the computer department in a world-wide acting electronics company.

Project leader of IT infrastructure projects in Austria and Germany. Many of these projects are trend-setting inside the company, like corporate environment -, networking- and information retrieval concepts.

Project manager of an electronic enforcement application.

Head of the development department for intelligent traffic systems and electronic design.

Head of project management and industrial engineering department.

Vienna, June 2014

Bibliography

[1] Peer Hasselmeyer, Henning Mersch, Bastian Koller, H. N. Quyen, Lutz Schubert, and Philipp Wieder. Implementing an SLA negotiation framework. In *Proceedings of the eChallenges Conference (e-2007)*, volume 4, pages 154–161, 2007.

[2] Heiko Ludwig. WS-agreement concepts and useagreement-based service-oriented architectures. *IBM Research Division, Technical Report*, 2006.

[3] W3C. OWL web ontology language overview.

[4] Mark Skilton and Capgemini Director. Building return on investment from cloud computing. *White Paper, The Open Group*, 2010.

[5] Philipp Wieder, Ramin Yahyapour, and Wolfgang Ziegler. *Grids and Service-Oriented Architectures for Service Level Agreements*. Springer Science & Business Media, August 2010.

[6] Irfan Ul Haq, Erich Schikuta, Ivona Brandic, Adrian Paschke, and Harold Boley. Sla validation of service value chains. In *Grid and Cloud Computing, International Conference on*, pages 308–313. IEEE, 2010.

[7] Alexander Keller and Heiko Ludwig. Defining and monitoring service-level agreements for dynamic e-business. In *Lisa*, volume 2, pages 189–204, 2002.

[8] John Wilkes. Utility functions, prices, and negotiation. *Market Oriented Grid and Utility Computing. Wiley Series on Parallel and Distributed Computing*, pages 67–88, 2008.

[9] C. Overton. On the theory and practice of internet SLAs. *Journal of Computer Resource Measurement*, 106(6):32–45, 2002.

[10] Antoine Pichot, Philipp Wieder, Oliver Wldrich, and Wolfgang Ziegler. Dynamic SLA-negotiation based on WS-agreement. *CoreGRIDTechnical Report, TR-0082*, 2007.

[11] Jun Yan, Ryszard Kowalczyk, Jian Lin, Mohan B. Chhetri, Suk Keong Goh, and Jianying Zhang. Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, 23(6):748–759, 2007.

[12] Kevin Kofler, Erich Schikuta, and others. A parallel branch and bound algorithm for workflow QoS optimization. In *Parallel Processing, 2009. ICPP'09. International Conference on*, pages 478–485. IEEE, 2009.

[13] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web services agreement specification (WS-agreement). In *Open Grid Forum*, volume 128, 2007.

[14] Sla Management Team. *SLA Management HandBOOK*. Enterprise Perspective, 2004.

[15] Oliver Waeldrich, Dominic Battr, Francis Brazier, Kassidy Clark, Michel Oey, Alexander Papaspyrou, Philipp Wieder, and Wolfgang Ziegler. Ws-agreement negotiation version 1.0. In *Open Grid Forum.[Accessed 20 May 2013]. Available from: http://www. gridforum. org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+ v1. 0. pdf*, 2011.

[16] Simon Parsons, Juan A. Rodriguez-Aguilar, and Mark Klein. Auctions and bidding: A guide for computer scientists. *ACM Computing Surveys (CSUR)*, 43(2):10, 2011.

[17] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The michigan internet AuctionBot: A configurable auction server for human and software agents. In *Proceedings of the second international conference on Autonomous agents*, pages 301–308. ACM, 1998.

[18] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and economic behavior*, 35(1):304–338, 2001.

[19] Y. Shoham. The zoology of auctions. *Lecture notes: Stanford University CS206Technical Foundations of Electronic Commerce*, 2001.

[20] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[21] Giuseppe Di Modica, Orazio Tomarchio, and Lorenzo Vita. A framework for the management of dynamic SLAs in composite service scenarios. In *Service-Oriented Computing-ICSOC 2007 Workshops*, pages 139–150. Springer, 2009.

[22] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[23] Farhana H. Zulkernine and Patrick Martin. An adaptive and intelligent SLA negotiation system for web services. *Services Computing, IEEE Transactions on*, 4(1):31–43, 2011.

[24] Srikumar Venugopal, Xingchen Chu, and Rajkumar Buyya. A negotiation mechanism for advance resource reservations using the alternate offers protocol. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 40–49. IEEE, 2008.

[25] Marco Comuzzi and Barbara Pernici. An architecture for flexible web service QoS negotiation. In *EDOC Enterprise Computing Conference, 2005 Ninth IEEE International*, pages 70–79. IEEE, 2005.

[26] Michael Parkin, Dean Kuo, and John Brooke. A framework & negotiation protocol for service contracts. In *Services Computing, 2006. SCC'06. IEEE International Conference on*, pages 253–256. IEEE, 2006.

[27] Ronald G. Ross. *Principles of the business rule approach.* Addison-Wesley Longman Publishing Co., Inc., 2003.

[28] Barbara Von Halle and Larry Goldberg. *Business Rule Revolution (ebook): Running Business the Right Way.* Happy About, 2006.

[29] Graham Witt. *Writing Effective Business Rules.* Elsevier, 2012.

[30] Gaihua Fu, Jianhua Shao, Suzanne M. Embury, and W. A. Gray. Representing constraint business rules extracted from legacy systems. In *Database and Expert Systems Applications*, pages 464–473. Springer, 2002.

[31] Mr Jrme Boyer and Hafedh Mili. *Agile business rule development*. Springer, 2011.

[32] Kapil Pant and Matjaz B. Juric. *Business Process Driven SOA Using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*. Packt Publishing Ltd, August 2008.

[33] W3C. RIF basic logic dialect (second edition).

[34] RuleML. RuleML wiki.

[35] W3C. SPIN - overview and motivation.

[36] Apache RDF. Apache jena - the core RDF API.

[37] Alec Sharp and Patrick McDermott. *Workflow Modeling: Tools for Process Improvement and Applications Development*. Artech House, 2009.

[38] Sujithra Sriganesh and Chandrashekar Ramanathan. Externalizing business rules from business processes for model based testing. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 312–318. IEEE, 2012.

[39] Artie Mahal. *How Work Gets Done: Business Process Management, Basics and Beyond*. Technics Publications, LLC, 2010.

[40] Paul Browne. *JBoss Drools business rules*. Packt Publishing Ltd, 2009.

[41] Ian Graham. *Business rules management and service oriented architecture: a pattern language*. John Wiley & Sons, 2007.

[42] SAP. SAP business rules management | SCN.

[43] Oracle. Oracle business rules overview.

[44] OpenRules. Product components.

[45] Drools. Drools - drools - business rules management system (java, open source).

[46] Rainer Alt and Hans-Dieter Zimmermann. Preface: Introduction to special section - business models. *Electronic Markets*, 11(1), 2001.

[47] Jaap Gordijn and Hans Akkermans. Ontology-based operators for e-business model de-and reconstruction. In *Proceedings of the 1st international conference on Knowledge capture*, pages 60–67. ACM, 2001.

[48] Alexander Osterwalder and Yves Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons, July 2010.

[49] Gnter Whe and Ulrich Dring. *Einfhrung in die Allgemeine Betriebswirtschaftslehre*. Vahlen, 2005.

[50] Karl Lechner, Anton Egger, and Reinbert Schauer. *Einfhrung in die allgemeine Betriebswirtschaftslehre*. Linde, 1994.

[51] Friedrich W. Selchert. *Einfhrung in die Betriebswirtschaftslehre: bersichtsdarstellungen*. Oldenbourg, 2002.

[52] Gnter Fandel and I. Produktion. *Produktions-und Kostentheorie*. Springer, 2010.

[53] Erich Schneider. *Einfhrung in die Wirtschaftstheorie: Band 4. Teil 1: Ausgewhlte Kapitel der Geschichte der Wirtschaftstheorie*. LIT Verlag Mnster, 1965 edition.

[54] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pages 1–11. IEEE, 2009.

[55] Derrick Kondo, Bahman Javadi, Paul Malecot, Franck Cappello, and David P. Anderson. Cost-benefit analysis of cloud computing versus desktop grids. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, 2009.

[56] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 446–452. IEEE, 2010.

[57] Armando Fox, Rean Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[58] Rodrigo N. Calheiros, Rajiv Ranjan, Csar AF De Rose, and Rajkumar Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009.

[59] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[60] Debabrata Dash, Verena Kantere, and Anastasia Ailamaki. An economic model for self-tuned cloud caching. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1687–1693. IEEE, 2009.

[61] Victor Chang, David A. Bacigalupo, Gary Wills, and David De Roure. A categorisation of cloud computing business models. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid 2010, 17-20 May 2010, Melbourne, Victoria, Australia*, pages 509–512. IEEE, 2010.

[62] Erik Theissen. Market structure, informational efficiency and liquidity: An experimental comparison of auction and dealer markets. *Journal of Financial Markets*, 3(4):333–363, November 2000.

[63] S. Sarah Zhang, Martin Wagener, Andreas Storkenmaier, and Christof Weinhardt. The quality of electronic markets. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10. IEEE, 2011.

[64] Ivan Breskovic, Ivona Brandic, and Jrn Altmann. Maximizing liquidity in cloud markets through standardization of computational resources. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 72–83. IEEE, 2013.

[65] Michael J. Barclay, Terrence Hendershott, and D. Timothy McCormick. Competition among trading venues: Information and trading on electronic communications networks. *The Journal of Finance*, 58(6):2637–2666, 2003.

[66] Kumar Venkataraman. Automated versus floor trading: An analysis of execution costs on the paris and new york exchanges. *The Journal of Finance*, 56(4):1445–1485, 2001.

[67] Larry Harris. *Trading and exchanges: Market microstructure for practitioners.* Oxford University Press, 2002.

[68] Jeffrey S. Rosenschein. *Rules of encounter: designing conventions for automated negotiation among computers.* MIT press, 1994.

[69] W3C. SPARQL query language for RDF.

[70] John Hebeler, Matthew Fisher, Ryan Blace, and Andrew Perez-Lopez. *Semantic web programming.* John Wiley & Sons, 2011.

[71] W3C. RDF 1.1 primer.

[72] MathPages. Probability of intersecting intervals.

[73] Thomas J. Archdeacon. *Correlation and regression analysis: a historian's guide.* Univ of Wisconsin Press, 1994.

[74] SPEC. SPEC power and performance committee.

[75] *Performance Evaluation: Metrics, Models and Benchmarks - SPEC International Performance Evaluation.*

[76] Alan O. Sykes. An introduction to regression analysis. 1993.

[77] A. Colin Cameron and Frank AG Windmeijer. An< i> r</i>-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics*, 77(2):329–342, 1997.

[78] Brown J. D. Questions and answers about language testing statistics: The coefficient of determination. 2003.

[79] Rand R. Wilcox. *Statistics for the social sciences.* Academic Press, 1996.

[80] SPEC. Server efficiency rating tool (SERT).

[81] Neil Rasmussen. Determining total cost of ownership for data center and network room infrastructure. *White Paper*, 6, 2011.

[82] Guoming Lai and Katia Sycara. A generic framework for automated multi-attribute negotiation. *Group Decision and Negotiation*, 18(2):169–187, 2009.

[83] Timothy Leung and William J. Knottenbelt. Global b2c and c2c online auction models. In *Proceedings of the 2011 Annual Conference on Innovations in Business and Management (CIBMP), London, UK*, 2011.

[84] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–ff, 1999.

[85] Carlos Molina-Jimenez, Santosh Shrivastava, and Massimo Strano. A model for checking contractual compliance of business interactions. *Services Computing, IEEE Transactions on*, 5(2):276–289, 2012.

[86] Christoph Redl, Ivan Breskovic, Ivona Brandic, and Schahram Dustdar. Automatic SLA matching and provider selection in grid and cloud computing markets. In *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, pages 85–94. IEEE Computer Society, 2012.

[87] Hind Saddiki and Hamid Harroud. An agent-based negotiation model for user-adaptive service provision. In *Multimedia Computing and Systems (ICMCS), 2012 International Conference on*, pages 973–978. IEEE, 2012.

[88] Ralph Vigne, Juergen Mangler, Erich Schikuta, and Stefanie Rinderle-Ma. A structured marketplace for arbitrary services. *Future Generation Computer Systems*, 28(1):48–57, 2012.

[89] N. Gregory Mankiw. *Principles of Microeconomics*. Thomson/South-Western, 2004.

[90] Werner Mach and Erich Schikuta. A consumer-provider cloud cost model considering variable cost. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 628–635. IEEE, 2011.

[91] W3C. Web service definition language (WSDL).

[92] Michael Parkin1 Peer Hasselmeyer2 Bastian Koller and Philipp Wieder. An SLA re-negotiation protocol.

[93] Andrs Micsik, Pter Pallinger, and Achim Klein. SOAP based message transport for the jade multiagent platform. In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Industry track*, pages 101–104, 2009.

[94] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software-Practice and Experience*, 31(2):103–128, 2001.

[95] V. Vishwanathan, J. McCalley, and V. Honavar. A multiagent system infrastructure and negotiation framework for electric power systems. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 1, pages 6–pp. IEEE, 2001.

[96] Ye Chen, Yun Peng, Tim Finin, Yannis Labrou, Scott Cost, Bill Chu, Rongming Sun, and R. Willhelm. A negotiation-based multi-agent system for supply chain management. *Working Notes of the agents*, 99, 1999.

[97] David GA Mobach, Benno J. Overeinder, and Frances MT Brazier. A resource negotiation infrastructure for self-managing applications. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pages 381–382. IEEE, 2005.

[98] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADEa FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.

[99] Maria Ganzha, Marcin Paprzycki, Amalia Pirvanescu, Costin Badica, and Ajith Abraham. JADE-based multi-agent e-commerce environment: Initial implementation. *Analele Universitatii din Timisoara, Seria Matematica-Informatica*, 42:79–100, 2005.

[100] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing multi-agent systems with JADE*, volume 7. John Wiley & Sons, 2007.

[101] Ernest Friedman. Jess in action: rule-based systems in java. 2003.

[102] Erich Schikuta, Helmut Wanek, and Irfan Ul Haq. Grid workflow optimization regarding dynamically changing resources and conditions. *Concurrency and Computation: Practice and Experience*, 20(15):1837–1849, 2008.

[103] Elisabeth Vinek, Peter Paul Beran, and Erich Schikuta. A dynamic multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment. *Procedia Computer Science*, 4:166–175, 2011.

[104] Parnia Samimi, Youness Teimouri, and Muriati Mukhtar. A combinatorial double auction resource allocation model in cloud computing. *Information Sciences*, 2014.

[105] Werner Mach, Benedikt Pittl, and Erich Schikuta. A prediction model for the probability of SLA matching in consumer provider contracting of web services. *arXiv preprint arXiv:1401.2440*, 2014.

[106] Maria Leitner and Stefanie Rinderle-Ma. A systematic review on security in process-aware information systemsconstitution, challenges, and future directions. *Information and Software Technology*, 56(3):273–293, 2014.

[107] Stefanie Rinderle-Ma and Maria Leitner. On evolving organizational models without losing control on authorization constraints in web service orchestrations. In *Commerce and Enterprise Computing (CEC), 2010 IEEE 12th Conference on*, pages 128–135. IEEE, 2010.

[108] Werner Mach and Erich Schikuta. A generic negotiation and re-negotiation framework for consumer-provider contracting of web services. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, pages 348–351. ACM, 2012.

[109] Ralph Vigne, Werner Mach, and Erich Schikuta. Towards a smart webservice marketplace. In *Business Informatics (CBI), 2013 IEEE 15th Conference on*, pages 208–215. IEEE, 2013.

[110] Werner Mach, Benedikt Pittl, and Erich Schikuta. A business rules driven framework for consumer-provider contracting of web services. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 229. ACM, 2013.

[111] Werner Mach and Erich Schikuta. Toward an economic and energy-aware cloud cost model. *Concurrency and Computation: Practice and Experience*, 25(18):2471–2487, 2013.

[112] Werner Mach, Benedikt Pittl, and Erich Schikuta. A forecasting and decision model for successful service negotiation. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 733–740. IEEE, 2014.