FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Market-oriented Service Allocation in Utility and Cloud Computing

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht von

Ivan Brešković, MSc
Matrikelnummer 0849140

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Prof. Dr. Schahram Dustdar, Mitbetreuung: Dr. Ivona Brandić

Diese Dissertation haben begutachtet:

(Prof. Dr. Schahram Dustdar) (Prof. Dr. Jörn Altmann)

Wien, 5. Juni 2013

(Ivan Brešković, MSc)

Erklärung zur Verfassung der Arbeit

Ivan Brešković, MSc
Tendlergasse 3/1, 1090 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Acknowledgments

There are a number of people without whom this thesis might not have been written and to whom I am greatly indebted.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Schahram Dustdar who gave me the freedom and encouragement I needed to pursue my PhD. I am also thankful to Ivona Brandić, my co-supervisor and project leader, for introducing me to the topic and giving me the opportunity to work in her group. She has provided me with constructive feedback over the last three years and given me the support needed to pursue my research unimpeded.

I am particularly grateful to my PhD examiner Prof. Jörn Altmann for his patience, encouragement, enthusiasm, motivation and immense knowledge. He made my stay at the Seoul National University a very pleasant and productive experience. My gratitude also goes to Simon Caton for hosting me at the Karlsruhe Institute of Technology and for showing me how fun research can be.

I am very thankful to Michael Maurer, Vincent Chimaobi Emeakaroha, Dražen Lučanin, Toni Mastelić and Soodeh Farokhi - my colleagues and project members who have made my time at the Vienna University of Technology memorable. Thank you for your constructive discussions and support, insightful comments, and making our group life lively and enjoyable.

I have been lucky enough to have the support of many good friends who have been the main sources of inspiration, laughter and joy. I am especially grateful to Morena Livaković for being a true friend and a constant source of support and encouragement over the last 13 years; Hrvoje Ribičić for being the most reliable and inspirational person in my life; and Florian Bader for his endless patience, support and belief in me. I would also like to thank several other friends who have made Vienna a very special place over all those years. Thank you all for being there to listen, drink and console. Life would not have been the same without you.

I owe a lot to my family, mainly to my father, my sister and my newly born brother, who have encouraged me and helped me at every stage of my personal and academic life, and longed to see this achievement come true. Thank you for instilling a love of creative pursuits and science within me, all of which is the ground of my accomplishments. This thesis is dedicated to you.

Ivan Brešković
June 2013
Vienna, Austria

Abstract

Utility and cloud computing are paradigms that offer computational services (e.g., software, hardware and infrastructure) in a manner similar to utilities such as water, electricity, gas and telephony - on demand, on a pay-as-you-go basis, and without regard to where the services are hosted or how they are delivered. Similar to some of these common utilities (e.g., electricity and crude oil), economic-oriented allocation, trading and quality of service (QoS) based scheduling of resources is executed over electronic market platforms named *cloud marketplaces*. However, despite numerous research works focusing on construction of a cloud marketplace, its implementation challenge is still awaited. Existing approaches are static and not capable of reacting to the dynamic nature of the underlying paradigm's context: high diversity of services and service types, and unpredictable user base and user behavior.

In this thesis, we address the issue of market diversity and dynamism through a novel methodology for the management of electronic marketplaces - *autonomic markets* - as a new way of thinking about the engineering of electronic market platforms. We argue that adding autonomic capabilities to market platforms will enable the establishment of fault-tolerant and reliable marketplaces, which in immature or changing paradigms are vital to counteract elements of uncertainty.

We begin our journey towards this vision by presenting a monitoring methodology, which includes a series of market goals from relevant literature, sets of extractable metrics for a market platform, and how to map (i.e., combine and transform) metrics to assess goal performance such that autonomic adaptation of the market could be undertaken if the current performance is insufficient.

Additionally, we explore open challenges of service discovery and selection in autonomic cloud markets. We argue that a large market diversity is one of the main causes of high transaction costs, poor market liquidity, and economic inefficiency of cloud marketplaces. We address this problem by channeling demand and supply into a limited number of adaptable ("standardized") services. Throughout the process of standardization we apply clustering algorithms and adaptation methods to select and manage standardized services that are structured based on the users' demands and offerings. Using a novel liquidity model that we introduce, we ensure that market liquidity is always maximized. Finally, we facilitate the process of discovering services in the cloud marketplaces by introducing methods for automatic matching and selection of cloud services.

To evaluate usefulness and feasibility of our approach, we build and herewith present conceptual and implementation details of an agent-based market simulator. Using this tool, we

simulate several realistic market scenarios and illustrate how important it is to be able to adequately monitor a market where sudden changes can lead to painful consequences. Furthermore, we show that such phenomena can be detected by our monitoring model.

In addition, we simulate a continuous trade between agents in both standardized and differentiated goods markets and demonstrate that standardization of goods in small markets (i.e., small number of market participants) may hurt market efficiency in terms of market liquidity, welfare, and transaction cost, but it brings enormous savings and benefits in the markets where the demand and the supply are sufficiently high.

Kurzfassung

Utility und Cloud Computing sind neuartige Paradigmen, die Rechenressourcen als Services (z.B. Soft- und Hardware, IT-Infrastukturen) in einer ähnlichen Art und Weise anbieten wie derzeit Wasser, Strom, Gas und Telefon angeboten werden – auf Abruf und mit genauer Dienstabrechnung unabhängig von der tatsächlichen Lokation der Services. Ähnlich wie bei Gas oder Erdöl werden bei der ökonomischen Ressourceallokation, und dem QoS-basierten Scheduling spezielle Marktplattformen vorausgesetzt, die als Cloud Märkte definiert werden. Obwohl es derzeit sehr viele Arbeiten zum Aufbau und zur Definition von Cloud Märkten gibt, ist die genaue Implementierung von solchen Märkten noch immer eine offene Fragestellung. Die derzeit existierenden Ansätze in diesem Bereich sind statisch und können nicht auf die dynamische Natur der Cloud Märkte reagieren, wie z.B. auf die hohe Diversität der Services, Service-Typen und das nicht vorausplanbare Benutzerverhalten.

In dieser Dissertation beschäftigen wir uns mit der Diversität der Märkte basierend auf einer neuartigen Methodologie für das Management von elektronischen Marktplätzen, den sogenannten autonomen Märkten. Dabei stellt diese Methodologie einen neuen Ansatz für das Engineering von elektronischen Marktplätzen. Autonome Eigenschafen ermöglichen dabei fehlertolerante und zuverlässige Marktplätze zu definieren. Dies ist notwendig, um den Unsicherheitsfaktoren der elektronischen Märkte entgegenzuwirken.

Als ersten Schritt präsentieren wir eine neuartige Monitoring Methodologie, die eine Vielzahl von Performanzindikatoren zum Aufbau eines elektronischen Markts voraussetzt und auch die entsprechenden Metriken berücksichtigt, die notwendig sind um diese Indikatoren zu messen. Dabei definieren wir auch wie diese Metriken auf die Indikatoren übersetzt werden können (z.B. Kombination und Transformation) um zu entscheiden, ob die derzeitige Performanz des Marktes ausreichend ist.

Wir beschäftigen uns auch mit der Auffindung der Services und einer automatischen Selektion der Services in Cloud Märkten. Des weiteren argumentieren wir, dass eine sehr hohe Diversität des Marktes eines der wichtigsten Gründe für die hohen Transaktionskosten, schlechte Marktliquidität und die ökonomische Ineffizienz der Märkte ist. Dieses Problem lösen wir durch die Bündelung von Angebot und Nachfrage in eine limitierte Anzahl von adaptierbaren (standardisierten) Ressourcen. Um standardisierte Services zu definieren und zu managen verwenden wir die Methodik der Clustering Algorithmen und diverse Adaptierungsmethoden die auf Angebot und Nachfrage der Benutzer basieren. Dabei verwenden wir ein neuartiges Model für das Management der Liquidität, das sicherstellt, dass ein Maximum an Liquidität erreicht wird.

Letztendlich ermöglichen wir durch die Verwendung der automatischen Vergleiche und Selektion einen automatischen Auffindungsprozess von Services in einem Cloud Markt. Um die Nutzbarkeit unseres Ansatzes zu evaluieren, diskutieren wir ein Konzept für einen agentenbasierten Market-Simulator. Mit diesem Tool simulieren wir verschiedene realistische Markt-Szenarien und illustrieren wie wichtig es ist auf eine adäquate Art und Weise den Markt zu überwachen - wobei schelle und unerwartete Marktveränderungen zu schwerwiegenden Konsequenzen führen können. Des Weiteren simulieren wir den kontinuierlichen Handel zwischen den Agenten in standardisierten und nicht standardisierten Märkten. Dabei demonstrieren wir, dass die Standardisierung in kleinen Märkten (d.h. sehr geringe Anzahl der Marktteilnehmer) unter Umständen die Markteffizienz stark beeinflussen kann. Sie kann z.B. eine schlechtere Marktliquidität, schlechtere ökonomische Wohlfahrt, und hohe Transaktionskosten hervorrufen. Gleichzeitig ruft diese Standardisierung aber auch viele Vorteile hervor, wie z.B. enorme Einsparungen, wenn Angebot und Nachfrage entsprechend hoch sind.

Contents

Previous publications

This thesis is based on work published in scientific conferences, workshops, journals and books chapters. For reasons of brevity, these core papers, which build the foundation of this thesis, are listed here once and will generally not be explicitly referenced again. Parts of these papers are contained in verbatim. Please refer to Appendix A for a full publication list of the author of this thesis.

Refereed publications in journals

1. Ivan Breskovic, Jörn Altmann, and Ivona Brandic. Liquidity Management in Utility and Cloud Computing Markets. Electronic Markets, Elsevier. (under review)
2. Ivan Breskovic, Jörn Altmann, and Ivona Brandic. Creating Standardized Products for Electronic Markets. Future Generation Computer Systems: The International Journal of Grid Computing and eScience, Elsevier (2012), DOI: 10.1016/j.future.2012.06.007

Refereed publications in conference proceedings

3. Ivan Breskovic, Ivona Brandic, and Jörn Altmann. Maximizing Liquidity in Cloud Markets through Standardization of Computational Resources. 7th International Symposium on Service Oriented System Engineering (SOSE 2013), 25-28 March, San Francisco, USA
4. Simon Caton, Ivan Breskovic, and Ivona Brandic. A Conceptual Framework for Simulating Autonomic Cloud Markets. 3rd International Conference on Cloud Computing (CloudComp 2012), 24-26 September 2012, Vienna, Austria
5. Christoph Redl, Ivan Breskovic, Ivona Brandic, and Schahram Dustdar. Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets. The 13th ACM/IEEE International Conference on Grid Computing (Grid 2012), 20-23 September 2012, Beijing, China
6. Ivan Breskovic, Christian Haas, Simon Caton, and Ivona Brandic. Towards Self-Awareness in Cloud Markets: A Monitoring Methodology. 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011), 12-14 December 2011, Sydney, Australia (Best Student Paper Award)

7. Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Schahram Dustdar. Cost-Efficient Utilization of Public SLA Templates in Autonomic Cloud Markets. 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011), 5-8 December 2011, Melbourne, Australia

8. Michael Maurer, Ivan Breskovic, Vincent C. Emeakaroha, and Ivona Brandic. Revealing the MAPE Loop for the Autonomic Management of Cloud Infrastructures. Workshop on Management of Cloud Systems (MoCS 2011), in association with the IEEE Symposium on Computers and Communications (ISCC 2011), 28 June - 1 July 2011, Kerkyra (Corfu) Greece

9. Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. Towards Autonomic Market Management in Cloud Computing Infrastructures. International Conference on Cloud Computing and Services Science (CLOSER 2011), 7-9 May 2011, Noordwijkerhout, The Netherlands

Book chapters

10. Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. Achieving Market Liquidity Through Autonomic Cloud Market Management. In Cloud Computing and Services Science (Service Science: Research and Innovations in the Service Economy), pp. 91-110. Editors: Ivan Ivanov, Marten van Sinderen, Boris Shishkov. Springer, 2012

CHAPTER 1

Introduction

A constant need for innovations and improvements of information systems has brought us to a period of fast development of information technologies (IT) and, finally, closer to the long-held dream of computing as a utility. Although sometimes observed as a market ploy by huge vendors to convince users to buy more products, utility computing is much more than that - it is a redesign of the fundamental technologies and infrastructures found in the today's data centers. Its core promise is turning computational resources (e.g., storage, networks, and applications) into services that IT will be able to deliver and charge for in the manner similar to the traditional utilities such as water, gas, electricity, and telephony. This idea was first suggested by John McCarthy of MIT as early as 1961, when he had said [2]:

"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

Computer science has since then rapidly advanced and numerous methodologies have been proposed, such as distributed systems, grid computing, virtualization technologies, and parallel computing [17, 91, 159, 175, 225]. However, in the today's business requirements, these technologies are often inefficient due to their lack in flexibility, scalability, and cost-effectiveness. As a means to address these issues and add new features to the concern of resource allocation and provisioning, cloud computing has emerged.

It is difficult to come up with a precise and comprehensive definition of cloud computing. Many individuals and organizations have made general statements about cloud computing and its advantages and weaknesses [56, 205, 228]. At the heart of it is the idea of highly scalable computing resources provided on demand and for the broad customer base [29,41]. It represents a business model that encompasses a variety of systems and technologies as well as service deployment models and business models. One of the most commonly cited definition of cloud computing is the one by the U.S. National Institute of Standards and Technology (NIST) that states: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources - for example, networks, servers, storage, applications and services - that can be rapidly provisioned and released with minimal management effort or service provider interaction" [150].

Cloud computing is commonly described by its five essential characteristics [150]: (1) on-demand self-service, i.e., a user can provision resources (e.g., software, infrastructure, network, etc.) when needed automatically and without requiring human interaction with each service provider; (2) broad network access, i.e., services are available over the network and accessed through standard mechanisms; (3) multi-tenancy, i.e., provider's resources are pooled to serve multiple consumers using a multi-tenant model; (4) rapid elasticity, i.e., services can be elastically provisioned and released to rapidly scale with demand; and (5) monitoring, i.e., systems are controlled and optimized on a pay-as-you-go basis. Terms by which the cloud services are "rented" are determined by Service Level Agreements (SLAs). SLAs are legally binding agreements between the two parties that state functional and non-functional performance promises made by a provider and fines for performance failures.

There are several cloud deployment models, where the most common include private, public, and hybrid clouds [140, 183, 226]. Private cloud infrastructure is deployed privately behind a firewall and used by a single organization comprising multiple consumers. Public cloud infrastructure is provisioned for open use by the general public and includes well defined payment and accounting mechanisms. Finally, hybrid cloud infrastructure is a composition of the two previous models. There are also numerous service delivery and deployment models in cloud computing [24, 48, 133], where the most common include: (1) software as a service (SaaS), i.e., offering provider's applications running on a cloud infrastructure; (2) platform as a service (PaaS), i.e., offering consumer to deploy an arbitrary application on the provider's cloud infrastructure using programming languages, services, libraries, and tools supported by the provider; and (3) infrastructure as a service (IaaS), i.e., offering consumers to run and deploy arbitrary software on a set of fundamental computing resources such as storage, network and processing power.

In order to achieve the status of computing as a utility, numerous research challenges are still left to the researchers. In this thesis, we investigate one such challenge: *automated service discovery and selection*. In particular, we examine how cloud users can discover and select best-fitting cloud services in the largely diverse and fragmented network of cloud providers. Through our discussions and studies of existing literature, we derive methods and frameworks for service discovery and selection that allow systems and software agents to make use of another's services without the need for continuous user intervention. We also demonstrate how the current methods of human-generated and human-consumed SLAs are not sufficient to evaluate, compare, and select SLAs and how a step towards standardization of SLA terminology could allow service agreements to be evaluated mechanically, thus reducing costs to consumers and providers.

To shortly introduce the work carried out in this thesis, in Section 1.1 we present our problem statement, detailing the motivation for our work. In Section 1.2, we present research questions that will be addressed in this thesis. Major contributions of the thesis are summarized in Section 1.3. Finally, in Section 1.4 we present the organization of the remainder of the thesis.

1.1 Problem statement and research objective

Utility and cloud computing paradigms have been since their beginnings inspired by the electrical power grid's reliability, ease of use, and pervasiveness [40]. Motivated by the success of this model, computer scientists have developed an analogous infrastructure - a *computational* power grid, which first lead to grid computing, and later to cloud computing [197]. These paradigms are comparable in numerous ways: they both offer resources (electricity and computation) as a utility, on-demand, without regard to where they are hosted or how they are delivered [48].

Currently fragmented, static, and shapeless cloud landscape hinders the paradigm's ability to fulfill its promise of ubiquitous computing on tap and as a commodity: prices are fixed; service level agreements non-negotiable, but tiered into categories like gold, silver and bronze; and users need to rely largely on search engines for provider discovery. To solve this issue, it is necessary to develop efficient methods for dynamic resource management, i.e., resource allocation, pricing, trading, and quality of service (QoS) based scheduling. In electrical grids, this task is given to the electronic market platforms named electricity markets. Similarly, management of computational resources is performed by *cloud markets*[1] [42, 116, 185, 214]. Cloud markets are computational platforms that allow resource consumers and providers to express their requirements and facilitate the realization of their goals [40]. Furthermore, they provide methods for resource discovery and strategies for dynamically scheduling applications at runtime depending on various user-defined QoS requirements, including costs, availability, and capability. Figure 1.1 presents a simplified overview of how a cloud marketplace with a support for these tasks should look like. Note that this figure also depicts a market component for adaptation, which will be discussed in detail shortly.

The implementation challenge of a cloud marketplace is still awaited, despite enough projects concentrating on cloud marketplace construction. Our argument is not that the technical challenges addressed by such projects are not important developments, but that they are not sufficient to completely capture a cloud market. We argue that such marketplaces do not efficiently cope with the paradigm's (and its followers') highly dynamic context such as high product variability, unpredictable participant behavior, and the emergence of new actors and actor types.

In the following, we identify five major challenges facing the implementation of electronic market platforms for utility and cloud computing.

Problem 1

Currently existing methodologies for the management of cloud computing marketplaces are static and unable to react to the dynamic characteristics of the underlying domain.

New challenges from a surge in Internet connected devices and digital ecosystems will emerge. Consequently, we do not yet know what situations will arise for future marketplaces, especially cloud marketplaces. We, therefore, need novel theories and paradigms to facilitate and control the next generation of electronic market platforms. A key challenge is that we do not know the most fitting anatomy of an appropriate market platform for these domains. Even assuming that an adequate platform has been designed and implemented, a subtle or disruptive

[1] In this thesis, we interchangeably use the terms cloud market, cloud marketplace, and cloud market platform.

change within the domain can mean that the platform no longer satisfies its domain requirements. Today, electronic market platforms do not even consider the prospect of online adaptation. Instead, they are reasonably static and not designed to handle changes in their operational environment or elements of uncertainty in their architecture.

In this thesis, we argue that the market engineering process for a cloud marketplace cannot extend traditional approaches. Instead, it is our vision to define a new methodology for the management of electronic markets where elements of dynamism or uncertainty are observable. We believe that the market engineering process in such scenarios can be orchestrated through the definition of market goals in combination with autonomic capabilities (self-optimization, self-configuration, self-healing and self-protection [125]). In other words, to build in dependability via the capability of adaptation at design time. This enables a market platform to autonomically adapt its configuration at runtime to satisfy a set of goals (e.g., levels of liquidity, privacy, welfare, energy efficiency and stability) and thus retain system-wide performance with respect to these goals. In Figure 1.1, this task is given to the component named "Market adaptation".

By making an electronic market platform autonomic, we hope to enable the creation of market platforms that evolve beyond their initial design principals and "learn" or evolve towards their (near) ideal or stable configuration(s). In making market platforms autonomic, we can begin to explore, analyze and understand how market platforms can evolve as well as the impact of different market models and goal sets.

Problem 2

Due to the lack of monitoring methodologies, it is not possible
to identify and react to potential inefficiencies in electronic marketplaces.

In our vision of autonomic market platform, market quality and performance are characterized by levels of satisfaction of certain market goals. In order to fulfill this idea, it must be viable to define, capture and manage knowledge from the market platform that represents these goals. However, despite the large importance of this task, there exists no methodology for monitoring economic capabilities of electronic market platforms. Through the several decades of research in electronic commerce and market design, numerous measures for expressing market quality have emerged [15, 19, 23, 111, 113, 115]. However, most of these works focus only on financial assets (e.g., stocks), which are relatively simple, standardized and well understood and, therefore, significantly differ from diverse and partly substitutable computational services. Furthermore, these works usually discuss quality measures on a theoretical level, neglecting the importance of implementing a monitoring framework for measuring the overall market performance. Finally, due to the complexity of some of the performance indicators and market goals (e.g., market liquidity), the research community has not yet defined a single unified method for capturing these measures that are applicable in commodity markets, and not only for stocks and financial assets.

It is our goal to define a monitoring methodology for assessing quality of cloud marketplaces that includes a series of realistic market goals, the sets of extractable metrics from a market platform, and how sets of metrics are combined and transformed to access goal performance. The monitoring methodology presents the first step towards the vision of autonomically adaptable electronic market platform through allowing the market to identify potential inefficiencies and threats. In Figure 1.1, the task of market monitoring is depicted as "Observe" step of the market adaptation layer.

Problem 3
Cloud landscape is fragmented, heterogeneous and diverse,
which may lead to poor market liquidity and trading activity.

Considering service allocation and provisioning, which are the main tasks of cloud markets, it is not sufficient to investigate only the implementation and adaptation of market platforms, but also the services that are traded within the market. Cloud markets comprise a virtually indefinite variety of computational service types. This is caused by the large diversity in terms of service specifications, resource utilization, and the ways of delivering computational services. First and foremost, there is a vast variety of service types due to the numerous Everything-as-a-Service (XaaS) delivery models (e.g., software, hardware, platform, and network as a service). Second, usage scenarios, pricing strategies, and allocation methods may differ to a large extent for each of the service delivery models (e.g., software services may be significantly less scalable and flexible than infrastructure units). Finally, even if assuming a market platform for trading services for only one delivery model, the number and heterogeneity of providers and services are often extremely large compared to other markets [3, 33].

Due to the extensive variety of services (goods), discovering and selecting a fitting service is often costly and the likelihood of finding a service that matches a buyer's requirement is likely to be very low. These issues result in poor market performance in terms of market liquidity [33, 148], which is an important measure of market quality. Market liquidity describes how easily and quickly it is possible to trade a certain volume of a considered good without causing a significant movement in its price. A market must ensure a proper level of market liquidity in order to attract traders and work efficiently [182].

Since liquidity depends on the diversity of goods that are traded in the market, it is reasonable to assume that reducing the heterogeneity of computational resources would have a positive effect on liquidity in cloud markets. This approach - standardization of products - showed positive effects on many other market types [131]. Standardization makes products commodity goods and brings numerous additional benefits, such as independence of single suppliers (commoditization), compatibility, interoperability, safety, repeatability, and quality [179].

Following the positive examples from the history of commodity markets, we begin to explore how standardization can be applied to computational goods to reduce diversity and increase liquidity in cloud computing marketplaces. Through standardization of computational goods, demand and supply are channeled into a limited number of standardized services, which in turn "homogenizes" the market. Standardization of cloud market products, i.e., computational services, is possible due to their substitutability (i.e., fungibility). Namely, computational services are fungible as they can be substituted by other, relatively similar services. By analyzing services demanded and supplied, it is possible to acquire knowledge about users' needs in the market and perceive the actions they perform. This knowledge facilitates apprehending the relations between different services and discovering what services are mutually interchangeable (and to which extent), which is a key requirement for product standardization. In Figure 1.1, the task of resource standardization is performed by the component named "Resource management".

Problem 4
Cost of service discovery and selection may present
a barrier for potential users to join the market.

In the "traditional" differentiated goods markets, the extensive variety of services often causes extremely high costs for market users due to the complex process of service discovery: users must manually check numerous offerings before finding the closest ones to their needs. The complexity of this task, as well as poor market liquidity, may repel potential users to join the market. Through the standardization of diverse and heterogeneous computational services, cost of discovering and selecting appropriate services is significantly reduced, but not fully vanished. As long as human intervention is needed for these tasks to be executed, the idea of utility computing cannot be fully achieved. In particular, in the long-held dream of computation as a utility, computing systems are able to independently and inexpensively locate and purchase certain services in order to achieve given tasks.

The goal of automatic and autonomic SLA discovery is particularly challenging as there exists no standard for specification of SLA-based services and service requirements. During the last decade, numerous research works have discussed standardization of SLA specifications and ontology-based solutions for matching SLAs and SLA elements [8, 78, 86, 104, 154, 161, 180]. However, these techniques are costly and unable of autonomic adaptation to the changing conditions of the underlying paradigms.

In the course of this thesis, it is our goal to define a set of methods and processes for automatic and autonomic execution of service discovery and selection tasks ("SLA matching" in Figure 1.1). Through feedback-based machine learning, these methods perform self-modification with the goal of minimizing and, finally, completely dismissing the need for human intervention.

Problem 5
There exists no simulation framework for testing hypotheses and methods
for adaptation of market platforms.

In the last years, several research groups and institutions have focused on the implementation of market platforms for allocating and scheduling computational resources in computing paradigms such as utility, grid and cloud computing. Many of these research works have resulted in open-source market frameworks, such as [3,39,123,130,158,184,214]. The majority of these works discusses implementations of markets, economics-aware service platforms, service interactions, business models, consumer/provider discovery, transaction facilitation, and (autonomic) policy management. However, none of these frameworks allows the market platform to change at runtime.

In order to verify our hypotheses and methods for creation of autonomic and adaptive market platforms, a controllable, flexible and reliable market simulation tools is necessary. Currently existing tools do not allow us to move towards more realistic scenarios (e.g., adding new trading artifacts, participant types, changes in participant behavior, market growth or contraction, etc.), nor they allow painless extension and control. Moreover, bringing certain level of dynamism in the majority of the existing market simulation tools is virtually impossible as they assume static participation such as the number of users and their demand remains fixed.

1.2 Research questions

Based on the problem statement described in the previous section, we derive five main research questions that are addressed by this thesis.

Research question 1
How to guarantee market stability and sustainability
despite dynamic changes within the (cloud computing) domain?

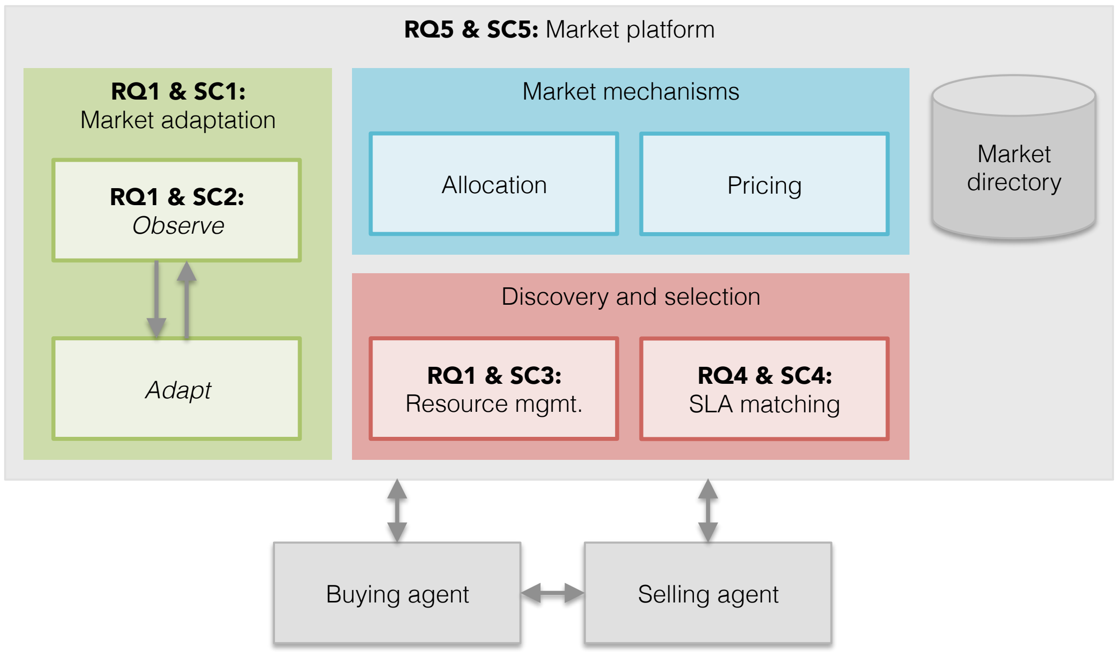
Although the current cloud realization offers a simple, fast and inexpensive way to bring consumers and providers together, it also suffers from many challenging situations. These include low market liquidity (caused by a broad resource variability and a low number of market participants), a fragmented array of independent providers, a few standard mechanisms for unilateral provider adoption and use, and an ever-changing user base and actions users perform in the market. In order to address these challenges, a cloud market should be dynamic and adapt to the current needs of its participants as well as address the impacts these requirements have upon the market itself. Although in this thesis we use the cloud computing paradigm as a use case, it should be noted that these requirements can be posed to any other type of electronic market as well. This research question discusses the “Market adaptation” component of Figure 1.1. It addresses Problem 1 presented in the previous section and will be fully elaborated in Chapter 3.

Research question 2
How to measure quality of an electronic marketplace?

In order to create a dynamic market platform that satisfies the needs of its participants and to enable the idea of adaptable markets, it is necessary to understand the meaning of market quality and performance and to define methods for their assessment. Current monitoring tools for electronic markets, clouds, and other complex systems do not fully capture economical behavior of the underlying paradigms. In order to enable the idea of a self-adapting marketplace, we must be able to observe its performance from different perspectives and define a complete monitoring model that includes infrastructural, economical, and other implementation-specific market properties. This research question is related to Problem 2 presented in the previous section. It discusses the ”Observe” task of the market adaptation component from Figure 1.1 and will be presented in detail in Chapters 3 and 4.

Research question 3
How to cope with the liquidity risks caused by
diversity and heterogeneity of cloud services?

A cloud market faces a huge diversity in services, service delivery types, and provider-based constraints and requirements. This diversity can cause serious problems to market stability and liquidity. In particular, high number of service types means that it is hard and costly to find a service that is required, thus reducing the utility of market users while increasing the transaction costs in the market. Market liquidity is an important measure of market quality and a guarantee of market efficiency. An illiquid market may negatively impact attractiveness and, therefore,



SC1: Conceptual design of an autonomic market platform (Chapter 3)
SC2: Monitoring methodology for electronic market platforms (Chapters 3 and 4)
SC3: Resource management and standardization in electronic marketplaces (Chapter 4)
SC4: Automated service discovery and selection of SLA-based resources (Chapter 5)
SC5: Implementation of an autonomic market simulator (Chapter 6)

Figure 1.1: Scientific contributions (SC) and organization of the thesis

repel potential consumers and disadvantage new providers. Therefore, in order to ensure market stability and sustainability, the marketplace must retain a high level of liquidity and regulate service diversity. This research question is related to the implementation of the "Resource management" component of Figure 1.1. It follows the discussion of Problem 3 presented in the previous section and will be elaborated in Chapter 4.

Research question 4
How to facilitate service discovery and encourage traders to join the market?

Market participation can be often costly: users must specify their service requirements, map the differences between their specifications and the ones of their potential providers, and manually select the best-fitting service offering. In a market trading numerous diverse services, this process can be particularly complex and may repel potential consumers and providers. It is, therefore, necessary to enhance and facilitate these tasks in the cloud marketplaces by offering automated methods for service discovery and selection. This research questions is associated to Problem 4 presented in the previous section. It discusses the "SLA matching" component of Figure 1.1 and will be elaborated in Chapter 5.

Research question 5
How to design a cloud marketplace capable of
autonomic adaptation and trading standardized services?

To explore the vision of an autonomic market, we need an experimental platform. Naturally, it is not possible to plainly map existing markets to fulfill this task. Therefore, an appropriate research methodology for their study is simulation, as it enables the creation of what-if scenarios and the ability to observe how autonomic adaptation evolves a market over time. To achieve this goal, we need an autonomic market simulation tool. Through simulation, we can implement economic and management models of autonomic markets to access their performance (with respect to goal fulfillment), tractability and feasibility for different adaptation strategies. Although real-life markets cannot be mapped directly for an autonomic market, their traces as well as event and trading catalogues can act as input data as a means to drive specific what-if scenarios. This research questions is associated to Problem 5 described in the previous section. It is related to the overall collaboration and implementation of the components presented in Figure 1.1 and will be discussed in detail in Chapter 6.

1.3 Scientific contributions

According to the research questions presented in the previous section, we highlight the following scientific contributions of this thesis. Each of the contributions is marked in Figure 1.1. These contributions have been previously published in several academic journals, conference proceedings, and books. We herewith specify for each contribution the references where it has been published.

Scientific contribution 1
Conceptual design of an autonomic market platform

To address the dynamic nature of user requirements and new services in cloud markets, we outline a methodology for the definition and management of cloud market platforms. In particular, we use the autonomic capabilities such as self-optimization, self-healing, self-configuration, and self-protection to create an *autonomic marketplace*. Our methodology is intended to facilitate adaptation at many levels of a market platform, where examples include: autonomic (economic) mechanism design, self-regulation, fault tolerance as well as autonomic market (re)engineering. In essence, we explore and discuss how the extended autonomic control loops can be applied to a complex array of parameterizable (hence adaptable) economic components, where each component can be imagined as a traditional managed element within the Autonomic Computing paradigm. By applying our methodology, a market platform can be considered as an interconnected network of managed autonomic elements. The autonomic market has the ability to change its initial design and evolve towards a stable configuration. Contribution 1 addresses Research question 1 presented in the previous section. It has been previously published in [35, 52] and will be discussed in detail in Chapter 3.

Scientific contribution 2
A monitoring methodology for assessing
quality and performance of electronic markets

We present a novel methodology for the monitoring of cloud markets. In detail, we identify available and measurable *low-level* monitoring data that is useful for (autonomic) markets and the necessary mappings to transform these metrics into *high-level* indicators for a given set of market goals. As a result of this study, we introduce monitoring sensors as an extension to GridSim, a well-known grid simulation tool, that are able of observing infrastructure-based information, market-mechanism-based properties, and other non-mechanism related data. We particularly focus on market liquidity and derive novel liquidity measures for autonomic markets trading adaptive cloud resources based on the existing measures from the financial literature. Additionally, we evaluate our monitoring methodology using realistic market scenarios. Contribution 2 addresses Research question 2 presented in the previous section. It has been previously published in [34, 35, 52] and will be fully elaborated in Chapters 3 and 4.

Scientific contribution 3
An approach for improving market liquidity through
"standardization" of SLA-based computational resources

To counteract the problem of large diversity of cloud services and properties of providers' offerings, we present an approach of *resource standardization*. In particular, we channel demand and supply into a limited number of services that satisfy the needs of as many market participants as possible. This approach assumes that computational services are mutually interchangeable (to a certain degree) and that reducing the number of service homogenizes the market and increases its activity and liquidity. In this thesis, we discuss methodologies and theories for creation and adaptation of standardized services and observe their performance in terms of several evaluation criteria in a controlled simulation environment. Contribution 3 addresses Research question 3 discussed in the previous section. It has been previously published in [33, 36–38] and will be discussed in Chapter 4.

Scientific contribution 4
Methods for automated SLA matching and selection

Service discovery and selection often incur significant market participation costs to users and may present a great obstacle in market stability and activity. In this thesis, we present methods for automated service discovery and selection. In particular, we discuss several feedback-based machine learning algorithms that match equivalent elements of differing SLAs and select best-fitting SLAs in terms of specification of services and required/offered values of quality of service (QoS) parameters. Using a simulation-based evaluation scenario, we demonstrate the benefits of this approach in terms of lower effort and cost of finding and selecting SLA-based services. Contribution 4 addresses Research question 4 discussed in the previous section. It has been previously published in [171] and will be presented in Chapter 5.

Scientific contribution 5
Design and implementation of an autonomic market simulation tool

In this thesis, we present the conceptual design and implementation details of an experimental platform for evaluation of the autonomic market vision. Although built as a tool for simulating cloud markets, its components can be easily used in the real-world marketplace implementation. Using our market simulator, we implement various economic models to evaluate their performance and feasibility with respect to the market adaptation strategies. Contribution 5 addresses Research question 5 presented in the previous section. It has been previously partly published in [52] and will be fully elaborated in Chapter 6.

1.4 Organization of the thesis

According to the research questions (RQ) and scientific contributions (SC) presented in the previous sections, the remainder of the thesis is organized as follows (also marked in Figure 1.1).

- Chapter 2 presents related work, compares it to the work carried out in this thesis, and outlines the enhancements that this theses has made. By studying the state of the art in market-oriented utility and cloud computing, we organize related work into the following categories: market adaptation, market-oriented resource management, monitoring in (cloud) markets, service discovery and selection, and experimental market platforms.

- Chapter 3 presents a vision of autonomic cloud market platforms as discussed in RQ 1 and SC 1. It first elaborates the motivation for this approach and details the steps of the autonomic MAPE-K loop that is used to implement it. Additionally, it presents a novel monitoring methodology for cloud markets as discussed in RQ 2 and SC 2, which is a first steps towards the vision of autonomic markets, and discusses its implementation as an extension to an existing grid simulation tool.

- Chapter 4 introduces an approach of automated standardization of computational resources in cloud markets as discussed in RQ 3 and SC 3. First, it details the liquidity risks in cloud markets and derives a set of measures for monitoring liquidity in electronic markets. Second, it examines how liquidity problems can be solved by reducing market heterogeneity and discusses the design of a market to support trading these services. Finally, it presents methods and algorithms for managing adaptive cloud resources and formalizes a cost-benefit analysis of this approach.

- Chapter 5 presents an approach of using machine-learning methods to match SLA-based service specifications with the goal of automatically recommending best-fitting service offerings to the market users and, therefore, reducing them cost of market participation. This contribution has been discussed in RQ 4 and SC 4.

- Chapter 6 discusses benefits and shortcomings of the existing market simulation tools and introduces a conceptualization and implementation details of an agent-based cloud market simulator. This contribution has been discussed in RQ 5 and SC 5.

- Chapter 7 presents the simulation-based evaluation of the methods described in all research questions and scientific contributions, i.e., in Chapters 3-5. For each approach, it gives a detailed description of the simulation environment and evaluation results. Finally, it summarizes all evaluation results and outlines the benefits and the shortcomings of the work carried out in this thesis.
- Chapter 8 presents the conclusion of the theses, discusses its limitations, and gives a critical reflection and an outlook into possible future work.

CHAPTER 2

Related work

In this chapter, we present related work, which we organize in the following categories: Section 2.1 presents the literature on autonomic adaptations in clouds and other similar systems. Section 2.2 describes the existing work related to the resource management in utility, grid, and cloud computing, including existing implementations and discussion of cloud marketplaces. Section 2.3 presents related work on monitoring in cloud computing and in electronic markets. In Section 2.4, we present the existing work on SLA discovery and selection and discuss adaptive specification of SLA-based services. Finally, Section 2.5 presents related work on implementation of experimental platforms for simulating electronic markets.

2.1 Market adaptation

Due to the growing importance of distributed systems (grids and clouds) in recent years, the scientific community focused on the theoretic foundations and research on how to make such systems adaptive and sustainable, often referring to the original self-* principles of autonomic computing [125]. While research was mainly visionary in the first years after the seminal Autonomic Computing article, these early works served as groundwork for more and more prototypical implementation of autonomic aspects in various systems. For example, [165] discuss the need for autonomic capabilities of distributed service systems and briefly outline the application of the self-* capabilities in this context. Today, most scientific work addresses technical issues to make systems autonomic, such as the development of negotiation protocols to make grid or cloud services self-adaptive [29], or considers autonomic service management frameworks without explicitly taking economic methodologies into account (e.g., [60, 134, 164]).

In contrast, research on autonomic systems that apply economic methods and considerations, as first proposed by [58], is still in its infancy. In particular, current research in this area often focuses on narrow and specific issues and therefore only partially considers the aspects needed for autonomic marketplaces. For example, a self-organizing resource allocation mechanism in dynamic Application Layer Networks is proposed by [192]. They do not, however, consider issues such as the adaptation of the market and the used mechanism itself depending on the available resources, which is a crucial element for potential autonomic marketplaces. [166] propose mechanisms that are able to adaptively adjust their parameters based on the past behaviour of participants. An example of economically-inspired market infrastructures is provided by [51] who present a self-optimizing infrastructure platform for service delivery using economic (congestion-based) pricing, but only consider the infrastructure level, and not the institutional level of the platform. [26] study the mapping of high-level business objectives to lower level objectives in order to enable autonomic optimization. However, they specifically study an autonomic DBMS and do not consider grid or cloud environments.

Besides the original self-* principles as described above, self-awareness (a key aspect within the context of monitoring methodologies for autonomic systems) is a more abstract concept that can be considered a building block of other principles. In order to facilitate self-awareness, a key aspect is the ability to monitor crucial attributes and other performance measures that provide information about the current state of the market or the platform. Performance monitoring is a widely used process that is applied in economics in areas such as principal-agent theory (monitoring the performance of agents in order to prevent hidden action, [129] pp. 121), public (government) institutions (i.e., institutions that are not directly involved in a competitive market [212]), and market performance with respect to market power and abuse in decentralized energy markets [170].

Another area that closely resembles the concept of self-aware markets is the monitoring of performance metrics in (Web) services or business processes. Monitoring the performance of Web Services is crucial due to their inherent dynamics and the complexities and dependencies that arise with their invocation in service composition. Run-time monitoring in this context has been addressed by several authors, e.g., [14, 16], and tries to provide methods and frameworks that address how certain attributes and metrics of Web Services can be continuously monitored. In an enterprise context, Key-Performance-Indicators (KPIs) are defined as attributes linked to the performance of the enterprise, and for each KPI target values are set according to management goals. KPIs, however, tend to be management oriented, as they represent high-level goals, and for this reason, need to be matched to lower-level metrics of the underlying involved business processes. [211] presents a framework that is able to derive dependencies between KPIs and the underlying metrics, in order to be able to identify the causes of KPI violation.

Although these monitoring models target runtime monitoring, they consider (single) service instances or enterprises rather than taking a market perspective. From a conceptual perspective, the approach that is most similar to ours is the mechanism by [166]. However, it lacks the needed detail on how such a monitoring infrastructure can be implemented. In this thesis, we address this point by presenting and evaluating our monitoring model as a precursor to autonomic cloud marketplaces.

There has also been work on generic autonomic frameworks. However, many approaches tend to be based on frameworks that are application specific. Some examples are given in [50] and [1]. The key limitation is that knowledge of the application domain is inherently embodied into the autonomic framework. As a result, these frameworks are not truly generic and lack portability to different domains, as their search space has been substantially pruned by means of their design. Other work has focused on specific components of the MAPE-K functional decomposition, such as specific strategies to implement the control loop, e.g., based on control theory [103] or utility models [135, 169]. Several works discuss the application of machine learning techniques such as case-based reasoning and rule-based approach for knowledge management in cloud environments [145–147]. However, although the presented approaches proved to be successful for changing resource configurations of virtual machines in clouds, their performance is dictated by the manual definition of adaptation rules. Consequently, they are only effective when the set of adaptation actions as well as their requirements and consequences are small, precise and constant. As it will be discussed in Chapter 3, this will not be the case in autonomic cloud markets, where many adaptation actions can be used to solve the same market inefficiencies.

To study of adaptation upon participant behavior a commonly used methodology is agent-based computational economics: the computational study of economies modeled as evolving systems of autonomous interacting agents to understand and simulate heterogeneous actors (agents) interacting in a variety of ways. Using this paradigm, economic problems can be understood as study emerging interactions among intelligent, self-interested agents, and leads to an understanding of the economy not from a macro level perspective as a system of aggregated demand and supply, but from a micro level as a result of the interaction of numerous independent individuals [196].

To study constraint solving for the interaction and evolution of market parameters a promising paradigm is chemical programming: the notion of computing as molecules representing data and procedures that float in a chemical solution and engage in reactions [13, 156]. Computation is realized as a series of reactions that transform data and potentially procedures. Computations are executed as long as reactions are possible; they are concurrent, independent, self-governed and guided by actual and local conditions. In other words, the paradigm provides a method to model a market as an organic system, where the specific properties of a market (contextual and institutional knowledge) are represented as properties of the chemical system.

2.2 Market-oriented resource management

2.2.1 Resource management in utility and cloud computing

Resource allocation and provisioning, as well as service negotiation and federation, have been discussed in numerous research works. For example, [59] introduced an agent-based approach for addressing federation problems in grids (e.g., resource selection and policy reconciliation) through automated agent negotiation. [176] introduced Claudia - a service management system that implements a new abstraction layer for the lifecycle of cloud services that allows for their automatic deployment and escalation depending on the service status. This abstraction layer can sit on top of different cloud providers, hence mitigating the potential lock-in problem and allowing the transparent federation of clouds for the execution of services. Garg. et al. [94] discuss the problem of service selection in utility and grid computing. The authors present two novel heuristics for scheduling parallel applications on utility grids that manage and optimize the trade-off between time and cost constraints. The performance of the heuristics is evaluated through extensive simulations of a real-world environment with real parallel workload models to demonstrate the practicality of our algorithms. [132] proposes the use of a cloud system as a raw computational on-demand resource for a grid middleware and provide a proof of concept by considering the DIET-solve grid middleware and the EUCALYPTUS open-source cloud platform. However, these works (and the majority of other related work on resource management) consider resource provision only from the provider's perspective and disregard cloud user's utility. Furthermore, they do not consider service discovery and selection through market-oriented platforms.

2.2.2 SLA management in research and existing cloud implementations

Over the past several years, many large IT companies have entered the cloud computing market by offering different types of services. Depending on the type of the service they offer, we can divide them into three groups based on the business model they offer: (1) infrastructure as a service (IaaS), e.g., [4, 5, 102, 119, 163, 198, 213]; (2) platform as a service (PaaS), e.g., [11, 70, 117, 181]; and (3) software as a service (SaaS), e.g., [101, 143, 152, 181, 201, 213]. The IaaS providers offer their computing resources on a pay-per-use basis in form of virtual machines, as it is the case with Amazon's EC2 service [4], or as a computing platform, as it is the case with the services offered by Tsunamic Technologies [203]. The shortcomings of the currently existing IaaS providers (as well as other cloud providers) is that despite the large variety of cloud services, they usually sell a single type of resources. For example, Amazon introduced only three derivations of their basic resource type [4]. SaaS providers offer their computing resources in combination with their own software components. Although they provide software services on a pay-per-use basis, integration of SaaS components with other solutions is often extremely hard due to the complexity of these services. Finally, PaaS providers offer computing platforms that allow users to create their own applications in combination with the supporting services of the provider. The goal of the PaaS model is to allow a seamless integration with the users' applications. However, this is still not the case as the standardization of common interfaces is still missing.

A shortcoming of the majority of the currently existing cloud providers, despite their business models and service types, is that they do not fully utilize the idea of service level agreements (SLAs). Namely, most of the providers (e.g., Microsoft Azure [213] and Amazon [4]) do not even provide their SLAs in one of the commonly used XML-based formats. Without allowing users' specifications of requirements in standardized formats, these providers can hardly participate in a cloud marketplace and, therefore, present a further threat towards cloud market fragmentation and illiquidity.

SLA management is an important technological aspect of cloud computing [173]. SLAs are legally binding contracts that represent a negotiated agreement between two parties, namely the service consumer and the service provider. An SLA defines a common understanding between the parties about different contract terms including responsibilities, guarantees, warranties, and penalties. Usually, an SLA in computing resource markets also specifies measurable metrics, the method for measuring those metrics, and the billing process.

Currently, there are two de facto standards for the specification of SLAs: (1) the Web Service Level Agreement (WSLA) specified by IBM [138, 208], and (2) the WS-Agreement defined and specified by the Open Grid Forum (OGF) [162, 216]. IBM's WSLA covers the requirements of commercial services while OGF's WS-Agreements focuses on the requirements of the scientific computing community (e.g., grid community). WSLA and WS-Agreement are successfully used in different projects jointly.

WSLA is an SLA specification language implemented as part of IBM Emerging Technologies ToolKit (ETTK) framework, which provides functionality for specification, creation, and monitoring of SLAs [85]. Each SLA expressed in the WSLA specification language contains three sections: (1) a section describing parties and the interfaces exposed to these parties, (2) a section containing the definitions of services, their operations, including service level attributes and metrics, and (3) an obligation section specifying service level objectives and guarantees. In WSLA there are two types of parties: signatory parties (service provider and service consumer), who sign the SLA, and supporting parties, who are authorized by those signatory parties to execute different activities (e.g., monitoring the metrics).

In contrast to WSLA, WS-Agreement is a XML language for specifying agreements as well as a protocol for advertising capabilities of service providers, and for monitoring compliance. Comuzzi et al. define the process for establishing SLA [73], which has been adopted within the SLA@SOI project [188]. The SLA@SOI project also proposes an architecture for monitoring SLAs, considering two requirements: the availability of historical data for evaluating SLA offers and the assessment of the capability to monitor the terms in an SLA offer [90, 168]. Koller and Schubert suggest autonomous QoS management using a proxy-like approach [126]. The implementation is based on WS-Agreement, whereby SLAs define certain QoS attributes (which a service has to maintain during its interaction with a specific customer).

2.2.3 Cloud marketplaces

Although the research in computing resource markets did not discuss the liquidity of goods in cloud computing markets, their perspective on computing goods is important. While the initial research on computing resource markets did not pay attention to the definition of computing resources at all, the latter research considered a simplified computing good definition.

The initial research consists of early grid market designs [39, 47, 158, 184]. It is generally believed that grid market designs were pioneered by Buyya et al. [44] in the Nimrod/G project. Buyya et al. continued their contributions in this field in [47], where they describe economic entities of a grid market. Similarly, GRACE provides an architecture for grid markets and outlined a market mechanism without defining the computing resource good [39]. Moreover, the process of creating agreements between consumers and providers has not been addressed. The latter research on grid markets considered a simplified version of a computing resource good. Developing the MACE exchange [184], the authors recognized the importance of developing a definition of a tradable good. They introduced abstract computing resources as services which can be traded. However, a detailed specification of a computing resource service has not been given and, hence, its effects on market liquidity cannot be assessed.

The Spawn market was envisioned to work with CPU time slices [207]. The fact that computing resources have been reduced to a single component is not realistic, as the CPU slice requirements depend on the CPU vendor due to different instruction sets. Similar to Spawn, POPCORN project provided another market for CPU-based resources [172]. The Tycoon market was developed before virtualization tools (e.g., Xen [217]) became widely used [130]. Initially, it worked with abstract computing cycles. Although it was planned to extend this market by making use of virtualization, the effort has been discontinued.

The SORMA project focused on fairness and efficient resource allocation [157, 158]. They identified several market requirements, such as allocative efficiency, budget-balance, truthfulness, and individual rationality [157]. However, they have not considered that a market can only function efficiently with a sufficiently large liquidity.

Beside these efforts, several market mechanisms have been developed [3, 74, 172, 174, 193, 204], and the idea of a resource definition has been mentioned [157]. However, further developments in this area have not been made. Furthermore, the SHARP marketplace has been introduced, which addressed a number of security concerns within a Grid market. It also took into account the fact that virtualization would have a large impact in a market [93]. The SHARP computing resources are defined as a `<type, count>` pair, where the count gives the quantity of resources corresponding to the type. In [122], the Shirako framework has been introduced, which is partially based on SHARP and is used to investigate resource allocations in computing resource markets. The authors even recognize that matching resources is far from trivial and that some requests for resources may have additional attributes that have to be considered.

GridEcon proposed a commodity market for cloud computing services [3, 174]. Although an explicit service level agreement for standardized cloud services [173], the cloud service requirements, and the requirements for trading have been defined and specified, the issue of adaptation of standardized goods has not been addressed. In the work on cloud computing value chains [153], many important issues of electronic markets (e.g., improved cloud pricing and licensing models) are discussed. However, while the diversity of virtualized resources was mentioned implicitly, the effect this diversity can have on the market has not been addressed.

Several other projects have presented frameworks for providing market-driven selection of computer servers based on the commodity market models. These projects include Faucets [123], G-commerce [214], Grid Markets Project [106], EPSRC GRAIL [105], and grid Market [142]. Similar to previously mentioned works, they have used only simplified definitions of goods and have not considered that a market can work efficiently only with a sufficient liquidity. Additional similar grid and cloud market models are described in [46, 64, 219]. A comprehensive survey on grid market systems is provided in [221].

In their analysis of future research within clouds, Weinhardt et al. [210] have listed many important issues (e.g., the development of an unified cloud API, the improved cloud pricing, and cloud licensing models) that need to be addressed. While the diversity of virtualized resources is mentioned implicitly, the effect this diversity can have on the market has not been addressed.

In addition to the introduction of virtualization in data-centers, several companies (e.g., Enomaly [83] or Fluid Operations [89]) now offer platforms to integrate in-house resources with externally purchased cloud computing resources. Consequently, these products allow users to turn their data center into a cloud and also allow users to act as providers for resource services. Using such software, data center operators could easily participate in an open cloud market. They could sell their excess capacity and purchase cloud resources from the open cloud market when demand is high.

Another company that provides a similar service is Zimory [229]. Its product turns a regular data center into an intra-company cloud allowing an efficient use of resources. It even allows its customers to sell spare capacity via its own marketplace (i.e., the Zimory Cloud). During times of high demand, resources can also be purchased via the same market place. Since Zimory only supports three virtualization mechanisms at present [229], the diversity of goods is already reduced dramatically. Furthermore, Zimory could limit the resource diversity using preconfigured virtual machines via its software package, thereby ensuring sufficient liquidity in the market.

2.3 Monitoring in cloud markets

Resource monitoring in cloud infrastructures is a common research topic in numerous research projects. For example, Fu et al. [92] proposed GridEye, a service-oriented monitoring system with flexible architecture that is further equipped with an algorithm for prediction of the overall resource performance characteristics. The authors discuss how resources are monitored with their approach in grid environment. However, as a shortcoming, they do not consider SLA management. Gunter et al. [108,200] presented NetLogger, which is a distributed monitoring system that can monitor and collect information of network. Unlike GridEye, NetLogger observes only network resources. Wood et al. [215] developoed a system named Sandpiper that automates the process of monitoring and detecting hotspots and remapping/reconfiguring virtual machines whenever necessary. Emeakaroha et al. [80–82] presented LoM2HiS, which is a monitoring framework for mapping low-level system information to certain high-level system performance metrics. Other similar tools have been proposed by [7, 12, 32, 120, 149, 177, 218]. However, all of these works work consider only infrastructures of grids and clouds (e.g., processor, memory, and network performance) and neglect their economic factors.

In information science and finance literature, the influence of internal system (i.e., market) structure and external factors on market quality has been excessively discussed. Examples of these studies are presented in [15] and [206]. However, considering the literature on this topic, market quality is usually defined in a very ambiguous way, often in terms of liquidity measures (e.g., [15, 19, 115]). Others also include information measures such as instance permanent price impacts and trade-correlated measures (e.g., [23, 111, 113]). In [224], the most prominent and important dimensions of market quality with a focus on quantifiably measures are identified and classified into three categories: activity [15, 115], liquidity [23, 121], and information [98]. Herewith, we focus on market liquidity since it is the considered to be the best measure for the attractiveness of a market as it indicates the ability to quickly trade large size of goods for low cost [109].

Although liquidity is commonly used in financial literature as one of the fundamental measures of market attractiveness, efficiency and activity, it is virtually impossible to give one definition of liquidity that covers all of its aspects and fits to all market scenarios. Due to this complexity, measuring market liquidity is far from trivial. However, there are several common measures for approximation of liquidity in market literature with spread measures (e.g., quoted, effective and realized spreads) [23, 121], market depth [15], and immediacy of matching [55] being the most prominent examples. Several other measures exist. For example, [223] proposed a new liquidity measure, `Illiq_Zero`, which incorporates both the trading frequency and the price impact dimensions of liquidity. Based on the transaction level data for 20 emerging markets from 1996 to 2007, the authors demonstrated that the new liquidity measure shows a high correlations with the liquidity benchmarks. [20] proposed using balance sheet and fund statements to measure liquidity, which is hardly implementable in electronic market platforms. [112] proposed some improvement measures based on the decomposition of transaction price time-series into a stationary and a random-walk component to address the shortcomings of the bid-ask spread as a measure of transaction costs identified in [107]. In the US government securities market, the yield spread between the on-the-run and the first off-the-run security is often used as a measure of liquidity [88, 100].

Overall, although existing liquidity measures are applicable to the markets trading financial assets, they cannot be directly applied to the computing resource markets due to the fact that the current computing resource market is built of an extremely high number of heterogeneous (i.e., quality differentiated) goods. The challenge is to adapt the common measures of liquidity so that they can capture liquidity of a set of differentiated, but partly substitutable goods. With such a definition, it will be possible to create a computing commodity market, which trades a limited number of standardized goods. A detailed discussion on common liquidity measures in financial markets and their possible application to computing resource markets is given in Chapter 4.

2.4 Discovery and selection of SLA-based services

Specifications of user requirements in clouds and grids have been discussed by several research projects [78, 104, 161]. As reported in [124], most projects use SLA specifications based on Web service level agreement (WSLA) and WS-Agreement, which lack support for economic attributes and negotiation protocols, as well as some non-functional properties, such as security. To compensate for these shortcomings, extensions to WS-Agreement has been proposed [173]. Oldham and Verma introduce the use of semantic Web technologies based on Web service description language (WSDL-S) and Web ontology language (OWL) for enhancing WS-Agreement specifications to achieve autonomic SLA matching [161]. Similar to that, Green introduces an ontology-based SLA formalization where OWL and semantic web rule language (SWRL) are chosen to express the ontologies [104]. Dobson and Sanchez-Macian suggest producing a unified quality of service (QoS) ontology, applicable to the main scenarios such as QoS-based Web services selection, QoS monitoring, and QoS adaptation [78]. Another approach, which has been presented in [8], introduces an autonomic grid architecture with mechanisms for dynamically reconfiguring service center infrastructures. It can be used to fulfill varying QoS requirements. Koller and Schubert discuss an autonomous QoS management, using a proxy-like approach for defining QoS parameters that a service has to maintain during its interaction with a specific customer [126]. Yarmolenko et al. make a case for increasing the expressiveness of SLAs [220]. This can possibly also increases market liquidity, if it comes to matching asks and bids and a common understanding of the SLA parameters has already been established. Our approach could be seen as complimentary in the sense that it makes sure that their precondition holds.

Most of the existing electronic marketplaces assume a unified specification of service requirements. In particular, they assume that all users in these market platforms use the same design languages and standards for defining their requirements. This simplification significantly helps in handling syntactic and semantic varieties in SLA templates. Currently, this is

Table 2.1: Comparison of the SLA mapping and other SLA management approaches

	Approach	**Examples**	**Strengths**	**Weaknesses**
1	SLA mapping	VieSLAf [30,31]	No explicit resource limitations; flexible SLA usage; user (market participant) driven; adaptation to changes in market	SLA translations are time intensive; system abuse by submitting fake SLA mappings
2	Standardized SLA templates for a specific consumer base	Amazon EC2 [4], Koller and Schubert [126]	Clearly defined resources; standardized SLAs are adaptable	Standardized SLAs may prevent users with unusual resource needs to join the market
3	Downloadable predefined provider-specific SLA templates	BREIN project [73], GEMSS project [18]	Clearly defined resources; standardized SLAs are adaptable	Plethora of SLA templates (worst case: each provider has their own template); trade-off between (possibly manually) utilizing a template and (possibly) manually integrating into the client software
4	Ontologies	Mukhopadhyay [154], Salama et al. [180], Oldhalm et al. [161]	Clearly defined resources; clear semantic definition of the meaning of one syntax description	There is no way to respond to consumers' needs in terms of adequate SLA templats; templates are generated in a static way; no variation in syntax allowed
5	Portal solution	Zimory [229]	Users agree on predefined requirements beforehand	Open for a limited client base only; trade-off between cost for enetring the portal and (possibly) short-term business

performed either by subscribing to a specific portal [229], by applying predefined ontologies [8,86,154,161,180], by downloading predefined SLA templates of a particular provider [18,73], or by using standardized SLA template defined for a specific user base [4,126]. However, applications of these approaches to resource markets in grid computing have demonstrated numerous disadvantages. All these approaches consider static definitions of SLA templates without a mechanism to reflect changes in the requirements of the consumer base. They do not provide the possibility to share the common understanding on what is needed in the market nor they provide flexibility in syntax for SLA specifications. Therefore, all of these methods could be hardly used in real-world trading scenarios. As it will be described in Chapter 4, we utilize the SLA mapping approach [28]. In this approach, syntax differences between semantically equal parameters of SLA templates are bridged by defining translations between differing properties of SLA parameters using Extensible Stylesheet Language Transformations (XSLT) documents named SLA mappings. Table 2.1 summarizes the strengths and weaknesses of all these SLA management techniques for handling SLA-based contracts.

Resource allocation, provisioning, QoS-based service selection, and negotiation in grids and clouds have been the subject of many research studies. Various resource-efficient and economically beneficent allocation techniques and methodologies have already been proposed to address these issues. These methodologies include game theoretical approaches [9,209], stochastic programming [54], bio-inspired mechanisms [61,187], auction-based algorithms [114,167,222] and agent-based approaches [65,186,199]. Most of these works perform QoS service selection and resource allocation based on some service performance indicators and economic indicators, including wait time optimization, utilization maximization and economic wastage minimization. However, none of the works identified the importance of market liquidity nor performed any kind of adaptation of resources traded in cloud markets.

Several works deal with autonomic QoS matching in various distributed environments (e.g., [63, 87, 128, 155, 202]). Similarly to our approach, these works also focus on matching requirements of different SLAs. In contrast to our work, however, they try to facilitate the matching process by introducing ontologies as enhancement to plain SLA documents, thus forcing market participants to specify the semantics of their requirements in such ontologies. In the broader field of computer science, there are many other approaches for automatic matching of various types of entities. Recent works include matching records of databases (e.g., [21, 22, 79, 96, 97]) and ontologies for semantic web (e.g., [77]). These works, however, significantly differ from our context.

Approaches for automatic discovery of mappings between various types of entities have also been discussed in other fields of computer science. [25], for instance, provide an approach for automatic search for mapping rules for XML Schemas. [27] describe an approach for automatic derivation of XSLT transformations between XML documents. However, these approaches either try to match similar entities using traditional similarity algorithms that only operate on a syntactical level, build upon explicitly described semantics of the analyzed entities, or use synonym databases to find semantic relationships between individual words. On the contrary, the approach proposed in this thesis focuses on the whole structure of SLAs rather than on individual entities. Moreover, our approach utilizes machine learning methods to learn from past experiences, thus continuously enhancing the knowledge about relationships between semantically equal SLA elements.

To facilitate automatic reasoning about the equality of SLA elements and creation of mappings between them, we propose strategies for learning from already established SLAs as well as from market participants' feedback in response to automatic recommendations. Learning methods for similar research problems have been evaluated in various fields of computer science. The ones most similar to our context deal with matching similar records in databases by utilizing classification methods (e.g., [21, 22, 62, 127, 227]). Most of these works use multiple classifiers allowing them to automatically adapt to the characteristics of record instances. Other learning methods deal with learning from past experiences. Many works use Case-Based Reasoning (CBR) for this purpose. For example, [146] used CBR to determine resource configurations of virtual machines in cloud environments. However, this and other similar works cannot be directly applied to our research problem since the input features used for classification as well as the models of cases utilized by CBR may be highly different to our approach. [76] give an overview about different methods for yielding best results with CBR, e.g., they discuss different methods to measure the similarity between cases and to retrieve best matching cases.

2.5 Experimental market platforms

Simulation of electronic markets for grid and cloud computing has been discussed in several large research projects, including SORMA [192], GridEcon [174] and 4CaaSt [151]. [192] developed a market simulator to compare centralized and decentralized service allocation mechanisms in market scenarios according to a defined set of metrics. In their work, they considered complex interdependencies that are broken down into two interrelated markets, namely a service market, which involves trading of application services, and a resource market, which involves trading of computational and infrastructure resources such as processors, memory, etc. [174] present the GridEcon platform - a testbed for designing and evaluating economics-aware services in a commercial cloud computing setting. The authors assume the difficulties in predicting the context of a service market and motivate development of an environment for evaluating its behavior in an emulated market platform. The platform is composed of the Marketplace, which allows trading goods using different market mechanisms, and the Workflow Engine, which enables a simple composition of a market environment by describing the service interactions between economics-aware services. [151] discuss a mechanism for the resolution of the customers' requirements that enhances the process of selecting cloud services from the business point of view. The work is related to the 4CaaSt project and aims to create a PaaS cloud platform that supports the optimized and elastic hosting of Internet-scale multi-tier applications.

[75] discuss a framework for modeling and simulating service-oriented applications and autonomic policies for service provisioning and resource orchestration for Application Layer Networks in utility computing environments. The approach is evaluated within CATNETS project and investigates the use of an economic model (Catallaxy) in distributed environments like grids and P2P networks. [194] discuss the design of a simulator with a set of features for simulation of grid testbeds as an extension to GridSim. They model heterogeneous computational resources of variable performance, scheduling of jobs based on various policies, differentiated network service, and workload trace-based simulation. More importantly, the framework has the ability to handle replication of data to several sites, query for location of the replicated data, access to the replicated data, and making complex queries about data attributes.

Although many of these market simulators successfully address some of the main challenges of electronic markets in distributed environments, they are fairly static and do not have any autonomic capabilities. Therefore to orchestrate and evaluate autonomic markets, a more flexible simulation approach is necessary.

CHAPTER 3

Self-awareness in market platforms

Today, electronic marketplaces are challenged by a highly dynamic context: high product variability, unpredictable participant behavior, and the emergence of new actors as well as actor types. Consequently, market situations that have previously been unimaginable will arise and novel theories and paradigms are needed to facilitate and control them. Examples of new market contexts are in the domains of smart grids, the Internet of services and cloud computing as well as social and collaborative environments. Many of these domains are already or will become inherently reliant upon the economic systems represented by electronic markets that can address their allocation problems.

However, these domains are dynamic and unpredictable, impeding the possibility of creating a fitting electronic market platform that would be "adequate" in all scenarios. Today's electronic markets are static and are not sufficient to completely capture such a marketplace. We believe that marketplaces need to be capable of reacting to the paradigm's (and its followers') highly dynamic context in order to ensure market stability and sustainability.

In this chapter, we address Research question 1 specified in Chapter 1 and define our vision of autonomic market platforms - economic systems that address the elements of dynamism and uncertainty by applying self-* properties to autonomically adapt their configuration at runtime and satisfy a predefined set of market goals. Through methodology commonly known from the Autonomic Computing paradigm, we create market platforms that are able of evolving beyond their initial design principals and evolve towards a better, stable configuration(s). We present this discussion in Section 3.1.

In this chapter, we additionally address Research question 2 specified in Chapter 1 and introduce our first steps towards the vision of autonomic cloud market: a novel methodology for the monitoring of cloud markets (Section 3.2). A large part of this methodology is the identification of monitoring data that is available and useful for autonomic markets, and the necessary mappings to transform these metrics into indicators for a given set of market goals. To demonstrate our approach, we utilize GridSim [43, 45] as a tool for the simulation of grid and cloud market behavior. We have extended GridSim with appropriate market and mechanism sensors as well as simple infrastructure sensors (Section 3.2.4). Based upon the monitoring metrics of

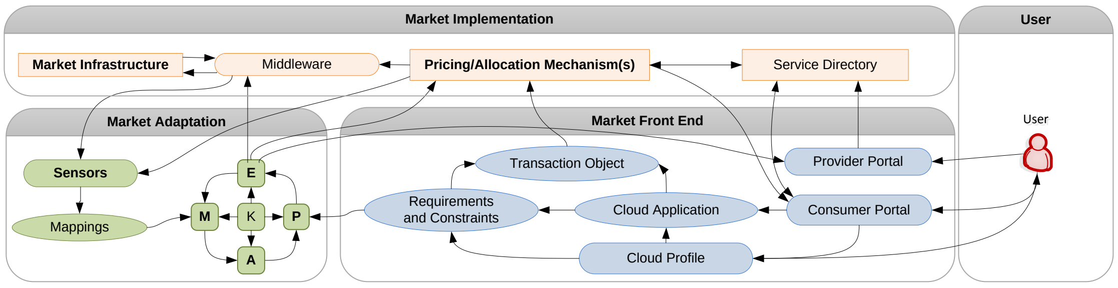


Figure 3.1: Self-aware cloud market

the market (which are translated from the low-level infrastructure measurement) our monitoring model can sense dynamic changes in market behavior, which is the first step towards establishing self-aware autonomic market platforms.

3.1 A self-aware market platform

Cloud computing promises a fast, inexpensive, and simple way to match consumers and providers. However, it also suffers from numerous challenges, such as: a large market diversity (resulting in liquidity risks), a fragmented and diverse array of service providers, and few standard mechanisms for unilateral provider adoption and use. In this thesis, we address these issues by enabling the marketplaces to *autonomically* dynamically adapt to the current need of market participants and to address the impacts these constraints have upon the market itself.

In our vision of an autonomic cloud marketplace, a market platform has the ability to change, adapt or even redesign its anatomy and/or the underpinning infrastructure during runtime in order to improve its performance. This can be done through autonomically applying horizontal or vertical scaling to the underlying computing infrastructure in response to the available resources. Similarly, it could tune or change the market mechanisms or its components in use during the trading process. In our vision, cloud services (e.g., software and hardware infrastructures) regardless of their provider are traded via electronic cloud markets. In such environments, four key independent components are needed. First, users (often represented by their agents) authenticate themselves and place bids for a certain service (as consumers) or define offers (*asks*) for services (as providers). Second, an *allocation mechanism* matches placed bids and asks, while a *pricing mechanism* determines price and quantity of a product to be traded between a consumer and a provider. Third, a market front-end, which may be similar to existing multi-provider dashboards (e.g., Rightscale[1] and IBM's Altocumulus[2]), allows users to access the marketplace. Finally, an autonomic adaptation component enables the market platform to modify itself.

Figure 3.1 illustrates the basic anatomy of an autonomic market platform, highlighting information flows, dependencies between components, and the inclusion of the MAPE-K cycle (Monitoring, Analysis, Planning, Execution, and Knowledge). We note that all components

[1]http://www.rightscale.com, Last accessed: May 2013
[2]http://almaden.ibm.com/asr/projects/cloud/, Last accessed: May 2013

shown in Figure 3.1, except the *Market Adaptation* component, have to various degrees been investigated in past and on-going projects such as SORMA,[3] GridEcon,[4] CloudBus[5], Social Clouds [57], Optimis[6] and 4caast[7].

In this figure, there are two primary potential adaptation points: (1) the infrastructure, and (2) the market configuration. Infrastructure here refers to the computational infrastructure of the market platform that enables its core functionality, i.e., computational resources, delivery mechanisms, communication channels, security procedures, etc. Adaptation in this context is what we commonly understand as an elastic infrastructure in the cloud paradigm. The market model, or to be more specific its institutional form, relates to the basic components of the market, e.g., rules of participation, allocation and pricing mechanisms as well as their parameters, tradable artifacts, market type (such as monopoly, duopoly, polyopoly, oligopoly, etc.), and market goals (such as liquidity, immediacy, stability, security, participant welfare, participation, energy efficiency, allocation efficiency, etc.). A given institutional form, i.e., an instantiated parameterization of these components, is what we refer to as a market configuration. An adaptation at the institutional layer, therefore, means a change in one or more parameter settings and hence a change in the market configuration. In following sections, we identify how the autonomic MAPE-K cycle can be leveraged for our vision of market adaptation, i.e., both its infrastructure and configuration.

3.1.1 Monitoring

Monitoring data is critical for the instrumentation of any form of adaptation. In Section 3.2, we focus on defining a monitoring methodology for an autonomic marketplace and demonstrate how the performance of a market platform can be measured with respect to a specific set of market goals. This task is performed by monitoring sensors, which gather low-level monitoring data from the market middleware and implementation of the market configuration. For example, monitoring sensors can be placed at different layers of the market platform and monitor the market performance from both the infrastructure (usage of computational resources) and institutional perspectives.

Market goals can be rather abstract concepts and cannot be directly derived from monitored data. Therefore, in order to assess a market's performance, low-level data is mapped into performance scores for the high-level (business or market) goals. Mappings are therefore used to combine and transform monitored data into indicators that determine each goal's performance, and therefore the performance of the market as a whole.

3.1.2 Knowledge

In the context of an autonomic marketplace, we can differentiate "knowledge" into several categories depending on specific states of an autonomic marketplace, as well as the current position

[3]http://www.sorma-project.eu, Last accessed: May 2013
[4]http://www.gridecon.eu, Last accessed: May 2013
[5]http://www.buyya.com/CloudBus, Last accessed: May 2013
[6]http://www.optimis-project.eu, Last accessed: May 2013
[7]http://www.4caast.eu, Last accessed: May 2013

in the MAPE cycle. Through this differentiation, it is clear that different categories of knowledge play key roles at different stages in the autonomic management of a market platform. These categories are as follows:

Empirical: empirically derived facts or observations, e.g., monitoring data; observed experiences/payoffs from previous adaptations; previous (adaptation history), current and planned (future) configuration(s); infrastructure status; and participant information.

Contextual: instance-specific knowledge, e.g., active marketplace goals; acceptable deviations from goals; goal priorities; goal/metric mappings; business model(s); initial/desired configuration(s); and adaptation ranking mechanism(s).

Institutional: knowledge concerning the economic anatomy of a market, e.g., valid alternative configurations; adaptation paths between configurations; parameter boundaries; market rules, constraints and regulations.

It is clear that empirical knowledge is gathered through monitoring and logging techniques, and therefore requires no further elaboration. This, however, is not the case for the other two categories. Contextual knowledge is initially set by a platform provider (i.e., the actor or legal entity that claims ownership of the marketplace) or its administration. It is not anticipated that an autonomic manager adapts or changes an entity within this category. Instead, these are the benchmarks and foundational knowledge entities for an autonomic manager that act as points of reference. Institutional knowledge is defined partially by the platform provider and partially established based upon contextual knowledge. For example, a platform provider may set and change specific parameter boundaries, market rules and regulations to control adaptation processes and define key aspects of the market which cannot change. We expect that potential configurations can be discovered automatically by considering: the initial market configuration, parameter boundaries, active goals, as well as the market rules and regulations. It is assumed that institutional knowledge will change and evolve with the marketplace.

As knowledge plays a vital role in many parts of our approach, it is critical that adequate knowledge management techniques are employed. In our context, knowledge management means the intelligent use of measured data, obtained by monitoring, as well as the leverage of existing knowledge for decision making processes. Figure 3.2 presents an overview of how knowledge management should be used to synchronize the remaining phases of the MAPE cycle. First, the monitoring phase delivers monitored information (empirical knowledge) that captures the current state of the market platform. Second, the *analysis* phase processes the raw monitored data in order to determine the current performance of the market in accordance to the currently active goals leveraging contextual knowledge. It then will use institutional knowledge to identify possible adaptation options if required. Then planning phase plans the execution of the recommended actions by exploring and selecting an adaptation path. It also inspects any previous adaptations in an attempt to prevent or minimize oscillation effects. The execute phase is the final one. It instruments the recommended actions within the market platform devices with the help of software actuators.

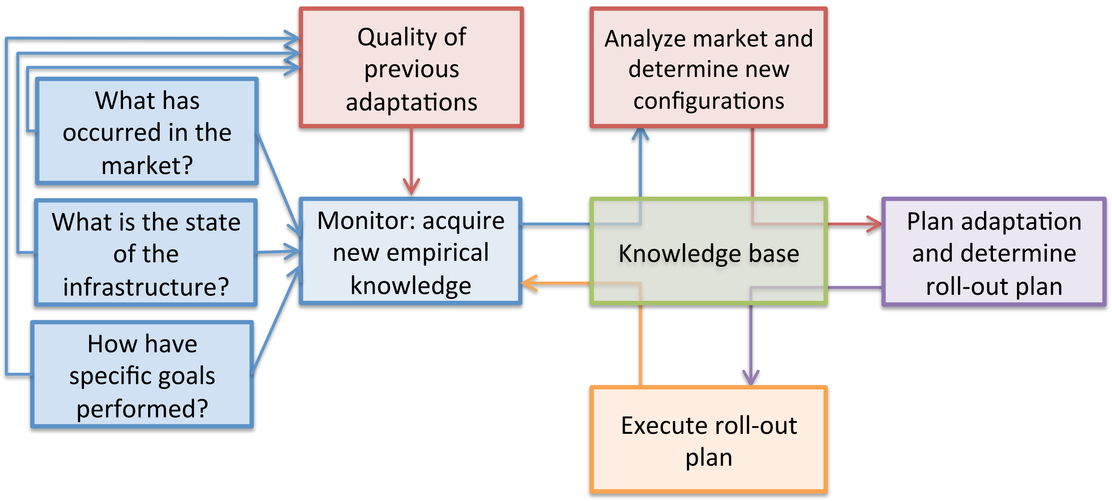

Quality of previous adaptations
Analyze market and determine new configurations
What has occurred in the market?
What is the state of the infrastructure?
How have specific goals performed?
Monitor: acquire new empirical knowledge
Knowledge base
Plan adaptation and determine roll-out plan
Execute roll-out plan

Figure 3.2: Knowledge management overview

3.1.3 Analysis

The analysis phase of the autonomic cycle is in the context of an autonomic marketplace by far the most challenging. In essence, it analyses mapped data from the monitoring phase to assess market performance, i.e., it uses empirical knowledge to assess market performance as defined by, and using, contextual knowledge entities. In the event that no unacceptable deviations in the goal set are identified, the MAPE cycle terminates. If this is not the case, an adaptation strategy needs to be devised. As mentioned above, there are two main adaptation options: market infrastructure (scaling in/out the market infrastructure, for example, by adding/removing VMs to change computational performance), or modifying the market's configuration. It will not always be trivial to decide which of these options is most fitting, or "best". For example, consider the simple case that an insufficient number of transactions are being performed per unit of time. This could be remedied in several ways, for example by: 1) scaling compute nodes up or the infrastructure as a whole out, 2) tuning the matching algorithm to reduce compute time (e.g., applying a heuristic instead of an optimal algorithm), 3) purging the order book(s) of redundant data (data reduction), or 4) tuning allocation mechanism properties (e.g., the maximum number of entries in the order book, the clearing or pricing functions). Of course, combinations of these options are also valid, as well as more aggressive adaptations like changing the allocation mechanism for another. Note that besides adapting the underlying infrastructure and configuration, it is possible to slightly alter the market environment. One such adaptation (management of resources traded in the market) will be discussed in Chapter 4.

To consider infrastructure changes is not too difficult to achieve through simulation and a large body of literature on performing or simulating infrastructure-based adaptation exists (e.g., [1, 49, 53]). This, however, is not the case for institutional adaptation. Most approaches that perform any form of institutional adaptation are "trivial" adaptation processes and typically rely on complicated economic models to facilitate decision making. Therefore, to facilitate institutional adaptation, we need to understand what different market configurations mean for the fulfillment of a given set of goals. For an arbitrary set of configurations, this is a not a trivial undertaking. However, for specific market configurations this can be achieved through simulation to enable the analysis of what-if scenarios to determine and assess adaptation options. To support such a simulation, we require rich models that capture the nuances of a market configuration. These models need to specify the: parameter settings, inter-parameter dependencies, potential control and adaptation points, and the parameters' relationship to the market context (i.e., artifacts/services traded).

The existence of different adaptation options as well as their possible combination is the primary source of complexity for the analysis phase. In the example, only one goal is affected (transaction throughput). However, if additional goal violations were observed, an increased number of adaptation options may also arise resulting in an increased computation effort to identify a "good" adaptation option. Therefore, an analysis component must access different adaptation options with regard to their expected improvement in the fulfillment of one or more goals, rank these options, and then determine the most suitable infrastructure adaptation and/or new market configuration. When multiple goals are under consideration, this may lead to (partial) impasses, and require additional contextual knowledge (e.g., goal priority) to make decisions. Therefore, ranking different adaptation options considering multiple attributes is essential, e.g.: the ease of orchestrating the adaptation; cost(s) of adaptation; expected improvement; whether other goals are positively or negatively affected by the adaptation, i.e., the pareto optimality of a solution; and level(s) of uncertainty.

3.1.4 Planning

Planning, in the context of autonomic adaptation, relates to two key endeavors. Firstly, the identification of the most suitable adaptation path to instrument infrastructure changes or a new configuration by leveraging contextual knowledge. Secondly, as an adaptation path may include more than one market component or steps, it is necessary to determine the order and timing of the adaptations to be instrumented. This may result in multiple rounds of the MAPE cycle, to observe how single changes have had an impact upon market performance, and ultimately lead to an iterative adaptation process. Here, deviations to the plan upon successive cycles can be expected, as a consequence of new empirical knowledge.

3.1.5 Execution

The execution phase is the instrumentation of an adaptation path. In the case of an infrastructure adaptation, this relates to an interaction with the resource fabrics through the platform middleware. For institutional adaptation, it refers to a check point of the current market, a new parameterization of the market configuration, and redeployment (if necessary) of effected market components.

3.2 The monitoring model

As already explained in Section 3.1, building a monitoring model is the first step towards the vision of an autonomic market platform - it enables the market platform to be self-aware, i.e., knowledgeable about its state at multiple levels. To achieve this goal, a monitoring methodology must be able to monitor the market performance from both the institutional and infrastructural perspectives.

The main motivation for building a monitoring methodology is the ability to continuously monitor market characteristics for the identification of potential inefficiencies in electronic marketplaces. Using monitoring data, the market platform should autonomically adapt its properties with the goal of acting against the inefficiencies detected in the market. In particular, the market platform would use the monitoring methodology to collect the values of a set of *monitoring metrics*. Using these metrics, the market platform defines a set of *market goals*, i.e., acceptable (e.g., threshold) values of the metrics. In case a monitored value exceeds the given threshold, a market goal is not satisfied and an adaptation cycle is triggered. However, monitoring metrics are usually high-level (business or market) goals that cannot be directly measured in the market platform. Instead, they must be evaluated against low-level monitoring data.

In this section, we focus on three aspects. Firstly, we identify the information that can be monitored and used to determine the performance of the market. Secondly, we identify a set of common high-level market metrics that are relevant to our scenario. Thirdly, as market metrics are generally quite abstract, much more than the lower-level monitoring data, we present mappings (i.e., appropriate aggregation functions) from the low-level measures to the high-level market metrics.

3.2.1 Retrieving information from the monitoring data

Low-level monitoring units, i.e., directly measurable information in the market, can be gathered at three levels:

1. **Market level**, i.e., information related to the market platform in general (e.g., number of active market participants in a given time interval),

2. **Mechanism level**, i.e., information related to the efficiency of a market mechanism (e.g., average matching price agreed between buyers and sellers), and

3. **Infrastructure level**, i.e., information related to the performance of the underlying operating system, hardware and software (e.g., CPU usage).

The quantity and type of data that can be observed in the market platform is vast. However, not all of the data is relevant for steering an autonomic market platform. Table 3.1 presents an initial set of the most important monitoring units that are easily observable in a market platform. As it can be seen, most of the monitoring units identified concern either users' activity in the market (e.g., number of active users, number and prices of their asks and bids, desired quantity of resources, etc.) or the economic performance of a market mechanism (e.g., quantity and prices of matches between asks and bids in a certain time period). Note that the market and infrastructure level data can be gathered in all trading scenarios, while some of the mechanism related information (e.g., size of the order books) may depend on the choice and implementation of an allocation mechanism.

Table 3.1: List of low-level monitoring units

Level	**Unit**
Market	Number of resource types traded in the market
Market	Number of (active) traders in a time period
Market	Participation costs in the market
Mechanism	Average clearing price
Mechanism	Number of allocated (i.e., matched) resources
Mechanism	Average price of bids and asks
Mechanism	Number of arriving bids and asks
Mechanism	Size of the order book(s)
Infrastructure	Time to compute an allocation and pricing
Infrastructure	Resource usage (e.g., CPU utilization)
Infrastructure	Costs of platform resources

Since the only purpose of collecting monitoring data is their aggregation in the more abstract market metrics, their choice and benefits will be discussed in the next section, where we present the market metrics.

3.2.2 Market metrics

A market can have several goals that it aims to achieve, e.g., concerning the market environment (i.e., what type of goods are traded, who owns the market, etc.) and the target group (system, consumer or provider). From the economic point of view, market goals are often abstract system-wide goals that depend on the market mechanism used. For example, maximization of *welfare* (the sum of consumer and provider surplus) is one of the most desirable and applied goals for markets. However, the downside of this particular metric is its dependence on the concept of providers' and consumers' utilities, which are difficult to capture universally. This means that not all possible market metrics can be monitored in a real environment. However, several useful market metrics exist that require derivable low-level monitoring measures and do not rely on abstract (economic) concepts: (provider and market) revenue, platform profit, number of allocations, transaction volume, platform execution cost(s), liquidity, number of active traders, etc.

Revenue and *platform profit* directly measure the attractiveness of the market to a specific group of users (providers and platform owners). Revenue is a good example of how the monitoring metrics represent a compromise between the usefulness of a metric and the availability of the low-level monitoring measures. Namely, since not all information can be directly measured in the market (i.e., internal costs of market participants), more interesting monitoring metrics such as providers' profit cannot be assessed. Therefore, instead of considering the profit, the best possible estimation is the revenue.

The *number of allocations* (i.e., matches between sellers' asks and buyers' bids), *number of active traders* (i.e., users that are actively participating in the market by sending bids and asks in a given time period) and *transaction volume* (i.e., the amount of resource traded in the market in a given time period) consider the impact of the market with respect to trades and can be important if, for example, market fees are charged per transaction (volume). *Platform execution costs* are directly relevant to the market platform owner, as they want to minimize the resources used to run the market. As some market mechanisms involve extensive computation (e.g., some auction types involve NP-hard calculations), this is an important consideration and represents a trade-off between efficiency and costs.

Probably the most interesting market metric is the measure of *market liquidity*. Liquidity is an important measure of market quality and a concept which is commonly used in financial markets. It describes how easy it is to trade a certain volume of the considered good. In more detail, liquidity is an asset's ability to be sold without causing a significant movement in the price and with minimum loss of value. Its rather abstract definition means that there is no single aggregate value for liquidity. Instead, there exist several standard measures that serve as a proxy for its assessment, out of which we herewith discuss three.

1. **Bid-ask spread** measures the difference between bid and ask prices of a good [6]:

$$BAS = \frac{avg.\,price\ bids - avg.\,price\ asks}{time\ interval} \tag{3.1}$$

2. **Market depth** considers the volumes of goods that can be traded at a certain point in time [15]:

$$V = \frac{amount\ of\ goods\ traded}{time\ interval} \tag{3.2}$$

3. **Immediacy of matching** determines the time needed for an order to be (successfully) matched [55]:

$$I = time_{trade\ executed} - time_{ask\ submitted} \tag{3.3}$$

A detailed discussion of these measures as well as importance of market liquidity and other methods for its estimation in cloud markets is given in Section 4.6.

Note that as with the low-level monitoring units, the presented set of monitoring metrics is not complete, but rather an initial set of metrics that are the most important for the assessment of cloud market performance. Other monitoring metrics (such as volatility and energy efficiency) will be explored in our future work.

3.2.3 Mapping monitored data to market metrics

As some market metrics presented in the previous section are rather abstract concepts, they cannot be directly derived from monitored data. Therefore, it is necessary to define not only units that can be monitored during the execution of the market, but also the mappings that combine and

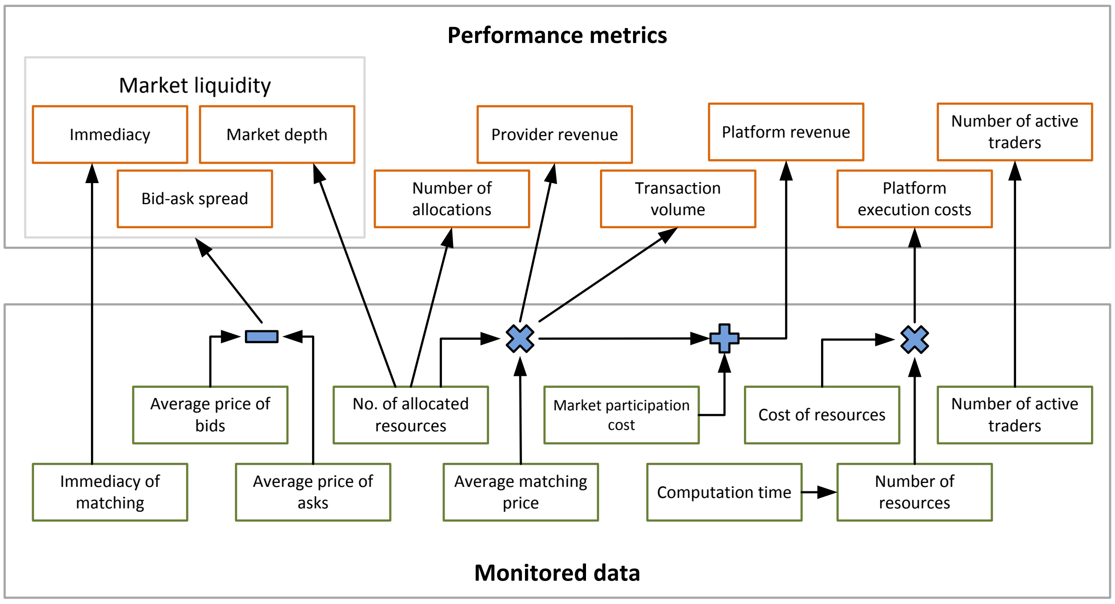


Figure 3.3: Mapping the monitored data to the market metrics

transform these units into indicators (i.e., metrics) that determine each goal's performance. Some monitoring units map directly to the metrics described previously (e.g., number of allocations), but for all other goals multiple metrics have to be considered. For example, the bid-ask spread is computed as the difference between two monitoring units: average of sellers' asks and average of buyers' bids, divided by the length of the monitoring interval. Platform revenue includes both the trading revenue of the total volume of all goods and the cost of market participation and is, therefore, defined as combination of three units: number of allocated resources multiplied by the average matching price and increased for the value of participation costs. Figure 3.3 presents these and other mappings implemented as a part of this thesis to demonstrate the practicability of our monitoring model.

3.2.4 Implementation of the monitoring model: an extension to GridSim

For the evaluation of the monitoring methodology presented in the previous section, we implemented monitoring sensors as extensions to GridSim, an open-source toolkit for conducting simulations in grid environments. Although GridSim simulates grid resource and network and does not consider the cloud computing paradigm directly, the choice of GridSim as the underlying architecture is adequate for several reasons. First, GridSim implements numerous reservation-based and auction mechanisms for resource allocations, including the double, English, Dutch, first-price sealed-bid, and continuous double auction mechanisms [10]. It also provides well-defined interfaces for the implementation of the additional mechanisms and algorithms. Second, it is designed as an extensible multi-layer architecture which allows new components or layers to be easily added and integrated into the framework [41]. It also provides a comprehensive facility for creating different classes of heterogeneous resources that can be aggregated using

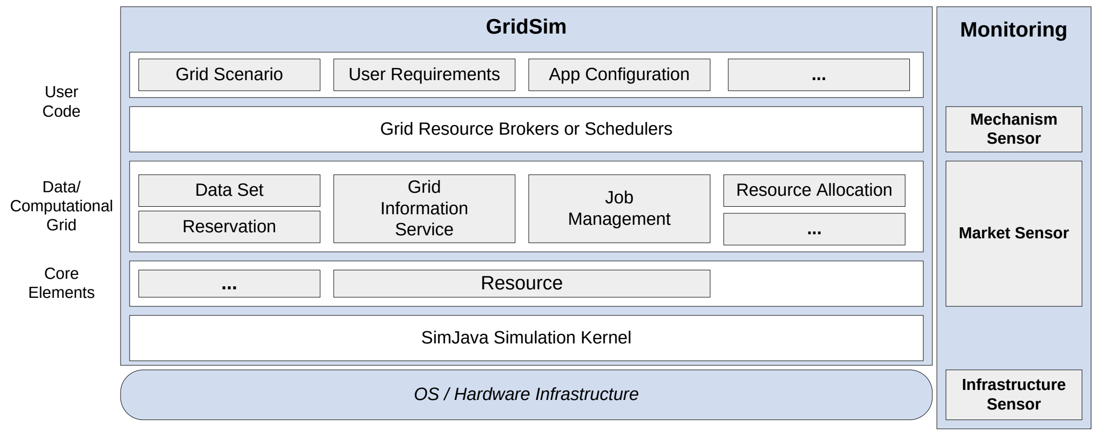


Figure 3.4: The layered architecture of GridSim and its components

application schedulers, also called the resource brokers, for solving compute and data intensive applications. Finally, as an open-source toolkit it has already been used for several research projects [43].

Regarding the difference between grid and cloud paradigms, it is important to note that resource markets for trading cloud and grid services do not significantly differ. In particular, although definitions of resources may not be the same, the techniques for matching buyers' bids and sellers' asks are equal. Therefore, although GridSim simulates grid infrastructures, it is still suitable for the implementation of the simulation of a cloud market. Moreover, due to the short history of clouds, most of the popular cloud simulation toolkits, such as CloudSim, do not yet capture market mechanisms [49], making GridSim a better choice for simulating cloud markets at this point in time.

The monitoring tool measures market and infrastructure performance of GridSim components placed in one or more architecture layers. Figure 3.4 presents the layered architecture of GridSim, its components and our extensions. As depicted, GridSim is placed on top of an operating system and a distributed hardware infrastructure. Communication between the infrastructure and other GridSim components is managed by the bottom GridSim layer. This layer additionally handles the interaction or events among GridSim components themselves. It is based on SimJava [118], a general purpose discrete-event simulation package implemented in Java that defines message passing operations, for communication between GridSim components [41]. The second layer models the core elements of the distributed infrastructure, such as resources (e.g., storage) that are essential to create simulations in GridSim. The third layer is concerned with modeling and simulation of services providing various functions such as managing job submission and providing information about available resources. The fourth layer contains components that aid users in implementing schedulers and resource brokers in order to test algorithms and strategies. Finally, the top layer helps users to define scenarios and configurations for validating the algorithms.

Our monitoring tool consists of three sensors: a mechanism sensor, a market sensor, and an infrastructure sensor. Following the layered architecture of GridSim, each sensor extends one or more GridSim layers to monitor market performance, as depicted in Figure 3.4.

Mechanism sensor monitors performance of a market mechanism from the economic perspective. For example, it monitors the number of arriving bids and asks, as well as the number of matches and the average price of a match between bids and asks in a time period. As already described, the allocation and pricing algorithms are handled by the GridSim's fourth architecture layer. Therefore, the mechanism sensor is placed on top of this layer where it uses the available event-based interfaces to receive notifications and properties of mechanism allocations and clearing prices. Once a match between a buyer's bid and a seller's ask is computed, the mechanism sensor receives and stores the necessary information in the knowledge component. Besides gathering the information on allocations, the mechanism sensor gathers the other mechanism-related information, such as the size of the order book(s), the number of the resources awaiting an allocation, etc.

Market sensor gathers information related to the market platform in general. Along with the mechanism sensor, it monitors the institutional performance of the market. The information it captures is not mechanism-related, but is important for assessing market performance with respect to the market metrics. For example, the market sensor stores and analyzes data related to the past and current number of active sellers and buyers, as well as the information about the resources traded between them. This is achieved by using GridSim interfaces of the architecture layers responsible for resource and job management (i.e., layers 2 and 3 in Figure 3.4).

Infrastructure sensor measures the performance of the market platform with respect to the usage of computational resources. In particular, the infrastructure sensor monitors the utilization and performance of the underlying operating system and hardware infrastructure, such as CPU utilization and speed, number of threads, heap memory and hard-disk usage, etc. The infrastructure layer monitoring is based on the interfaces of the `java.lang.Management`, which is a common management interface for monitoring and management of the Java virtual machine and the host operating system.

CHAPTER 4

Adaptable cloud resources

Today's "traditional" electronic marketplaces support the trading of differentiated services: a buyer's requirement is compared to all sellers' offerings to find the best matching service. This process is often inefficient due to the market dynamism and a large diversity in resources, and requires buyers and sellers to invest a large effort to find the best matching service offerings. Additionally, this extreme market differentiation causes low market liquidity. In this chapter, we address this problem (also specified as Research question 3 in Chapter 1) through the *standardization* of computational resources.

Similar to the existing approaches in other types of commodity markets, we consider standardization as a process of channeling demand and supply into a limited number of "standardized products". However, unlike in traditional commodity markets, standardized products (as well as demand and supply) are herewith continuously monitored and adapted in order to always reflect current market trends. Note that the creation and adaptation of standardized products is a form of autonomic market adaptation, as described in Chapter 3. Nevertheless, unlike other common adaptation actions introduced before, standardization performs adaptation to the market environment, and not the market infrastructure and configuration.

The remainder of this chapter is organized as follows. In Section 4.1, we demonstrate the liquidity risk(s) in cloud markets. As a means to address this issue, in Section 4.2 we introduce our approach of standardization of SLA-based service specifications. In Section 4.3, we discuss and formally define SLA-based resources traded in cloud markets. In Section 4.4, we discuss design of a market platform capable of trading standardized services. This discussion is continued in Section 4.5 where we present methods and algorithms for achieving this vision. As ensuring liquidity of electronic marketplaces assumes that it is possible to monitor and assess liquidity in these environments, in Section 4.6 we introduce methods for estimating liquidity in cloud markets based on the discussion introduced in Section 3.2. Furthermore, we present utility and cost models for measuring benefits and costs of our approach.

4.1 Liquidity problem in utility and cloud computing markets

A cloud market (and an electronic market in general) is a platform where demand and supply for certain (computational) goods meet in order to: (1) offer goods/services in a structured manner, which includes service selection and discovery; (2) negotiate the price, service requirements and conditions under which the services are sold, purchased or rented; (3) set up legally binding electronic contracts (e.g., service level agreements) detailing the trading conditions and party obligations; and (4) pay and deliver selected services in a manner specified by the derived contracts [95, 136]. In the case of utility and cloud computing paradigms, the goods that are to be traded in the electronic market are computing service units characterized by their scalability and elasticity [33, 35]. These service units provide a programming interface (API) that enables machines (and the market) to interact with the service and are encapsulated within a virtualization layer that enables sharing physical infrastructure (e.g., storage) and migrating applications from one physical server to another.

Virtualization allows simplified integration of computational resources into companies' IT infrastructures. If a cloud provider uses the same virtualization tool as a consumer, server images can easily be moved between the provider's and the customer's resources, thereby reducing the cost of moving the data and the application(s) to the cloud. For an open cloud computing market, however, virtualization introduces the problem of an endless availability of different resource types that are made available in the market. In essence, any provider can define their own resource types. This is often done to differentiate from market competitors. Namely, in a market with differentiated goods providers differentiate themselves from each other either by changing the pricing scheme or by changing the good itself. Changing the price is simple, but can be copied by other providers easily. Changing the good, on the other hand, does not have this disadvantage. In computing resource markets, which make use of virtualization technology, this can easily be accomplished by changing the attributes of the virtual machines (VMs) that are traded. In general, providers can change any of the attributes of a VM: number of CPU cores, CPU speed, main memory size, hard disk space, bandwidth, quality of service (QoS) guarantees, as well as the quantity of active VMs. While some values of these attributes are limited to a small range (e.g., the CPU speed is limited to the maximum speed of the physical device), other values can be almost arbitrarily chosen.

The disadvantage of this differentiated goods market lies in the large amount of different resource types in the market, causing two liquidity problems for service providers and consumers:

1. the quantity of available resources of a certain type is often small, thus decreasing the likelihood of matching demand and supply (i.e., the matching probability); and

2. the cost of finding an appropriate service provider or a consumer is very high. Instead of comparing a few offers, a consumer is now facing a problem of having to compare all offers to find the offer that closely matches his needs.

As already discussed in Chapter 3, market liquidity is an important measure of market quality and a concept which is commonly used in financial markets, but can be applied to other types of markets as well. In its essence, it describes how easy it is to trade a certain volume of the considered good. A market is liquid when it has a high level of trading activity, where one can buy and sell with the minimum price deviation.

The essential characteristic of a liquid market is that there are sufficiently many ready and willing buyers and sellers at all times. Market liquidity also depends on the ease with which market participants can carry out transactions. Thus, other things being equal, lower transaction costs contribute to higher market liquidity. In particular, if transaction costs and the costs of the participation in the market are high, the gap between the effective price received by the seller and that paid by the buyer of a service will be large and it will be difficult to match sell and buy orders [72]. Furthermore, if participation costs are high enough to constitute an entry barrier, the market will attract fewer dealers and investors, also lowering trading activity and, consequentially, market liquidity.

In order to work efficiently and to guarantee market stability, a marketplace should have a sufficiently high liquidity. In order for a market to be deeply liquid, a quick, simple and inexpensive exchange of products between buyers and sellers needs to be possible. In markets with a high variety of resource types, as it is the case with cloud markets, this means that it is necessary to ensure a large likelihood of finding a seller's offering for every buyer's requirement and vice versa.

4.2 Improving market liquidity through standardization of SLA-based cloud services

The means to address the liquidity issue in utility and cloud markets is in our work through channeling demand and supply into a limited number of automatically defined computational resources named standardized resources. This approach is based on two important assumptions:

1. Reducing the number of resources has the effect of homogenizing the resource market and, therefore, reduces the cost of finding a requested service in the market (i.e., the search cost).

2. Computational resources are mostly fungible, i.e., they are capable of mutual substitution. Resource fungibility is an asymmetric relation and, although it dictates that a resource is substitutable by another resource, a user's satisfaction of utilizing the other resource does not have to be the same.

In differentiated goods markets, tradable goods are those that are offered by service providers. For a service consumer it is, naturally, not always possible to find an exact service that is required. However, as computational services are mostly fungible, consumers can choose from many other services that are similar to the one that is required. The process of service standardization supports the consumer in that: It selects a set of service offerings that satisfy the largest amount of consumers' needs. In our work, however, we do not only talk about selection of existing services, but also creation of new service specifications, as service offerings are flexible and cover more than one specific computational setting. From this perspective, service standardization chooses one specific setting that can be offered by (as many as possible) service providers and satisfy (as many as possible) consumers' needs at the same time.

The process of computational resource standardization comprises an analysis of demand and supply and the creation of a set of resources with specifications closest to the needs of service consumers and service providers. This process may create new resource specifications that are close to the required and offered services. The granularity of the newly created specifications depends only on the current demand and supply. For example, in a market trading computational instances in the infrastructure-as-a-service model, providers offer infrastructure services in forms of virtual machines (VMs), and consumers search for appropriate VMs for running various applications. Specifications of the offered resources are analyzed and grouped into several categories based on their performance, stability, spatial distribution and security. Example of groups of service offerings are virtual environments appropriate for running medical applications, scientific applications, and military intelligence applications. Based on the demand and supply distribution, one standardized resource for each of these groups could be created or more specific branches could be derived. For example, instead of infrastructure appropriate for running medical applications in general, VM offerings could be grouped into those appropriate for running surgery, oncology and computed axial tomography applications. More specifically, resources could be grouped based on the proposed contractual values of certain service objectives in terms of, for example, performance and stability, and additional “sub-niches” could be created (e.g., infrastructure for running medical applications in small ambulances or large city hospitals).

The goal of the resource standardization in electronic markets is mainly limiting the trade to a small number of service types instead of trading numerous differentiated resources. This, in turn, is expected to decrease the heterogeneity of the market and, therefore, the search cost. Consequently, it improves market liquidity. However, it must be considered that the limitation of resource types to be traded in the market most probably results in the reduction of users' utility (i.e., the reduction of satisfaction with the purchased standardized services, as standardized services do not fully match their requirements). Only if the difference between the utility from the decreased search cost and users' reduced utility from the decreased utility is still positive, resource standardization is justified. The trade-off between utility and cost incurred by this approach will be discussed in detail in Section 4.6.

In the following, we describe how requirements of cloud services can be specified using templates of service level agreements and demonstrate how we use the so-called SLA mapping approach as a technique for SLA matching. Furthermore, we discuss in detail the idea of cloud resource standardization and present the methodologies and algorithms for creation and management of standardized cloud services.

4.3 Towards adaptive standardized services in cloud marketplaces: the SLA mapping approach

Before discussing the process of standardization of SLA-based cloud resources, it is important to give a definition of resources and formalize their specification. In our vision of cloud markets, computational services can be described using templates of SLAs, which are collections of service level requirements that have been negotiated and mutually agreed upon by a service provider and a service consumer. SLA templates may differ in their purposes: (1) specification of consumers' service requirements and providers' service offers (*private SLA templates*), and (2) description of (standardized) computational resources that are traded in the market (*public SLA templates*). On the one hand, consumers and providers manually create private SLA templates when submitting service requirements and offerings to the market. On the other hand, public SLA templates are created automatically by the market platform in certain time intervals.

Essentially, SLA templates specify characteristics of services and their SLA parameters, i.e., observable and measurable service properties. Each SLA parameter is defined through:

1. *SLA parameter description section*, i.e., basic parameter properties (e.g., parameter name);
2. an *SLA metric*, which defines a method for measuring the value of the SLA parameter either by aggregating one or more other metrics through a function (composite SLA metrics) or by specifying a measurement directive (resource metrics); and
3. a service level objectives (SLO), which represents provider and consumer obligations and defines guarantees and constraints that may be imposed on the SLA parameter.

Examples of composite metrics are "maximum response time of a service", "average availability of a service", or "minimum throughput of a service". Examples of resource metrics are "system uptime", "service outage period", "number of service invocations". Measurement directives specify how an individual metric can be accessed. Typical examples of measurement directives are the uniform resource identifier of a hosted computer program, a protocol message, or the command for invoking scripts or compiled computer programs. Functions specify how a composite metric is computed. Examples of functions are formulas of arbitrary length containing average, sum, minimum, maximum, and various other arithmetic operators, or time series constructors.

Regarding the specification of service level objectives, first the validity period is specified. It indicates the time intervals for which a given SLA parameter is valid. The SLO also defines a predicate that specifies the threshold and the comparison operator (greater than, equal, less than, etc.) against which a computed SLA parameter is to be compared. The result of the predicate is either "true" or "false". Actions, finally, are triggered whenever a predicate evaluates to "true", i.e., a violation of a guarantee has occurred.

Specification-wise, private and public SLA templates only differ in the definition of service obligations. In private SLA templates, service objectives are defined as ranges of acceptable values (e.g., property "storage" of a service larger than 30TB) in order to allow a certain level of flexibility and improves the probability of finding a match between a consumer's request and a provider's offer. Public SLA templates define single objective values of a certain service (e.g., storage of exactly 35TB) instead.

To cover the wide-range use of the SLA templates, we utilize the SLA mapping approach [28,31,36]. In this approach, syntax differences between semantically equal parameters of SLA templates are bridged by defining translations between differing properties of SLA parameters using Extensible Stylesheet Language Transformations (XSLT) documents named *SLA mappings*. The differences mapped by SLA mappings can range from very simple syntax variations (e.g., different names for the same SLA parameter) to very complex distinctions (e.g., differently expressed, but semantically equal methods for calculating a value of a service level objective).

SLA mappings, therefore, allow large flexibility in specifications of service requirements, which corresponds to the real-world heterogeneity in application definitions.

Based on the mapping complexity and target values, we differentiate two types of SLA mappings:

Ad-hoc SLA mappings define translations between parameter existing in the user's private SLA template and the public SLA template. Within this mapping type, we distinguish *simple ad-hoc mappings*, i.e., mapping of different values for an SLA element (e.g., a mapping between the names "CPUCores" and "NumberOfCores" of an SLA parameter, or a mapping between two different values of a service level objective), and *complex ad-hoc mappings*. The later one maps between different functions used for calculating a value of an SLA parameter (e.g., defining a mapping for a metric unit of a value of an SLA parameter such as "Price" from "EUR" to "USD" can translate one function for calculating price to another one).

Future SLA mappings define wishes for adding (or deleting) a new SLA parameter that is supported by the private template to a public SLA template. Unlike ad-hoc mappings, future mappings cannot be applied immediately.

Figure 4.1 represents a sample public SLA template and a service consumer's private SLA template. Consumer's private template differs from the public template in four ways: (1) the name of the SLA parameter "Price", (2) the unit of a value of the parameter "Performance" (i.e., the parameter metric), (3) the parameter "Availability", which does not exist in the consumer's private SLA template, and (4) the SLA parameter "ClockSpeed", which exists in the consumer's private template but not in the public SLA template. Therefore, the consumer will create four SLA mappings to bridge these difference, namely (i) a simple ad-hoc mapping to map the dif-

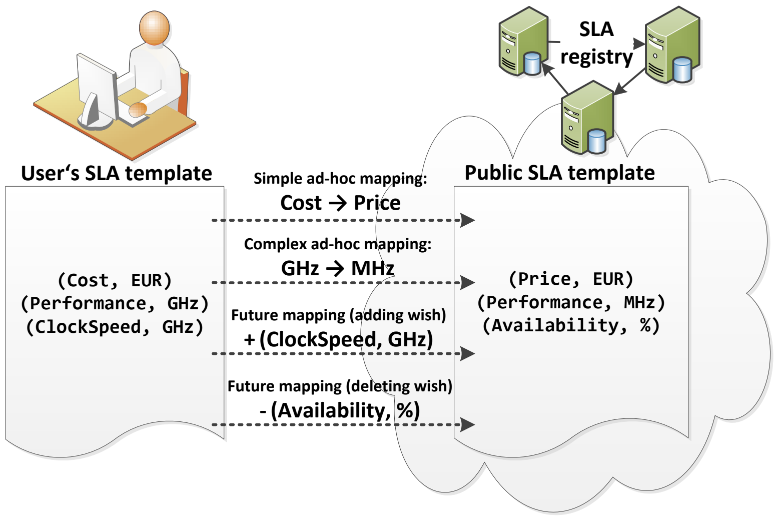


Figure 4.1: An example of the SLA mapping process

ference of the parameter name, (ii) a complex ad-hoc mapping for defining a mapping function for measuring performance, (iii) a deleting wish (future mapping) for wishing for the deletion of the "Availability" parameter from the public SLA template, and (iv) an adding wish (future mapping) for adding a new parameter "ClockSpeed" into the public SLA template.

SLA mappings are important due to the lack of semantic descriptions of SLA parameters. Finding equivalent parameters in SLA templates is impossible unless a user has created a mapping to define this equality. Since standardized products (i.e., public SLA templates) are derived from users' service requirements (i.e., private SLA templates), understanding users' private SLA templates is crucial. Therefore, when entering the market, users are asked to create SLA mappings for their SLA templates.

For the purpose of creating SLA mappings, the market's service directory contains a specially formatted *index SLA template*, which is, in its essence, a catalogue of all SLA parameters that have been used in the marketplace so far. The index template is not used for trading, but only as a market entry point for users to submit SLA mappings. This step is necessary in order to be able to compare users' service specification to all existing specifications in the market.

4.3.1 Formalization of the SLA-based service specification model

To facilitate the discussion on design of algorithms for SLA template adaptation and creation of SLA mappings, we formally define SLAs and SLA templates as assumed in this thesis. In the following, we observe a set of market users $U = \{u_1, u_2, u_3, ..., u_n\}$ where each user $u \in U$ has a role of either a service provider or a service consumer in the market. To simplify our model, we assume that every user $u \in U$ offers or requires exactly one service described by one private SLA template. Let the market contain the set of SLA templates $T = \{T^u \cup T^p \cup t_{index}\}$, where $T^u = \{t_1^u, t_2^u, t_3^u, ..., t_n^u\}$ is a set of private SLA templates of users $u_1, u_2, u_3, ..., u_n$ respectively, $T^p = \{t_1^p, t_2^p, t_3^p, ..., t_m^p\}$ a set of public SLA templates, and t_{index} the index SLA template.

Each SLA template $t \in T$ is defined through a set of SLA parameters $P_t \subseteq P$, where $P = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_k\}$ is a set of all SLA parameters. Every instantiation of an SLA parameter $\alpha \in P$ in an SLA template $t \in \{T \setminus t_{index}\}$ is defined through a function of its basic properties, i.e., parameter description (e.g., name), SLA metric and a value objective in the following way.

$$\Omega : P \to \bigcup_{\alpha \in P} D(\alpha) \times \bigcup_{\alpha \in P} F(\alpha) \times \bigcup_{\alpha \in P} S(\alpha) \tag{4.1}$$

where $D(\alpha) = \{d, d', d'', ...\}$ is a set of possible parameter descriptions, $F(\alpha) = \{f_\alpha, f'_\alpha, f''_\alpha, ...\}$ a set of parameter metrics, and $S(\alpha) = \{s_\alpha, s'_\alpha, s''_\alpha, ...\}$ a set of possible parameter objectives for the SLA parameter α. An SLA parameter is represented as a tuple $(properties, metric, SLO)$, i.e., for an SLA parameter α in an SLA template t it holds

$$\Omega(\alpha) \mapsto (D(\alpha), F(\alpha), S(\alpha)) \tag{4.2}$$

Unlike other SLA templates, the index SLA template t_{index} contains all parameters from the set P and contains no SLO values since it only represents a catalogue of market's SLA parameters and is not used for trading. Therefore, for the index SLA template it holds:

$$\Psi(\alpha) : P \to \bigcup_{\alpha \in P} D(\alpha) \times \bigcup_{\alpha \in P} F(\alpha) \tag{4.3}$$

If the value of an SLA parameter α in an SLA template t is $SLA_t(\alpha) = (d, f_\alpha, s_\alpha)$, then we can define the projection function Π as

$$\Pi_D(SLA_t(\alpha)) = d$$

$$\Pi_F(SLA_t(\alpha)) = f_\alpha$$

$$\Pi_{SLO}(SLA_t(\alpha)) = s_\alpha$$

with the abbreviation

$$D^t_\alpha = \Pi_D(SLA_t(\alpha))$$

$$F^t_\alpha = \Pi_F(SLA_t(\alpha))$$

$$S^t_\alpha = \Pi_{SLO}(SLA_t(\alpha))$$

In our SLA model, we simplify the description section of the SLA parameters and assume that they are defined only by a parameter name. Therefore, $D(\alpha)$ is a set of string values representing possible names of the SLA parameter α. On the other hand, the specification of SLA metrics is fairly complex. As described in Section 4.3, an SLA metric either defines a function for aggregating other SLA metrics or it defines a measurement directive. The aggregation of SLA metrics can be easily represented using a tree structure where a composite SLA metric is an inner node, and a resource SLA metric is a leaf node. As the values can be measured only using the resource metrics, each aggregate (and, naturally, resource) metric is finally represented as an algebraic function on measurement directives. Note that two SLA metric functions f_α and f'_α may use another aggregation function to yield the same result. For example, the following functions always give the same result (i.e., they are "semantically equal") despite different specifications:

$$f_\alpha = \frac{(a-b)c}{d}$$

$$f'_\alpha = \frac{ac}{d} + \frac{ad - bc}{d} - a$$

The SLO value of an SLA parameter $\alpha \in P$ is string, boolean or a numerical value. In case of numerical SLOs, private and public SLA template differ in their definition: private SLA templates define SLOs as single real values, i.e., $s_\alpha = a \in \mathbb{R}$, and public SLA templates define SLOs as ranges of real values, i.e., $s_\alpha : a \leq f_\alpha \leq b$ with $a, b \in \mathbb{R}$.

4.4 Designing a marketplace to support standardized services

SLA mappings allow market users to easily map the differences between specifications of their service requirements to the service offerings in the market. On the other hand, they also enable market to easily retrieve knowledge about users' requirements and offers in the market. Namely, as the semantic relations between the parameters of different services are not available due to the differences in their syntax descriptions, the application of SLA mappings make these relations finally available. Based on this knowledge, the market has the power to automatically create and adapt service offerings (i.e., standardized services) based on the current market demand and supply. In this section, we give an overview of this process.

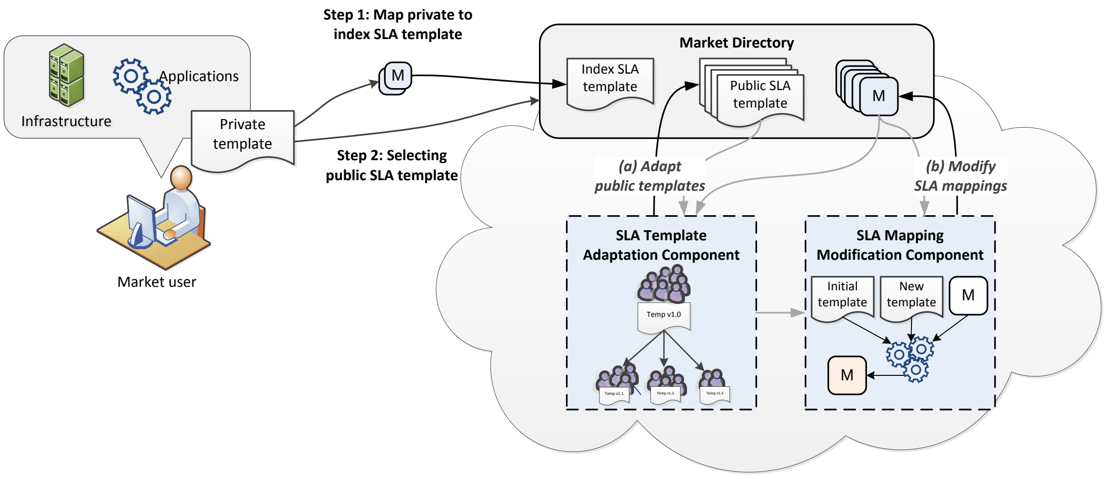

Step 1: Map private to index SLA template
Applications
Infrastructure
Private template
Market user
Step 2: Selecting public SLA template
M
Market Directory
Index SLA template
Public SLA template
M
(a) Adapt public templates
(b) Modify SLA mappings
SLA Template Adaptation Component
Temp v1.0
SLA Mapping Modification Component
Initial template
New template
M
M

Figure 4.2: The SLA mapping approach

Figure 4.2 depicts the process of creating, adapting and trading standardized services as assumed in this thesis. As already discussed, the cloud market only allows trading products that are described by public SLA templates (i.e., the standardized products). When entering the market to offer a service (in case of a service provider) or to request a service (in case of a service consumer), users first "map" their services (as described by their private SLA templates) to the index SLA template (step 1 in Figure 4.2). This process includes fetching the index SLA template, matching parameters from their private SLA template to the index template, and creating the SLA mappings to bridge the differences in the specifications of the SLA parameters of the two templates. In case a user cannot find an SLA parameter from their private SLA templates in the index SLA template, the new parameter is added.

After creating the mappings, the user searches through the market directory and selects the best fitting public SLA template from the catalogue (step 2 in Figure 4.2). The criteria under which public templates are selected include template structures (i.e., whether public SLA templates define all SLA parameters necessary for the user's application) and recommended quality of services (QoS) objectives (i.e., whether values of SLOs are in the acceptable ranges defined by the user's private SLA template).

From the users' perspective, the next step is negotiation with the provider offering the requested service, which includes agreeing on service price, parties' obligations, service provisioning details and other necessary information. Figure 4.2, however, does not depict this process as we are in this chapter interested only in the selection and adaptation of standardized services. Therefore, other trading mechanisms such as service allocation (incl. bidding process), pricing and utilization are herewith not considered.

From the market perspective, the next step is creation and/or adaptation of standardized service specifications (step (a) in Figure 4.2). Standardized cloud products are created and later modified based on the requirements of the market participants. This process is executed automatically in certain time intervals and is achieved through the application of clustering algorithms and adaptation methods, as it will be detailed in Section 4.4.1. Finally, in step (b) in Figure 4.2, SLA mappings are automatically modified to fit the newly created public SLA templates. The motivation and execution of the latter process will be detailed in Section 4.4.2.

4.4.1 Automatic creation and adaptation of standardized SLA products

In the process of creating and adapting standardized cloud products, the market platform utilizes clustering algorithms to group similar demand and supply (i.e., users' private SLA templates) and adaptation methods to create new standardized products (i.e., public SLA templates) from the groups of similar private SLA templates. This process requires three steps, which are depicted in Figure 4.3.

First, a clustering algorithm is applied to group structurally similar private SLA templates. Two SLA templates are similar by their structures if parameter descriptions and metrics of these templates are similar with respect to the distance function defined in Section 4.5.2. Often, these groups are subgroups of existing templates (i.e., branches of new products). For example, considering medical applications, one group of requirements might be for surgical applications, while an another one might be for services in oncology.

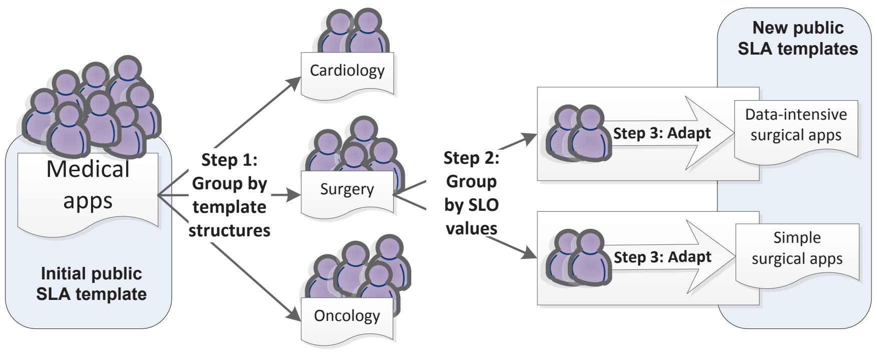

Medical apps
Initial public SLA template
Step 1: Group by template structures
Cardiology
Surgery
Oncology
Step 2: Group by SLO values
Step 3: Adapt
Step 3: Adapt
New public SLA templates
Data-intensive surgical apps
Simple surgical apps

Figure 4.3: Adaptation of SLA templates

In the second step of the adaptation process, for each generated cluster, a clustering algorithm is applied to create subgroups of requirements according to the SLO values expressed in users' private SLA templates. For example, in the group for surgical applications, a clustering algorithm might recognize a subgroup for data-intensive surgical applications (e.g., for big city hospitals), and another for simple surgical applications (e.g., for small ambulances).

For each of the subgroups of user requirements created by the second iteration of clustering, a new public SLA template is created. This is performed by applying an adaptation method (step 3 in Figure 4.3). The adaptation method determines for each SLA parameter from the current public SLA template whether its properties should be changed, a new parameter should be introduced, or an existing parameter should be deleted. After the new public SLA templates have been created, SLA mappings to the private SLA templates are created, the new public SLA templates published in the directory, and the new public SLA templates are assigned to the user.

Public SLA template adaptation is executed continuously. In this process, the newly created SLA templates replace the “deprecated” templates. However, there still may be some ongoing auctions and trades that use the old public SLA templates. In this case, it is ensured that all the auctions and trades come to an end before removing the templates from the directory.

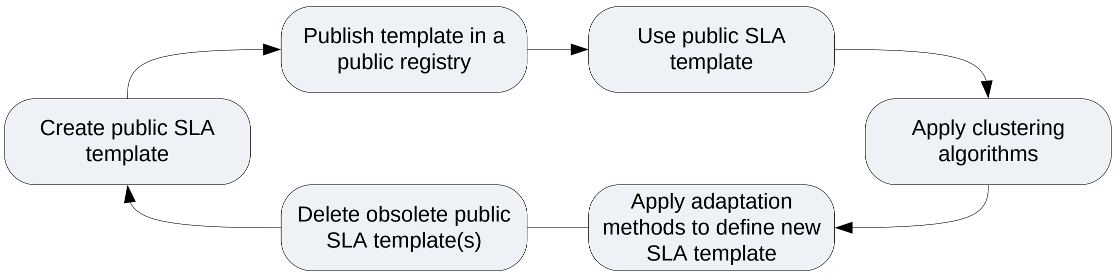

Publish template in a public registry
Use public SLA template
Create public SLA template
Apply clustering algorithms
Delete obsolete public SLA template(s)
Apply adaptation methods to define new SLA template

Figure 4.4: A typical lifecycle of an SLA template

To summarize, Figure 4.4 illustrates the lifecycle of an SLA template and demonstrates how public SLA templates are created and managed. In particular, six lifecycle phases can be distinguished: (1) creation of public SLA templates, (2) publishing the public SLA templates in a registry, (3) the use of public SLA templates for SLA mapping, (4) the definition and application of template adaptation methods, (5) the definition of adapted public SLA templates, and (6) the deletion of “unpopular” SLA templates.

4.4.2 Adapting SLA mappings to the new public SLA templates

After the creation and adaptation of standardized cloud services, the next step in the market adaptation process is modification of the SLA mappings to fit the newly created public SLA templates. Namely, although new public SLA templates reflect users’ needs more precisely, users might prefer keeping the old public templates instead of using the new ones. This is because of the cost of creating new SLA mappings to the adapted public SLA templates, while the usage of the existing templates does not incur any additional cost. However, allowing users to utilize outdated templates results in a constant increase of the number of public SLA templates in the service directory. Consequentially, due to the increase in number of goods, this action increases the costs of searching for trading partner and implicitly has a negative impact on market liquidity. To prevent this, the market should enable users to utilize new public SLA templates without facing additional costs.

To achieve this goal, we investigate automatic modifications and creations of SLA mappings in this section. Namely, we update SLA mappings for users by concatenate the existing SLA mapping with the same SLA mapping used to transform the current public SLA templates into the new public templates (step (b) in Figure 4.2). As it will be shown in Chapter 7, this approach dramatically reduces the cost for users and enables the market to delete the old public SLA templates from the market directory, ensuring low cost of market maintenance.

4.5 Methods and algorithms for managing adaptive cloud resources

In this section, we discuss the details of creating and adapting standardized products. First, we describe the clustering algorithms to group similar requirements of users (Section 4.5.1) based on a function for computing similarity of SLA-based resources (Section 4.5.2). Second, based on the requirements that define a cluster, we compute a new standardized product (i.e., a public SLA template) that will be the closest to the needs of the group of users. For this purpose, we utilize the adaptation methods (Section 4.5.3). Finally, after the new public SLA templates have been created, we apply an algorithm to automatically modify or create new SLA mappings to the new public templates so as to reduce the cost to users (Section 4.5.4). The methods and algorithms that will be presented in this section have been implemented as extensions to VieSLAf [30, 31], which is a tool for semi-automatic management of SLA templates and SLA mappings previously developed at the Vienna University of Technology.

4.5.1 Grouping private SLA templates

For the purpose of grouping similar private SLA templates, we apply two clustering algorithms: DBSCAN and k-means.

DBSCAN is a data clustering algorithm that is based on the density distribution of nodes and finds an appropriate number of clusters [84]. The ϵ-neighborhood of a point p is defined as a set of points that are not farther away from p than a given distance ϵ. A point q is directly density-reachable from the point p if q is in the ϵ-neighborhood of p and the number of points in the ϵ-neighborhood of q is bigger than $MinPts$. A cluster satisfies two properties: (1) each two points are mutually density-reachable, and (2) if a point is mutually density-reachable to any point of the cluster, it is part of the cluster as well.

k-means is a clustering method that partitions N data points into k disjoint subsets S_j containing N_j data points so as to minimize the sum-of-square criterion $\sum_{j=1}^{k} \sum_{n \in S_j} d(x_n, \mu_j)^2$, where x_n is the nth data point, μ_j the geometric centroid of the data points in S_j, and d a distance function calculating the similarity of the two elements [139]. The algorithm, given an initial set of k means, assigns each data point to a cluster with the closest mean. It then calculates new means to be centroids of observations in the clusters and stops when the assignments no longer change the means. In our context, a data point is a user's private SLA template, and a cluster centroid is a new public SLA template for the group of users.

A frequent problem in k-means algorithm is the estimation of the number k. Within this section, we discuss two approaches:

1. *Rule-of-thumb* is a simple but very effective method for estimation the number k. The variable k is set to $\sqrt{N/2}$, where N is the number of entities [141].

2. *Hartigan's index* is an internal index for scoring the number of clusters introduced in [110]. Let $W(k)$ represent the sum of squared distances between cluster members and their cluster centroid for k clusters. When grouping n items, the optimal number k is chosen so that the relative change of $W(k)$ multiplied with the correction index $\gamma(k) = n - k - 1$ does not significantly change for $k + 1$, i.e.,

$$H(k) = \gamma(k)\frac{W(k) - W(k+1)}{W(k+1)} < 10 \tag{4.4}$$

   The threshold 10 shown in Hartigan's index is also used in our simulations. It is "a crude rule of thumb" suggested by Hartigan [110].

As already discussed, the number k determines the number of standardized services (i.e., public SLA templates) and, therefore, affects market diversity and implicitly market liquidity. Since our goal is to maximize market liquidity, we aim at choosing the number k so that this goal is achieved. For this reason, besides the two given heuristic methods, the third method of determining the number k in our work is through the measure of market liquidity. As a method for quantifying liquidity in cloud markets we use a modification of liquidity measures from Section 3.2, which we will introduce in Section 4.6. Using this liquidity measure, we select the number k in which liquidity is maximized.

Applying the k-means clustering algorithm significantly differs in the situations in which the number k is determined by the rule-of-thumb and by the Hartigan's or the maximization function of the liquidity approximation. Namely, in the rule-of-thumb scenario, the number k is an a priori knowledge: the algorithm creates k clusters and groups the points around the centroids. On the other hand, the number k is in the other scenarios an a posteriori knowledge. In particular, the clustering algorithm must be applied for a numerous number of iterations (with different ks) and the result clusters are selected in which the Hartigan's index or the market liquidity are maximized. These processes, therefore, may create large redundancies and require more computational power and time. However, as it will be discussed in detail in Chapter 7, this process may be significantly optimized and time-efficient.

4.5.2 Computing distance between SLA templates

For determining the ϵ-neighborhood of a clustering point in case of DBSCAN and for computing the sum-of-square criterion in case of the k-means algorithm, we must define a function measuring distance (i.e., similarity) of two clustering elements, i.e., SLA templates. For this purpose, we use an n-tuple representation of SLAs. Note that since a cluster centroid is also an SLA template, this method can be also used to measure the distance between a clustering element and a cluster centroid.

As already discussed, an SLA comprises SLA parameters, which in turn aggregate SLA metrics and SLO values. From the SLA specification perspective, we may differ the *SLA structure*, i.e., structural organization and SLA specification, from the *contractual values and obligations* of the SLA. Essentially, the structure includes SLA parameter descriptions and metrics, while the SLA values include numerical, boolean and string values of SLOs and SLA attributes. Having this in mind, we create an n-tuple representation of an SLA, where first $n-1$ elements of the tuple contain SLA values and the last (nth) element comprises the SLA structure. By using such a representation, we can represent the distance between two SLA templates also as an n-tuple, where first $n-1$ elements contain the differences between the two values of each of the parameters, while the final element contains a value representing the difference between the structures of the SLAs.

To formalize using the SLA specification model defined in Section 4.3.1, we observe two SLA templates $t_1, t_2 \in T^u \cup T^p$ containing SLA parameters $\{\alpha, \beta, \gamma, ...\}$. If Δ is an SLA structure, the SLAs t_1 and t_2 are represented as n-tuples $\vec{t_1}$ and $\vec{t_2}$ respectively as:

$$\vec{t_1} = (S_\alpha^{t_1}, S_\beta^{t_1}, S_\gamma^{t_1}, ..., \Delta_{t_1}) \tag{4.5}$$

$$\vec{t_2} = (S_\alpha^{t_2}, S_\beta^{t_2}, S_\gamma^{t_2}, ..., \Delta_{t_2}) \tag{4.6}$$

Note that the order of SLO values in the tuples must be made consistently with all SLAs that are considered in order to allow a simple comparison between the tuples. In case there exists an SLA parameter (and, consequentially, an SLO) that exists in some, but not in all SLA templates, the values of that SLO should be set to $\emptyset$ for the SLA templates that do not contain the SLO.

An n-tuple $D_{\vec{t_1},\vec{t_2}}$ representing the distance between two SLA templates is defined as:

$$D_{\vec{t_1},\vec{t_2}} = (f(S_\alpha^{t_1}, S_\alpha^{t_2}), f(S_\beta^{t_1}, S_\beta^{t_2}), f(S_\gamma^{t_1}, S_\gamma^{t_2}), ..., F(\Delta_{t_1}, \Delta_{t_2})) \tag{4.7}$$

where f is a function calculating the difference between two SLA values, and F a function for calculating the difference between two SLA structures.

The result of the function f depends on the type of its arguments and is defined as follows.

$$f(S_\alpha^{t_1}, S_\alpha^{t_2}) = \begin{cases} |S_\alpha^{t_1} - S_\alpha^{t_2}|, & \text{if } S_\alpha^{t_1}, S_\alpha^{t_2} \text{ are numerical} \\ 0, & \text{if } S_\alpha^{t_1}, S_\alpha^{t_2} \text{ are not numerical and } S_\alpha^{t_1} = S_\alpha^{t_2} \\ 1, & \text{if } S_\alpha^{t_1}, S_\alpha^{t_2} \text{ are not numerical and } S_\alpha^{t_1} \neq S_\alpha^{t_2} \end{cases} \tag{4.8}$$

However, the first condition of Equation (4.8) assumes that the numerical SLO values are represented as single real numbers, which holds only in case of public SLA templates. Therefore, in order to compute a distance between SLOs of private SLA templates, which are represented as ranges of real values, we need a novel method.

In order to compute the distance between SLO value ranges, we utilize the Pompeiu-Hausdorff metric [178], which defines this value as a maximum distance of one range to the nearest point in the other range. In detail, the Pompeiu-Hausdorff distance $d_h(X, Y)$ between two non-empty subsets X and Y of a metric space M (in our case, two intervals in $\mathbb{R}$) is defined as:

$$d_h(X, Y) = max\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y)\} \tag{4.9}$$

where sup represents the supremum, inf the infimum, and $d(x, y)$ any metric between points x and y. The Pompeiu-Hausdorff distance is calculated by considering a point $x \in X$ and finding the least distance to a point $y \in Y$. This calculation is repeated for all $x \in X$ to find the maximum value. In the next step, the same process is performed with the roles of X and Y reversed. Finally, the largest of these two values is taken as the result.

In our case, X and Y are values of service level objectives of an SLA parameter α from private SLA templates t_1 and t_2, noted $S_\alpha^{t_1}$ and $S_\alpha^{t_2}$. Since SLO values are unbounded sets of real numbers, $d(x, y)$ is the Euclidean distance between points x and y. Having said that, considering Equation (4.9), for any two bounded ranges $S_\alpha^{t_1} = [x_1, x_2]$ and $S_\alpha^{t_2} = [y_1, y_2]$, it is simple to show that their Pompeiu-Hausdorff distance is

$$d_h(S_\alpha^{t_1}, S_\alpha^{t_2}) = max(|x_1 - y_1|, |x_2 - y_2|) \tag{4.10}$$

The distance between the structures of SLA templates is expressed as a number of differences between parameter properties of two SLA templates. This value is calculated by iterating through each SLA parameter contained by at least one of the SLA templates and determining the distance for this SLA parameter. For two SLA templates t_1 and t_2, the distance between their structures is defined as:

$$F(\Delta_{t_1}, \Delta_{t_2}) = \sum_{p \in t_1 \cup t_2} d_p^{\Delta}(t_1, t_2) \tag{4.11}$$

where the distance between the specification of an SLA parameter α of the two SLA templates with respect to the parameter properties (i.e., its parameter description and metric) is defined as:

$$d_p^{\Delta}(t_1, t_2) = \begin{cases} 0, & \text{if properties of } p \text{ are same in } t_1 \text{ and } t_2 \\ 1, & \text{if } t_1 \text{ or } t_2 \text{ does not contain } p \text{ or if one property of } p \text{ differs in } t_1 \text{ and } t_2 \\ 2, & \text{if both properties of } p \text{ differ in } t_1 \text{ and } t_2 \end{cases} \tag{4.12}$$

Using the given distance functions the n-tuple representing the distance between n-tuple representation of two SLA templates can be easily computed: first $n - 1$ elements are Pompeiu-Hausdorff distances between templates' SLO values (Equation (4.10)), and the last (nth) is the distance between templates' structures (Equation (4.11)). After the result tuple has been calculated, it can be used to generate a single numerical value representing the distance between the two SLA templates. This value can be computed by applying a simple function on the n-tuples (e.g., summation of element values). However, in order to do so, the result tuple must be first normalized. This step is necessary as the SLO values of different SLA parameters are usually expressed in different measurement units and the values are, therefore, not mutually comparable. Normalization function fits all SLO values into the range $[0, 1]$, which is done by applying the following equation:

$$d_n(S_\mu^{t_1}, S_\mu^{t_2}) = \frac{d_h(S_\mu^{t_1}, S_\mu^{t_2}) - min\big(d_h(S_\mu)\big)}{max\big(d_h(S_\mu)\big) - min\big(d_h(S_\mu)\big)} \tag{4.13}$$

where $d_h(S(\mu))$ is the set of all distances between all SLA templates with respect to the SLO value of the SLA parameter μ. If a value is contained by only one of the two templates, the distance is maximum, i.e., equal to 1. In case the SLO values are expressed in a unit different from the SLO unit defined by the initial public SLA template, the value is being converted into the unit of the public template before computing the distance. Note, however, that the clustering methods presented earlier in this chapter are applied in two phases: first to group by templates' structures, and then by SLO values. In these cases, only certain elements of n-tuple are considered (first $n-1$ elements when grouping by SLO values and only the last element when grouping by structures). A single distance value for the entire n-tuple is, therefore, only used in special cases when the difference in terms of both aspects of SLA templates (i.e., structures and SLO values) are needed in an equal extent.

To demonstrate the methods for computing the distance between SLA templates, we use the following example. Table 4.1 depicts three private SLA templates t_1, t_2, and t_3 and an initial public SLA template t_{init} to which private templates are associated. Parameter definitions are given as tuples $(Name, Unit, SLO)$, in which the elements represent the parameter description (in this example simplified to the parameter name), the parameter metric (in this example simplified to the metric unit), and the SLO value.

Table 4.1: Sample SLA templates

SLA template	**SLA parameters**
t_{init}	$\pi : (ResponseTime, second, \langle 1, 3 \rangle)$ $\mu : (ErrorRate, percentage, \langle 0, 1 \rangle)$
t_1	$\pi : (RespTime, second, \langle 1, 4 \rangle)$ $\mu : (ErrRate, percentage, \langle 0, 2 \rangle)$
t_2	$\pi : (ResponseTime, millisecond, \langle 800, 3300 \rangle)$ $\mu : (ErrorRate, percentage, \langle 1, 3 \rangle)$
t_3	$\pi : (RespTime, millisecond, \langle 1100, 4500 \rangle)$ $\mu : (Error, percentage, \langle 0, 1 \rangle)$

With respect to the SLA parameter π, private SLA templates t_1 and t_2 differ in both the parameter name and the parameter unit. Therefore, the distance between the template structures $d_{S,\pi}(t_1, t_2)$ is equal to 2. Since the specification of the parameter μ differs only in its name, the distance $d_{S,\mu}(t_1, t_2)$ is equal to 1. In total, the distance between structures of t_1 and t_2 is equal to $d_S(t_1, t_2) = d_{S,\pi}(t_1, t_2) + d_{S,\mu}(t_1, t_2) = 3$. Similarly, we get the distances $d_S(t_1, t_3) = 2$ and $d_S(t_2, t_3) = 2$.

Before computing the distances between the SLO values, they must be converted to the unit stated in the initial public SLA template. This is performed by applying the SLA mappings submitted by the users. Then, the distance between SLO values between t_1 and t_2 with respect to the SLA parameter π is calculated as:

$$d_h(S_\pi^{t_1}, S_\pi^{t_2}) = max(|1 - 0.8|, |4 - 3.3|) = 0.7$$

Similarly, the distances for the other SLA parameters and SLA templates can be calculated:

$$d_h(S_\pi^{t_1}, S_\pi^{t_3}) = 0.5, d_h(S_\pi^{t_2}, S_\pi^{t_3}) = 1.2, d_h(S_\mu^{t_1}, S_\mu^{t_2}) = 1, d_h(S_\mu^{t_1}, S_\mu^{t_3}) = 1, d_h(S_\mu^{t_2}, S_\mu^{t_3}) = 2$$

After all Hausdorff distances have been calculated, they can be normalized using the Equation (4.13):

$$d_n(S_\pi^{t_1}, S_\pi^{t_2}) = \frac{d_h(S_\pi^{t_1}, S_\pi^{t_2}) - min\big(d_h(S(\pi))\big)}{max\big(d_h(S(\pi))\big) - min\big(d_h(S(\pi))\big)} = \frac{0.7 - 0.5}{1.2 - 0.5} = 0.285$$
$$d_n(S_\mu^{t_1}, S_\mu^{t_2}) = \frac{1-1}{2-1} = 0$$

Finally, the total distance between SLA templates t_1 and t_2 with respect to the SLO values is

$$d_S(t_1, t_2) = d_n(S_\pi^{t_1}, S_\pi^{t_2}) + d_n(S_\mu^{t_1}, S_\mu^{t_2}) = 0.285$$

4.5.3 Adaptation methods for the evolution of public SLA templates

For adapting the standardized products (i.e., the public SLA templates), so that they optimally reflect user requirements, we utilize three adaptation methods. For each public SLA template (in this process called *initial public SLA template*) and for each SLA parameter contained in the initial public SLA template, these methods determine whether the current parameter name and metric should be changed, a new parameter should be added, or an existing one deleted. This is performed by analyzing the distribution of parameter preferences of the users, who use the public SLA template. This analysis comprises sorting, classification, and counting of SLA mappings that users created for an SLA parameter. In particular, the adaptation methods apply selection criteria (which are specific to each of the methods), in order to find the SLA parameter value of each SLA parameter that are preferred by users. These values are then used to define a new public SLA template and replace the initial public SLA template.

To demonstrate the workings of these methods, we use the following example. We consider the evolution of an SLA parameter π occurring in an initial public SLA template T_{init}, when adapting the template based on SLA mappings of 100 users. The name and metric of the parameter π in the initial template is $(Price, EUR)$. It is assumed that 20% of all users do not use the parameter π in their private SLA templates, and the rest of them define SLA mappings according to the distribution presented in Table 4.2. Note, the last column of the Table 4.2 represents the number of non-mapped values, i.e., the number of private SLA templates containing the same value as in the initial public SLA template. Table 4.3 represents the distribution of an SLA parameter μ, which exists in 75% of users' private SLA templates and does not exist in the initial public SLA template.

The **maximum method** selects the option that has the highest number of SLA mappings. This option is called the maximum candidate. The maximum candidate is then used in the new SLA template. If there is more than one maximum candidate with the same number of SLA mappings, one of them is chosen randomly. With respect to the given example, since the majority of users utilize the parameter π, it stays in the template. $Rate$ becomes the new parameter name because of the highest number of mappings (40% in comparison to 30%, who want to keep the name $Price$, i.e., who did not create mappings). The price will be expressed in Japanese Yens due to the highest number of mappings. The parameter μ will also be in the new template, since

Table 4.2: Distribution of mappings for the SLA parameter π

a) Name of the SLA parameter π

Name Used:	*Cost*	*Charge*	*Rate*	(*Price*)
Number of Mappings:	15%	15%	40%	(30%)

b) Metric of the SLA parameter π

Metric Used:	*USD*	*GBP*	*YPI*	(*EUR*)
Number of Mappings:	38%	2%	40%	(20%)

Table 4.3: Distribution of mappings for the SLA parameter μ

a) Name of the SLA parameter μ

Name Used:	*MemoryConsumption*	*Consumption*
Number of Mappings:	70%	30%

b) Metric of the SLA parameter μ

Metric Used:	*Mbit*	*Gbit*	*Tbit*
Number of Mappings:	5%	55%	40%

more than 50% of the users utilize it in their private SLA templates, and parameter properties will be $(MemoryConsumption, Gbit)$.

In order to increase the requirements for selecting the maximum candidate, the **threshold method** introduces a threshold value. In this method, the property is chosen if its property value is used more than the given threshold and has the highest count. If more than one parameter property value satisfies the two conditions, one of them is chosen randomly. Throughout the evaluation in this thesis, we fix the threshold to be 60%. In the given example, neither the mappings for the name nor for the metric of the parameter π exceeds the threshold value. As for the new parameter μ, it will be represented in the updated public template according to the properties chosen by the maximum method.

The **significant-change method** changes an SLA parameter property value, only if the percentage difference between the maximum candidate and the current public SLA template value exceeds a given threshold, which we assume to be significant at $\sigma_T > 15\%$. In the given example, 40% of users have the name $Rate$ for the parameter $Price$, while 30% of users use the same name as in the public SLA template. Since the percentage difference of 33% is higher than the given threshold, $Rate$ will be chosen as the new name for the parameter. As the new metric, YPI will be chosen. As the parameter μ does not exist in the old public SLA template, the decision about this SLA parameter is made as described for the maximum method.

Regarding the SLO values of public SLA templates, they are computed using the Marzullo's intersection algorithm [144]. Marzullo's algorithm is an agreement algorithm used to select sources for estimating accurate time from a number of noisy time sources. Given a set of intervals, it will return the central point of the smallest interval consistent with the largest number of sources. In the following, we demonstrate the working of the Marzullo's algorithm on an example.

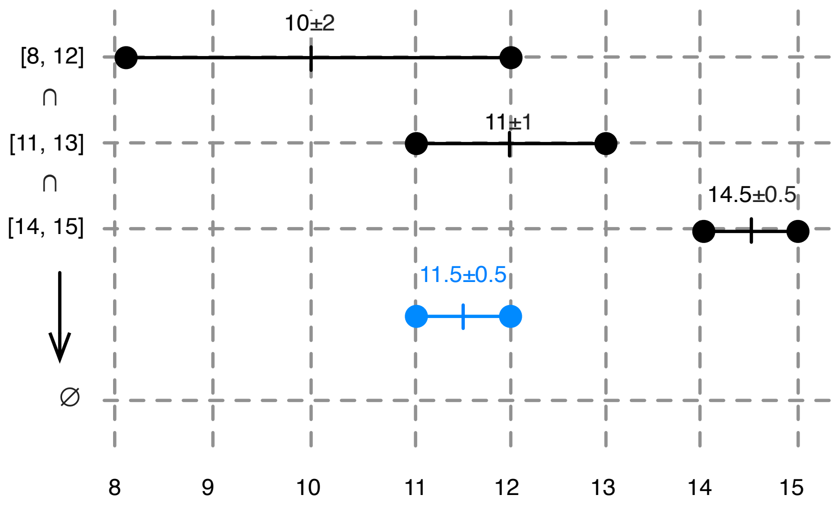


Figure 4.5: Example of applying the Marzullo's intersection algorithm

Consider creating a public SLA template for a group of 3 private SLA templates t_1, t_2, t_3. For an arbitrary parameter α, the private SLA templates contain the following values:

$$8 < F_{\alpha}^{t_1} < 12,$$

$$11 < F_{\alpha}^{t_1} < 13,$$

$$14 < F_{\alpha}^{t_1} < 15.$$

As demonstrated in Figure 4.5, the algorithm selects the interval that is consistent with the largest number of sources (in this case $11 < t < 12$). Since unlike the private SLA templates a public SLA template uses single numberical values for parameter SLOs, the algorithm takes the middle value (in this case $t = 11.5$) as the SLO value of the newly created public SLA template.

4.5.4 Automatic adaptation of SLA mappings to the new public SLA templates

In order to reduce the cost of creating new SLA mappings for users and, therefore, make the market more attractive, users' SLA mappings are automatically redefined once a new public SLA template has been introduced. This way, users' existing SLA mappings can always be used for newly created public SLA templates.

For our discussion of the algorithm for autonomic SLA mapping, we consider the index public SLA template t_{index}, a user's private SLA template t_{user}, an initial public SLA template t_{init}, and a newly generated public SLA template t_{new}. As already defined in Section 4.3.1, an SLA parameter α of an SLA template t is defined by its description D_{α}^{t}, its metric F_{α}^{t}, and its SLO value S_{α}^{t}, which is denoted as $[D_{\alpha}^{t}, F_{\alpha}^{t}, S_{\alpha}^{t}]$. An SLA mapping between descriptions of an

SLA parameter α in SLA templates t_1 and t_2 is denoted as $D_\alpha^{t_1} \leftrightarrow D_\alpha^{t_2}$. Additionally, we use the function χ to determine whether an SLA template t contains an SLA parameter α:

$$\chi_t(\alpha) = \begin{cases} false, & \text{if } t \text{ does not contain } \alpha \\ true, & \text{if } t \text{ contains } \alpha \end{cases} \tag{4.15}$$

The algorithm iterates through all SLA mappings applied to transform t_{init} into t_{new}, and, for each mapping, it executes one of the possible transformation actions (Figure 4.6). As a first step, the algorithm checks whether α exists in at least t_{init} or t_{new} and that its properties differ in those templates. In case that this does not hold, the existing user's SLA mappings for this parameter are kept identical (lines 1-2).

In case α exists in all observed SLA templates (line 3), one of the following actions is executed:

1. If a parameter property did not differ in t_{init} and t_{user}, but it changed in t_{new}, a new SLA mapping is created to map the newly created difference (lines 4-5).

2. If a parameter property differs in all three templates, a new SLA mapping is created (lines 6-7). It is a combination (concatenation) of two existing mappings so that the output of one becomes the input for the second mapping, as illustrated in Figure 4.7.

3. If a parameter property does not differ in t_{user} and t_{new}, the existing SLA mapping is deleted (lines 8-9).

In case α was deleted from t_{init} but needed by the user, the algorithm informs the user about this action as the user will not be able to utilize the parameter anymore (lines 11-12). The algorithm also deletes a possibly existing SLA mapping for the deleted parameter (lines 13-14).

If the parameter was removed from the public SLA template, but the user does not need it, there is nothing to be executed (lines 16-17).

If a new parameter is introduced in t_{new} (line 18) and if the user's private SLA template contains that parameter (line 19), a new SLA mapping is created, which is a combination of two SLA mappings: (1) the SLA mapping between the property values stated in the user's private SLA template and the index template, and (2) the SLA mapping between the values stated in the new public SLA template and the index template (line 20). Besides creating a new SLA mapping, the algorithm also informs the user about the possibility of using an additional SLA parameter in the the market (line 21). Note, the SLA mapping to the parameter of the index SLA template exists, since the index template contains parameters from all public and private SLA templates and since all users create mappings to those parameters when entering the market. If a new SLA parameter is introduced in t_{new}, but the user does not have it in the private SLA template, the algorithm only informs the user about the possibility of using a new SLA parameter as soon as the user manually created an SLA mapping (lines 22-23).

Finally, if the parameter properties changed in t_{init} and t_{new}, but the user does not utilize the parameter, there is no need to perform any action as the changes in the parameter properties are not of the user's concern (lines 25-26).

```
1:  if $\chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = false \vee$
      $(\chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = true \wedge D_\alpha^{t_{new}} = D_\alpha^{t_{init}})$ then
2:    keep identical SLA mappings
3:  else if $\chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = \chi_{t_{user}}(\alpha) = true$ then
4:    if $D_\alpha^{t_{init}} \neq D_\alpha^{t_{new}} \wedge D_\alpha^{t_{init}} = D_\alpha^{t_{user}}$ then
5:      create $D_\alpha^{t_{new}} \leftrightarrow D_\alpha^{t_{user}}$
6:    else if $D_\alpha^{t_{new}} \neq D_\alpha^{t_{init}} \neq D_\alpha^{t_{user}}$ then
7:      combine $D_\alpha^{t_{new}} \leftrightarrow D_\alpha^{t_{init}}$ and $D_\alpha^{t_{init}} \leftrightarrow D_\alpha^{t_{user}}$
8:    else if $D_\alpha^{t_{new}} \neq D_\alpha^{t_{init}} \wedge D_\alpha^{t_{new}} = D_\alpha^{t_{user}}$ then
9:      delete SLA mapping $D_\alpha^{t_{init}} \leftrightarrow D_\alpha^{t_{user}}$
10:   end if
11: else if $\chi_{t_{new}}(\alpha) = false \wedge \chi_{t_{init}}(\alpha) = \chi_{t_{user}}(\alpha) = true$ then
12:   warn user about limited usage of $\alpha$
13:   if $D_\alpha^{t_{init}} \neq D_\alpha^{t_{user}}$ then
14:     delete existing SLA mapping $D_\alpha^{t_{init}} \leftrightarrow D_\alpha^{t_{user}}$
15:   end if
16: else if $\chi_{t_{init}}(\alpha) = true \wedge \chi_{t_{new}}(\alpha) = \chi_{t_{user}}(\alpha) = false$ then
17:   do nothing
18: else if $\chi_{t_{init}}(\alpha) = false \wedge \chi_{t_{new}}(\alpha) = true$ then
19:   if $\chi_{t_{user}}(\alpha) = true$ then
20:     combine $D_\alpha^{t_{new}} \leftrightarrow D_\alpha^{t_{index}}$ and $D_{index}(\alpha) \leftrightarrow D_\alpha^{t_{user}}$
21:     inform user about possible utilization of $\alpha$
22:   else
23:     inform user about new parameter $[D_\alpha^{t_{new}}, F_\alpha^{t_{new}}, S_\alpha^{t_{new}}]$
24:   end if
25: else if $\chi_{t_{init}}(\alpha) = \chi_{t_{new}}(\alpha) = true \wedge \chi_{t_{user}}(\alpha) = false$ then
26:   do nothing
27: end if
```

Figure 4.6: Algorithm for automatic SLA mapping modifications

Note, Figure 4.6 deals with parameter descriptions only, while our implementation also considers their metrics. Our implementation simply achieves that by replacing D_α^T with F_α^T in the algorithm presented.

4.6 Cost-benefit analysis of the approach

On the one hand, the approach presented in this chapter is designed to improve market liquidity. This is the case since the number of resource traded in the market is reduced, which is expected to positively reflect on the effort and probability of finding a trading partner in the market. How-

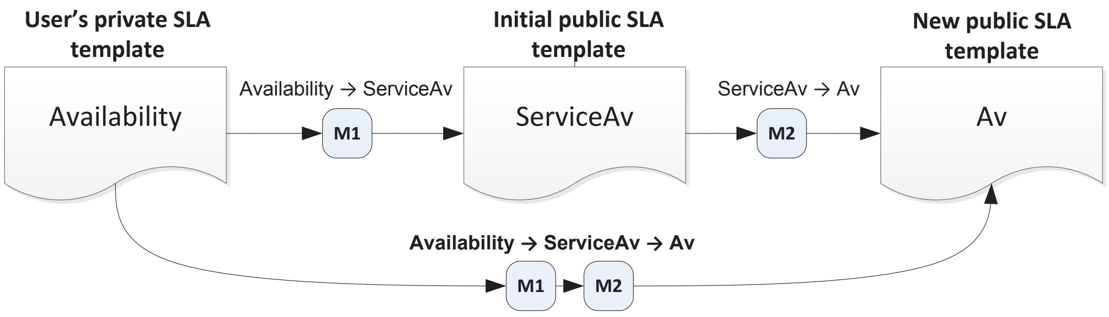


Figure 4.7: Building an SLA mapping as a combination of two mappings

ever, demonstrating the impact on market liquidity is hard as there are no measures of market liquidity in electronic commodity markets. For this reason, we have in Section 3.2 introduced our novel liquidity measures for cloud markets based on the common measures from the existing literature on financial and stock markets. In this section, we continue the discussion on market liquidity in cloud markets and particularly focus on its aspects of resource diversity and complexity.

Besides liquidity, which measures the quality of the market, in this section we discuss the positive and negative effects of the SLA mapping approach. In particular, SLA mapping bring numerous (hardly quantifiable) benefits: they allows flexible SLA usage without explicit resource limitations, i.e., they enable market participants to define their requirements in arbitrary forms and to trade (standardized) goods without changing already existing private SLA templates that may still be in use. Furthermore, they facilitate adaptations and changes in the market. However, the SLA mapping approach also incurs costs to market participants as they must find differences in SLA specifications and (manually or automatically) map those differences. In Section 4.6.2, we define a cost model to represent the benefits and detriments to the market participants incurred by the SLA mapping approach.

4.6.1 Measuring market liquidity

Ensuring liquidity of electronic market assumes that it is possible to monitor and assess liquidity in these environments. However, due to its complex definition, measuring market liquidity is not a trivial task. Many factors affect liquidity, including institutional factors such as securities law, the regulation and supervision of dealers, and accounting rules. Equally, environmental factors such as the macroeconomic situation and changes play a role. Due to the numerous aspects of liquidity and its complex and fairly abstract definition, quantifying liquidity is not a trivial task. This conclusion was also remarked by O'Hara ([160] pg. 125), who wrote that "liquidity, like pornography, is easily recognized, but not so easily defined".

As we have already discussed in Section 3.2, there exist, however, research works that investigate how some environmental factors affect liquidity and how this effect can be formulated and market liquidity quantified. Most of the work focus on financial markets [15, 23, 55, 121, 224]. Although differing in the definition of liquidity and the methods for its assessment, these works commonly conclude that the highly abstract definition of market liquidity cannot be expressed as an aggregate value. Instead, there exist several standard measures that serve as proxies for its assessment. The most common measures, which we have briefly mentioned in Section 3.2, include bid-ask spreads [6], market depth [15], and immediacy [55]. In the following, we summarize how each of these measures can successfully capture at least one of the perspectives of market liquidity.

Bid-ask spread denotes the amount by which an ask, i.e., a seller's offered price, exceeds a bid, i.e., a buyer's requested price. The bid-ask spread essentially measures the difference in price between the highest price that a buyer is willing to pay for a product and the lowest price for which a seller is willing to sell it. Large bid-ask spreads indicate high buying possibilities of the buyers: their request prices are higher than the prices offered by the sellers, which results in more numerous trades in the market. Since the large bid-ask spread points to a high trading dynamism, it also leads to the conclusion that the market liquidity rises proportionally to the increase of its value.

Market depth measures the volume of goods traded in the market, i.e., the units that can be sold or bought for a given price impact. Particularly, market depth refers to the maximum size of a trade for any given bid/ask spread. A market may be considered deeply liquid if there are ready and willing buyers and sellers in large quantities, which is directly related to the concept of market depth as a large number of market traders and service offerings as well as a well-designed allocation mechanism result in a large trading volume in a time period. This suggests that a high market depth implies that the assets can be easily purchased or sold. Therefore, high market depth indicates high market liquidity.

Immediacy refers to the time needed to successfully trade a certain amount of a product at a prescribed cost. Essentially, immediacy can be measured as the time passed between the submission of a requirement for a service to a market and the allocation (i.e., a match) between the buyer's requirement and a seller's offering. Depending on the actor, it is possible to differentiate buyer's immediacy from seller's immediacy. Small immediacy characterizes a small time needed to close a trade and indicates a liquid market.

The presented measures for approximating market liquidity are commonly used in financial literature to measure liquidity of monetary assets. As discussed in Section 3.2, our modification of these measures is capable of measuring liquidity in cloud markets. However, in this chapter, we investigate liquidity measures that are capable of assessing liquidity of numerous heterogeneous (but partially substitutable) goods. The process of service selection and matching is, therefore, different and usually more complex than in financial and standardized commodity markets. Additionally, unlike in financial markets where we investigate liquidity of particular stocks and assets, in utility and cloud markets we are interested in the liquidity of the market itself, as many services are substitutable. Furthermore, because of the dynamic cloud market with its changing goods, it will be unthinkable that only one standardized good will emerge. Consequently, it is highly likely that a certain number of standardized and substitutable goods will be available in the market. Therefore, we need novel measures for approximating liquidity of cloud markets (and other electronic markets with similar characteristics) in order to demonstrate the effects of market diversity to market liquidity. For this reason, instead of coping with all of the above-mentioned factors affecting liquidity and building a comprehensive monitoring model for electronic markets, we herewith focus only on the aspects of market liquidity that are directly and unambiguously affected by resource standardization.

One of the key factors of market liquidity is the prices of goods traded in the market: liquidity is strongly affected by buyers' bids and sellers' asks for goods and strongly depends on the market's allocation mechanism as well as the pricing methods used. However, from the perspective of the standardization of goods in electronic markets, we are interested only in the explicit impact of the quantity and structure of the goods on market liquidity. For this reason, the effects that the standardization of services may have on the prices in the market are out of scope of this thesis. To achieve this, we simplify the definition of liquidity and assume a static user behavior in terms of pricing. Namely, we assume that the standardization of goods does not affect the bidding strategies of market participants: they are willing to bid for the standardized goods with the same prices as for the differentiated goods. Note that this assumption would most probably not hold in the real-life markets for several reasons. For example, users' satisfaction with the standardized goods may be lower than with the exact goods they need, which would result in the lower bidding prices. On the other hand, the positive impacts on market liquidity and participation costs (which will be demonstrated in Chapter 7) would have positive impacts on the bidding prices. However, the assumption of "static pricing" provides a simplified view on market liquidity and allows us to avoid uncertainty about the real cause of the change of market prices.

To quantify the impact of the product standardization on market quality, we consider the definitions of the common liquidity measures. Due to the simplification of the assessment model, we are not interested in the bid-ask spread, as it only depends on the current market prices. However, we are interested in the other two standard measures: market depth and immediacy. For the sake of consistency, we refer to these methods as overall *market depth* and *search cost*.

Overall market depth. Similarly as in financial markets, we use market depth to indicate the number of matches between requirements and offers during the trading time. In financial markets, depth points to the trading volume of one asset. In the computing resource market characterized by the heterogeneity of services, however, market depth can be seen as the cumulative value for all goods in the market. To differentiate between these measures, we use the term *overall market depth* to indicate the cumulative trading volume in computing resource markets.

Search cost. In its original definition, immediacy strictly represents the *time* needed to successfully trade a product in the market. It is presented in time units and is defined for every single asset in the market. Although it is a valuable indicator of market liquidity, it is hard to strictly associate it to the variety of resource types in the market, as many factors (e.g., performance of market mechanisms and pricing algorithms, as well as various exogenous factors) may affect immediacy. To avoid these conflicts, we consider immediacy in its broad form: the effort needed to be invested in order to find a trading partner. In our context, this effort describes the number of comparisons between a buyer's (or a seller's) requirement and sellers' (or buyers') offerings in the market until the most suitable service in the market is found. The effort is, hence, associated to the search of a fitting service offering and is, for this reason, termed search cost. The search cost is of a particular importance in those markets in which goods are purchased and (re)sold very often. Considering cloud computing, this is the case in markets in which computational resources are rented on a short term (e.g., Amazon EC2 Spot Instances[1]). Note that the search cost strictly correlates to the immediacy, since a large search cost is always a result of more numerous execution steps in the market, which requires more computation time.

Having the two liquidity measures in mind, we conclude that the goal of increasing market liquidity can be achieved by increasing the overall market depth while reducing the buyers' and sellers' search cost. Due to the inversely proportional relation between these values, this goal can be additionally expressed as maximization of the aggregate liquidity measure

$$lqdt_a = \frac{overall\ market\ depth}{search\ cost}. \tag{4.16}$$

Note that Equation (4.16) does not present a "final" and "unique" measure of liquidity, i.e, a measure that depicts market liquidity independently from the overall market depth and the search cost. The "optimal" market setting is the one in which the search cost is low and the overall market depth high. This is not guaranteed by a high proportion between these values as it is possible that, for example, an extremely low level of search cost compensates for a very low overall market depth. This setting, although it has a high aggregate liquidity, points to a poor market setting. Therefore, all presented liquidity measures are equally important in the sense that they cannot be interpreted autonomously, i.e., without considering other measures simultaneously.

4.6.2 Utility-cost model

The liquidity model presented in the previous section helps us to assess how *tradable* certain (standardized) goods are. In this section, we focus on another perspective of market quality: how satisfied users are with the services and what cost(s) the standardization approach incurs to them. The utility-cost model, which will be presented in this section, has therefore the goal to demonstrate usefulness of the SLA mapping approach by considering how similar (or how different) standardized services are from what the users wish to sell or purchase in the market.

For representing the cost-benefit model of the SLA mapping approach, we define a utility and cost functions. In particular, using the notation defined in Section 4.3.1, we define the utility function u^+ of a user $user$ with respect to the difference of the properties of SLA parameter α of the public SLA templates and the users' private templates. The definition of the utility function follows the distance idea introduced in Section 4.5.2:

[1]http://aws.amazon.com/ec2/spot-instances/, Last accessed: May 2013

$$u^{+}_{user}(\alpha) = \begin{cases} 0, & \chi_{t_{user}}(\alpha) \neq \chi_{t_{new}}(\alpha) \vee \chi_{t_{user}}(\alpha) = \chi_{t_{new}}(\alpha) = false \\ A, & \chi_{t_{user}}(\alpha) = \chi_{t_{new}}(\alpha) = true \wedge \\ & D_{\alpha}^{t_{user}} \neq D_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} \neq F_{\alpha}^{t_{new}} \\ B, & \chi_{t_{user}}(\alpha) = \chi_{t_{new}}(\alpha) = true \wedge \\ & ((D_{\alpha}^{t_{user}} = D_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} \neq F_{\alpha}^{t_{new}}) \vee \\ & (D_{\alpha}^{t_{user}} \neq D_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} = F_{\alpha}^{t_{new}})) \\ C, & \chi_{t_{user}}(\alpha) = \chi_{t_{new}}(\alpha) = true \wedge \\ & D_{\alpha}^{t_{user}} = D_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} = F_{\alpha}^{t_{new}} \end{cases} \tag{4.17}$$

As stated in the definition of the utility function, the user gains utility for an SLA parameter, only if it can be utilized, i.e., if it exists in both their private SLA template and in the new public SLA template. The utility depends on the similarity of the specification of the SLA parameters in the two SLA templates: it is assumed to be A if both the description and the metric of α differ in the user's private SLA template and the public SLA template; B if only one of the parameter properties (i.e., either description or metric) differs; and C if SLA templates do not differ with respect to the parameter.

Although the SLA mapping approach brings many benefits to the market participants, it also incurs cost. The cost is incurred when a public SLA template has been adapted and the user has to define and submit new SLA mappings for the changed parameter properties. The cost function u^{-} for a user $user$ with respect to an SLA parameter α is defined as follows:

$$u^{-}_{user}(\alpha) = \begin{cases} 0, & \chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = false \vee \chi_{t_{user}}(\alpha) = false \\ 0, & \chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = \chi_{t_{user}}(\alpha) = true \wedge \\ & \big(D_{\alpha}^{t_{user}} = D_{\alpha}^{t_{new}} \vee D_{\alpha}^{t_{user}} = D_{\alpha}^{t_{init}}\big) \wedge \\ & \big(F_{\alpha}^{t_{user}} = F_{\alpha}^{t_{new}} \vee F_{\alpha}^{t_{user}} = F_{\alpha}^{t_{init}}\big) \\ 0, & \chi_{t_{new}}(\alpha) = \chi_{t_{user}}(\alpha) = true \wedge \chi_{t_{init}}(\alpha) = false \wedge \\ & D_{\alpha}^{t_{user}} = D_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} = F_{\alpha}^{t_{new}} \\ D, & \chi_{t_{new}}(\alpha) = \chi_{t_{init}}(\alpha) = \chi_{t_{user}}(\alpha) = true \wedge \\ & \big((D_{\alpha}^{t_{user}} \neq D_{\alpha}^{t_{new}} \wedge D_{\alpha}^{t_{user}} \neq D_{\alpha}^{t_{init}}) \vee \\ & (F_{\alpha}^{t_{user}} \neq F_{\alpha}^{t_{new}} \wedge F_{\alpha}^{t_{user}} \neq F_{\alpha}^{t_{init}})\big) \\ D, & \chi_{t_{new}}(\alpha) = \chi_{t_{user}}(\alpha) = true \wedge \chi_{t_{init}}(\alpha) = false \wedge \\ & \big(D_{\alpha}^{t_{user}} \neq D_{\alpha}^{t_{new}} \vee F_{\alpha}^{t_{user}} \neq F_{\alpha}^{t_{new}}\big) \end{cases} \tag{4.18}$$

The cost function specifies the cost incurred by the necessity of creating new SLA mappings. A user has no cost for an SLA parameter if: (1) the parameter does not exist in the private SLA template or in neither the initial (current) SLA public template nor the new public SLA template; (2) the parameter properties of the user's private SLA template are the same as in the new public SLA template or as in the initial (current) public SLA template. In both cases, the SLA mappings have already been created; (3) the parameter was added to the new SLA template (i.e., the SLA parameter did not exist in the initial public SLA template), but its properties are the same as in the user's private SLA template. The user faces the cost D if he must create SLA mappings for the parameter. This is the case if: (1) the parameter properties are different in the user's private SLA template and in the public SLA template; or (2) the parameter was added in the new public SLA template, but with the properties differing from those of the user's private SLA template.

Note, with the support of automated mapping, the cost of adaptation for the user would become quite low (Section 4.5.4). The user would only face cost when joining the system and when the mapping directory does not contain the mapping needed. However, independently of whether the cost is carried by the user or by the marketplace, the cost needs to be considered for objectively evaluating the clustering and adaptation approach.

We define the overall utility U^+ and the overall cost U^- for all users C as:

$$U^+ = \sum_{user \in U} \sum_{\alpha \in P} u^+_{user}(\alpha), \quad U^- = \sum_{user \in U} \sum_{\alpha \in P} u^-_{user}(\alpha)$$

where P is the set of all SLA parameters occurring in T_{user}, T_{init} or T_{new}. The overall net utility U^O is then calculated as

$$U^O = U^+ - U^-. \tag{4.19}$$

The return values of the utility function and the cost function are not strictly defined. However, considering the type of efforts and benefits, it should hold that $0 \leq A \leq B \leq C$ and $0 \leq D$. For our simulations, we fix these return values at $A = 1$, $B = 2$, $C = 3$ and $D = 1$.

CHAPTER 5

Automated service discovery

Although most of the existing marketplaces use SLAs to express and negotiate user requirements and offers for services, there exists no general standard for their specification. Therefore, although syntax specification of some requirements may vary among different SLAs, their semantic meaning may be the same from the perspective of market participants. As discussed in the previous chapter, we use the SLA mapping approach to bridge these differences and allow using SLAs specified in different languages, standards, and properties. Nevertheless, creating and managing SLA mappings is expensive: users must manually find differences in SLA specifications and create (often very complex) mappings before trading in the market. Even after creating the mappings to the index SLA template, users must invest a significant effort to search through all service offerings in the market and discovering those that may be matched with the user's requirements.

A high cost for entering the market and finding a matching trading partner (i.e., market participation and transaction costs) may prevent users to join the market, thus causing a market failure. Therefore, in order to ensure market well-being, we must enhance and facilitate the process of creating SLA mappings and selecting matching services.

Besides improving market performance, facilitated SLA matching presents a step towards the future of cloud markets: automatic service discovery and selection. Having these abilities, consumers' systems and tools may automatically recognize that they need additional computing resources and, again automatically, purchase a service in a market without requiring any form of human interaction. This form of automatic service discovery and selection is one of the main prerequisites of ubiquitous computing as a utility.

In this chapter, we address Research question 4 specified in Chapter 1 by assessing the cost of creating SLA mappings and matching service specifications with the SLA mapping technique and discussing the methods for reducing this cost in cloud markets. In particular, we present an approach for automatic discovery of *semantically equal* SLA elements and creation of SLA mappings that map the differences in their syntax specification. Furthermore, using the automatic SLA matching algorithms, we allow the autonomic provider selection in cloud computing marketplaces.

In order to achieve our goals, we propose storing and managing history data of user requirements as well as of mappings between corresponding requirements in a market knowledge repository. We suggest machine learning and automatic reasoning methods to analyze this data and automatic matching of SLA elements and generation of mappings based on the knowledge about common requirement specifications gained through this process. As the result, our approach facilitates automatic recommendations of trading partners and SLA mappings, thus reducing the cost of joining the market.

The remainder of the chapter is organized as follows. Section 5.1 details the motivation for this work and shortly summarizes the methodology of our approach. Section 5.2 describes methods for discovering equal elements of differing SLAs and discusses automatic creation of SLA mappings. Section 5.3 continues this discussion and presents how matching SLA templates can be discovered using the derived notion of SLA element equality. Finally, Section 5.4 presents the cost model for the evaluation of the approach.

5.1 Background

As described in the previous chapter, the standardized goods market differs from the differentiated goods market in the sense that the market users do not search for service offerings of other users, but "associate" their private SLA templates with public SLA templates that are closest to their requirements. Therefore, for the successful establishment of the contract between service buyers and sellers, their requirements must be matched with the ones specified in the selected public SLA template. Since there exists no general standard for specifying requirements in SLA templates, their definition may vary among templates. For that reason, we use the SLA mappings. The main purpose of SLA mappings is to bridge the differences between *syntax inequalities* of specific SLA elements that are *semantically equal* in two SLA templates. Semantically equal SLA elements have the same meaning from the perspective of the market participants involved in a trade. On the other hand, SLA elements are equal by syntax if their language specification is identical. Note that in our vision of cloud markets, syntax equality of SLA elements implies their semantic equality.

Market participants are asked to create SLA mappings between their own private SLA template and the index SLA template when entering the market, which is a key requirement for the successful management of market participants' requirements and for regulating the market. As already discussed, mapping translations can range from very simple, such as incompatible names of SLA elements (e.g., different names used for the same SLA parameter) to complex translations, such as methods for calculating parameter values with different metrics (e.g., different units used to express the same parameter value).

When specifying SLA mappings between elements of two SLA templates, market participants need to assess the equality of both elements from their own perspective. Contradictions caused by different opinions on the semantic equivalence of SLA elements play a subordinate role in autonomic market management since the market may be regulated according to their meaning for the majority of market participants. Only during the negation of final contracts such contradictions may be detected and solved by mutual agreements between two trading parties.

In order to enhance the SLA management techniques in cloud markets by automatically matching (elements of) SLAs and SLA templates, it is first necessary to extract the knowledge from specifications of user requirement for services (i.e., private SLA templates). Note that this is only possible if SLA templates and SLA mappings are managed by the cloud market platform. Specifications of requirements and their variations over different SLA templates can then be analyzed and generalized knowledge can be learned and reused for generation of SLA mappings between unknown SLA templates. To realize this goal, a cloud market should implement additional knowledge mechanisms dealing with several important tasks such as (1) storing and managing data about the history of SLA templates and mappings, (2) learning the "semantics" necessary for finding equivalent SLA elements and generation of adequate SLA mappings, and (3) automatic SLA element matching and generation of SLA mappings between SLA elements. The knowledge that is gathered through this process is finally used for two major tasks: **automatic SLA matching**, i.e., recognition of equal elements of two SLAs and automated creation of SLA mappings between them, and **automatic provider selection**, i.e., recommendation of those public SLA templates to the market participants that are the closest to the requirements specified by their private SLA templates. In the following, we explain the processes of achieving these goals.

5.2 Automatic matching of SLA elements

The process of matching semantically equal SLA elements from differing SLA templates is important for enabling automatic provider selection in cloud markets. This process is executed in three steps. First, for each pair of two SLA elements, the probability for their equivalence (i.e., their similarity) is computed. Then, based on the computed probabilities, the equivalence of the elements is determined. Finally, if two SLA elements are semantically equal, the SLA mappings are automatically created to bridge the possible differences. When measuring the similarity of a pair of SLA elements, we consider their two properties: element description, stating basic properties such as name and measurement unit, and element metric, describing a method for measuring the element value.

5.2.1 Similarity of element descriptions

Properties of element descriptions are represented by string values. For calculating the similarity of these properties, possible characteristics of their specifications have to be considered. The most common characteristics we have identified are variations in the order of single characters or character blocks (e.g., "Memory Consumption" vs. "Consumption Memory"), additional characters or words (e.g., "Consumption Memory" vs. "Consumption of Memory"), and usage of abbreviations or synonyms (e.g., "Memory Consumption" vs. "Memory Usage").

Calculation of the similarity between description properties can be realized by utilizing string similarity metrics. A large number of these metrics has been discussed in existing research [21, 71, 137]. They may be roughly separated into two groups: character-based and vector-space based methods [22]. While the former rely on character edit operations (e.g. insertions, deletions, and substitutions), the latter "transform strings into vector representations on which similarity computations are conducted" [22]. In this thesis, we use the character-based string similarity metric *Levenshtein distance* for this purpose due to its good performance with short strings that only contain a small number of variations in their syntax. Levenshtein distance is defined as "the minimum number of insertions, deletions or substitutions necessary to transform one string into another" [22]. In our model, the similarity of two strings is computed as the difference between the length of the longer of the two strings and the number of character modifications needed to convert one string to another (i.e., the Levenshtein distance) divided by the same maximum distance:

$$p_{equal}(S_1, S_2) = \frac{max(|S_1|, |S_2|) - d_L(S_1, S_2)}{max(|S_1|, |S_2|)} \tag{5.1}$$

where S_1 and S_2 are the two strings, $|S_1|$ and $|S_2|$ their lengths, and $d_L(S_1, S_2)$ their Levenshtein distance.

Although string similarity metrics promise good results for identification of small variations in description properties, they are not able to recognize semantically equal properties that differ significantly, as it is the case of synonyms or abbreviations. These differences may only be detected by having concrete knowledge of possible synonyms or abbreviations. Hence, besides string similarity, our approach utilizes Case-Based Reasoning (CBR) methods for learning semantically equal description properties from established SLA mappings. In particular, semantic equivalence of individual description properties is determined in four phases:

1. retrieval of the most similar case to the new case by calculating the similarity between SLA element description properties used in both cases and selecting the case with the highest similarity;

2. reuse of the knowledge of the similar case to solve the new problem;

3. revision of the proposed solution by analyzing users' feedback to the recommendation of the matching SLA elements; and

4. retainment of the parts of user experience likely to be useful for future problem solving (i.e., storing users' feedback as part of the utility of a case).

If reoccurring patterns are detected by CBR, the similarity of two element description properties is calculated based on the similarity between the old case and the new case, which is again computed using the Levenshtein similarity metric.

Figure 5.1 illustrates how an existing CBR case could be reused for solving a similar new case (the symbol $\cong^S$ is used to denote *semantic equivalence* of two SLA elements): While two cases describe SLA metrics for significantly different purposes (namely for measuring *memory usage* and *bandwidth usage* of a cloud service), their similarity lies in the unit conversion from "Gbit" to "Mbit" and vice versa. Therefore, an SLA mapping with an associated translation function specified in the existing case can be reused in the new case to capture the unit conversion.

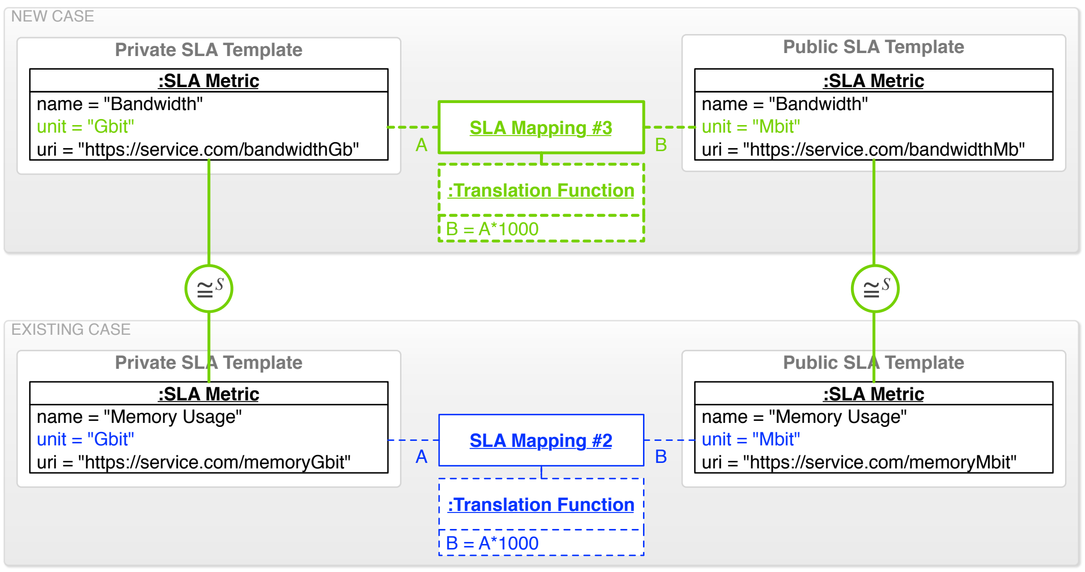


Figure 5.1: Reusing a CBR case

5.2.2 Similarity of element metrics

Element metrics describe the methods for measuring SLA element values. They can either directly specify a measurement source by stating its URI or indirectly reference and/or aggregate other SLA elements for retrieving their measurements. Composite aggregation of measurements is specified as a function of the metrics.

On the one hand, similarity of element metrics stating measurement sources is done analogously to element description properties, since URIs may abstractly be seen as strings. Similarity of properties that reference individual SLA elements should only ensure equality of the referenced elements. On the other hand, computing similarity of the metric properties defined by functions, i.e., combinations of several QoS measurements, involves determination of the equality of the functions itself. Namely, due to the possible variations and transformations inside the functions (e.g., introduction of substitutions, change of the variable orders, utilization of different units, etc.), this process must involve several steps:

1. checking if all referenced SLA elements used in both functions are semantically equal;
2. ensuring that the both functions use the same variable assignment (i.e., two semantically equal SLA elements are assigned to the same variable); and
3. proving the logical equivalence of both formulas using an algebra solver.

If one of these steps fails, functions are considered as non-equivalent. Otherwise, the elements are considered equivalent.

As an example, one might think of two semantically equivalent SLA metrics measuring the “error rate” (er) of transactions: While in the first case the “error rate” is calculated by setting the "number of erroneous transactions" (et) in relation to the "total number of transactions" (tt) ($er = et/tt$), in the second case the "error rate" is calculated by relating the "number of correct transactions" (ct) to the "number of total transactions" and inverting the resulting value ($er = 1 - ct/tt$). The "number of correct transactions" is itself managed by a third, separate SLA metric and calculated by subtracting the "number of erroneous transactions" from the "number of total transactions" ($ct = tt - et$). Following the above steps, determining semantic equality of both SLA metrics that measure the "error rate" involves the following reasoning chain:

1. All referenced SLA elements used in the metrics' functions are semantically equivalent (for the "total number of transactions" this is obvious; for the "number of correct transactions" it can be proven that the value can be calculated out of the "number of total transactions" and the "number of erroneous transactions", as defined in the third SLA metric: $ct = tt - et$).

2. both metric functions use the same variable assignment (i.e., semantically equivalent SLA elements referenced in both functions are described by the same variable names: $er = er$, $ct = ct$, and $tt = tt$).

3. Both formulas are logically equivalent (i.e. $et/tt = 1 - ct/tt$).

Thus, we could automatically prove semantic equivalence of both SLA metrics that calculate the "error rate" using different (nested) functions. In our approach, logical equivalence of metric functions is checked by Symja [195], a Java-based algebra solver library that facilitates parsing algebraic expressions, automatically detecting variables in expressions, and solving not only numeric, but also symbolic expressions (i.e., expressions containing abstract variables), which is of great importance in our context.

While the similarity of element description properties and element metrics stating measurement sources is expressed as a continuous probability, the one of element metrics defined by functions can only be expressed by a binary classification since two functions can either be logically equal or not. This is expressed as a discreet value, i.e., 0 (non-equivalence) or 1 (equivalence) of a pair of functions.

5.2.3 Final decision on element similarity

To make a final decision on semantic equality of SLA elements, the similarities of their individual properties have to be combined to an overall value representing the probability of element equality. For this purpose, we utilize machine learning methods. As input features for classification, we use the equality probabilities of the individual element properties that are precalculated using the string similarity measurements, CBR-enhanced similarity detection, and algebra solver. Classification is done separately for each SLA element in the hierarchical order (i.e., from the SLA metric level up to the SLO level). In our evaluation, we utilize the Support Vector Machines (SVM) method for implementing the classification strategies for the individual SLA elements.

SVM is a concept in statistics and computer science for a set of related supervised learning methods that analyze data and recognize patterns in such way that they take a set of input data and predict, for each given input, which of two possible classes forms the input. In our scenario, SVM uses a training phase in which it builds an internal model for prediction of the final binary classifier for the semantic equality of SLA elements. The internal SVM model uses the equality probabilities of the individual SLA element properties as the input features and produces a binary output with *semantically equal* and *semantically different* as possibly classes for the given input. Training of the SVMs requires creation of training examples by retrieving pairs of SLA elements from a private and a public SLA template that have been associated by a market participant, determining the semantic equality of each pair based on the market participants feedback to an automatic recommendation of SLA mappings, and calculating the similarity probabilities between corresponding properties of both SLA elements. The similarity probabilities are then forwarded to the SVM as input features while the semantic equality (expressed as a discreet value, where 0 stands for non-equality and 1 for equality of both SLA elements) is used as the target output feature.

5.2.4 Automatic generation of SLA mappings

Once the equal SLA elements from two different SLA templates have been matched, the SLA mappings must be created to bridge the discovered differences. SLA mappings are automatically created and recommended to the user.

Generation of SLA mappings for incompatible element description properties is straightforward, since they only map already identified differences in string values. On the other hand, generation of more complex mappings between incompatible element metrics involves creation of suitable translation functions, which can later be used to translate measurements defined by one SLA metric into the other and vice versa. In the simplest case, two semantically equal SLA metrics represent the same measurement defined in the same unit, but differ in their structural specification. In this situation, it is enough to create an SLA mapping referencing the SLA metrics and defining a translation function for converting a measurement from one SLA template into another. In more complex cases, two semantically equal SLA metrics additionally differ in their units. In this case, generation of the translation function involves knowledge about how the units of both metrics are mathematically related. This knowledge can be obtained from the previously used SLA mappings by utilizing CBR methods. If an equivalent unit translation has been used in a similar case, knowledge about specification of the translation function can be transferred to the new case. If a similar case does not exist, the user must manually define the mapping.

After receiving the recommended SLA mappings, a user submits his feedback to the recommendation component stating the correctness of the recommended data. In case of a negative feedback, a user can state whether the SLA mapping was incorrect or an SLA element was wrongly matched. This data is then used for the learning process of the CBR and SVM methods. Note that dealing with the contradictory user feedback is out of the scope of this thesis.

5.3 Automatic SLA selection

Automatic SLA selection assumes assessing semantic equality of two SLA templates and requires matching of equivalent SLA elements from those templates. Given a user's private SLA template and a set of public SLA templates, the matching process starts by iterating through all public SLA templates and checking the similarity of all SLA elements contained by the user's private SLA template and the currently examined public SLA template. For each pair of SLA templates, the overall equivalence probability is computed as the relative number of matched SLA elements in the total number of elements of both SLA templates. Finally, the public SLA template with the highest equivalence probability is chosen as the optimal offering, but only if its equivalence probability exceeds a predefined threshold, usually in our work set to 70%. Otherwise, a user is notified that the appropriate service offering currently does not exist in the market.

5.4 The cost model

When initially joining a cloud market, users need to execute several manual steps to find an adequate cloud service and establish a contract for its usage. Manual execution of these steps involves a high effort for market participants. In this section, we build a simple cost model to quantitatively assess this effort. This model will then be used in our further discussions on the benefits of the approach presented in this chapter.

When joining the market, market participants need to search and select an adequate provider service offering that best matches their own requirements. Since in our vision cloud services traded on the market are described by public SLA templates, market participants must associate their private SLA template with the best-matching public SLA template. In our cost model, we mark the cost of manually searching for the best-matching public SLA template with the variable a. This cost is common for the traditional cloud markets in which SLA templates are not automatically recommended to the participants, but also when a template has been recommended that is not the best-matching for the user. On the other hand, a market participant has no cost if he must not manually search for an appropriate public SLA template, which is in case when a public SLA template is automatically and correctly recommended to the user.

After selecting an adequate provider, market participants need to specify SLA mappings between their private SLA template and the chosen public SLA template to bridge any possible syntax differences. In our cost model, we mark the cost of manually creating a new SLA mapping with the variable c. On the other hand, a user has no cost if he must not create an SLA

Table 5.1: The cost model

Cost	**Cost description**	**Value**
a	Manual association of a public SLA template	10
c	Manual creation of a new SLA mapping	10
d	Identification of an incorrectly recommended mapping	10

mapping, which is in the case when a mapping is not necessary or the correct SLA mapping was automatically provided to the user. Since the recommended SLA mappings may also be incorrect, a user may have an additional cost d to delete the recommendation, i.e., to send a negative feedback. Note that in some cases a user may have to both delete the recommendation and create a new SLA mapping, which results in the total cost of $c + d$.

The presented cost model is in our simulations used to compare the benefits of automatically creating SLA templates and SLA mappings in cloud markets. The values of the variables a, c and d used in our simulations are depicted in Table 5.1. Note that, considering the presented model, the absolute values of the costs are less important than their relative difference.

CHAPTER 6

Agent-based market simulator

In order to test our hypotheses and methods presented in the previous chapters, we have extended two well-known simulation platforms: GridSim [41, 45, 194] and VieSLAf [30, 31]. GridSim is an open-source toolkit for conducting simulations in grid environments, including market-based service allocation. As described in Chapter 3, we have extended GridSim to (partly) support our idea of an autonomic cloud market by extending it with our monitoring sensors. On the other hand, we have used the VieSLAf tool to demonstrate our idea of cloud resource standardization, as described in Chapter 4. VieSLAf is a tool for semi-automatic management of SLA templates and SLA mappings and is, therefore, a suitable environment for the simulation of our approach. Nevertheless, these tools have proven to be unsuccessful in capturing our idea of self-aware marketplaces. For this reason, we address Research question 5 specified in Chapter 1 and implement our market simulator, which we present in this chapter.

This chapter is organized as follows. In Section 6.1, we describe the benefits and the shortcomings of GridSim and VieSLAf with respect to the implementation of the autonomic cloud marketplace and outline the motivation for building a new framework. In Section 6.2, we present a conceptualization of a simulator for autonomic markets. Finally, in Section 6.3, we discuss the implementation of our agent-based cloud market simulator.

6.1 Motivation

By reusing and extending existing solutions and implementations of frameworks, we contribute to the evolution of a single tool instead of contributing to the creating a chaotic plethora of simulation frameworks. Furthermore, reusing existing tools allows us to focus on the novelties and simply demonstrating that our approaches are easily implementable in existing environments. The reason why we chose the two tools in our work - VieSLAf and GridSim - is because they seemed to fit well to our needs. On the one hand, VieSLAf has been developed for the lifecycle management of adaptable SLAs and SLA templates. It initially supports the SLA mapping technique and is easily extendable with our methods for liquidity prone resource adaptation and automatic SLA matching and selection. On the other hand, GridSim is a well-known open-source tool that initially supports numerous reservation-based and auction mechanisms for resource allocations and provides well-defined interfaces for the implementation of the additional mechanisms and algorithms.

Despite offering a good support and implementation of a market framework, integration and mutual communication between VieSLAf and GridSim is hardly achievable as they significantly differ in their implementations, interfaces, and input/output data. However, even if this was possible, these tools demonstrate numerous shortcomings when it comes to the implementation of a self-aware cloud marketplace. For example, VieSLAf demonstrated immense problems with scalability. Namely, with the growth of the user base and the number of SLA templates, the simulation time increased exponentially. Applying elemental methods for resource adaptation (e.g., clustering and SLA distance calculation) was, therefore, extremely inefficient and costly. VieSLAf also demonstrated numerous stability problems. Another issue is that it was built to work in the Microsoft .NET environment and was, therefore, not able to be run on our Ubuntu-based evaluation cluster. VieSLAf also demonstrated numerous problems with integration with another tools (including GridSim), as it does not provide any communication interfaces nor it allows a simple implementation of one. Finally, the data model of VieSLAf is very complex and its extension is very hard and costly.

Regarding GridSim, despite its large flexibility, numerous interfaces and multi-layered architecture, creating a simulation scenario is not a trivial task, as many market actions and the creation of communication objects between the layers of GridSim are left to the user. However, GridSim does provide a small set of examples that illustrate the implementation of simple trading scenarios. In our feasibility study, we applied one of the example scenarios. This example allowed us to control basic trading properties, i.e., the number of buyers and sellers in the market and the number of requests per buyer, etc., which for our purposes was adequate. It also enabled the construction of a market, establishment of participants and resources, and provided an easy platform upon which to implement a monitoring framework. It was also straightforward to implement a basic benchmark scenario to test the monitoring framework. However, we encountered difficulties when we created more realistic and elaborate scenarios, for example: different participant types (e.g., malicious users, market makers, speculators, monopolists, and other strategic behaviors); more complicated trades, i.e., multiple resource entities; dynamic context: adding or removing participants or resource types at runtime; and engineering aspects like market growth or contraction. When trying to create such scenarios, we encountered runtime exceptions for the following reasons: (1) GridSim expects the number of users to remain fixed; (2) it is not possible to change the quantity of resources that sellers offer and buyers request, i.e., total supply and demand is predefined; (3) new resource types cannot be added at runtime; and (4) it is not possible to manage the timings of the bid/ask submissions, this is controlled by GridSim's event handlers, which makes it difficult to implement users with specific participation strategies. Through our efforts to counter these as well as other challenges, we were moving away from GridSim's initial use case, eventually making it impossible to control and extend further. Consequently, we were no longer confident that changes in the market were engineered by us as opposed to errors in the GridSim runtime. It would be easy to say that this is a failing of GridSim, but our scenarios were straying outside of GridSim's scope.

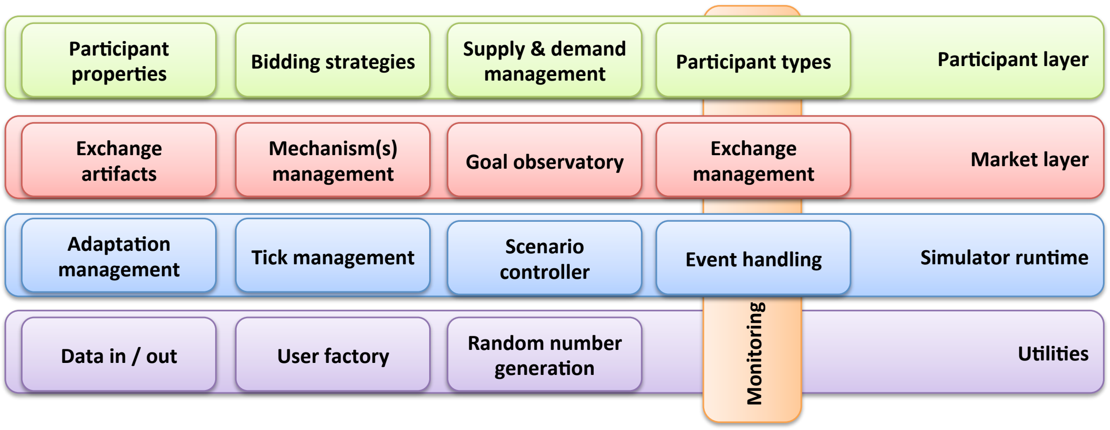


Figure 6.1: Conceptual framework of a simulation environment for autonomic markets

6.2 Conceptualization of a simulator for autonomic markets

Based upon our experiences with VieSLAf and GridSim, we realized that trying to simulate different aspects for the study of autonomic markets needs a more flexible simulation approach. In Chapter 2, we discussed some alternatives to GridSim, but failed to see the ability to capture all aspects that we feel are necessary without significant effort in the extension of an approach. In this section, we define the key features that are required and propose a conceptual architecture for an autonomic market simulator that will act as a testing environment for the future studies. Figure 6.1 illustrates the layers and components of our proposed simulation architecture, which are as follows.

The Monitoring Framework captures key information on the market platform through links to the Participant, Market and Simulator layers, and makes this information available to the components that require it (e.g., the Goal Observatory). Monitoring information here captures the state of: mechanisms, the market in general and the computational infrastructure.

The Participant Layer captures the aspects necessary to represent market participants as well as their various nuances and differentiating factors. The key component is *participant type*, which identifies whether a participant is a consumer, provider, prosumer, or broker. It also enables different participant flavors like market makers, speculators, monopolists, aggressive and passive participants. In accordance to the typical market simulators, we define *bidding strategies*, as well as the management of *supply and demand*. We use the word “management” to illustrate that this is not a statically defined process, but entails stochastic and dynamic behaviors such as participants joining or leaving the market, as well changes in their individual properties and requirements over time. *Participant properties* capture additional information needed for each participant type, e.g., range of wealth, resource types offered/desired, etc.

The Market Layer defines the components to implement an electronic market. This includes: the *artifacts* to be traded, including their type, quantity and period of availability or desirability; different allocation *mechanisms* like the English or Continuous Double Auction, but also the means to create custom mechanisms and have multiple active mechanisms. Mechanism management here refers to the programming constructs to transparently include any arbitrary mechanism by exposing a standard interface. A mechanism manager controls how bids and asks are passed to a mechanism and when instances are created and destroyed; a *Goal Observatory* for defining goals and keeping track of their adherence via the monitoring framework; an *exchange management* that keeps track of all incoming asks and bids, matches, as well as one or more active mechanisms; and finally, *adaptation management* as an instantiation of a market adaptation component.

The Simulator Layer is the basis for the simulator. It includes a singleton *event handler*, as this enables a simple programming model without the need for complex thread or concurrency management, and a *tick manager* to control "time" in the simulator as a sequence of discrete periods. In each tick, we invoke participants in a renewed random order, and give them the option to "act", i.e., do something in the market. We also define a *scenario controller*, which through the event handler can instigate new scenarios for observation, based upon the current time. The scenario controller permits us to create issues of instability or change specific settings in the market to study how the market changes, and later how adaptation actions have improved or worsened the situation. We can also layer (simple) scenarios to create more complex compound scenarios.

The Key Utilities Layer assists in market simulation and includes: readers for trace data from existing markets to "stimulate" market events or scenarios as well as writers to store monitoring data; a *participant factory* to facilitate the generation of multiple participant types based on a set of input parameters; and as a key premise for all simulators, a *random number generator* which can simply be the inclusion of the Colt Library[1] or similar.

6.3 Implementation details

Based on the conceptual model defined in the previous section and having in mind the process of dynamically trading adaptive cloud resources, we have developed a multi-component market simulation framework. The architectural representation of this framework is depicted in Figure 6.2. In the following, we outline responsibilities of each of the framework components depicted in this figure. Note that each of the components is implemented in Java programming language.

6.3.1 Market directory

Market directory is the elemental component of the market platform as it holds information and data required for the proper functioning of all remaining market components and is, therefore,

[1]Colt Library, http://acs.lbl.gov/software/colt/, Last accessed: May 2013

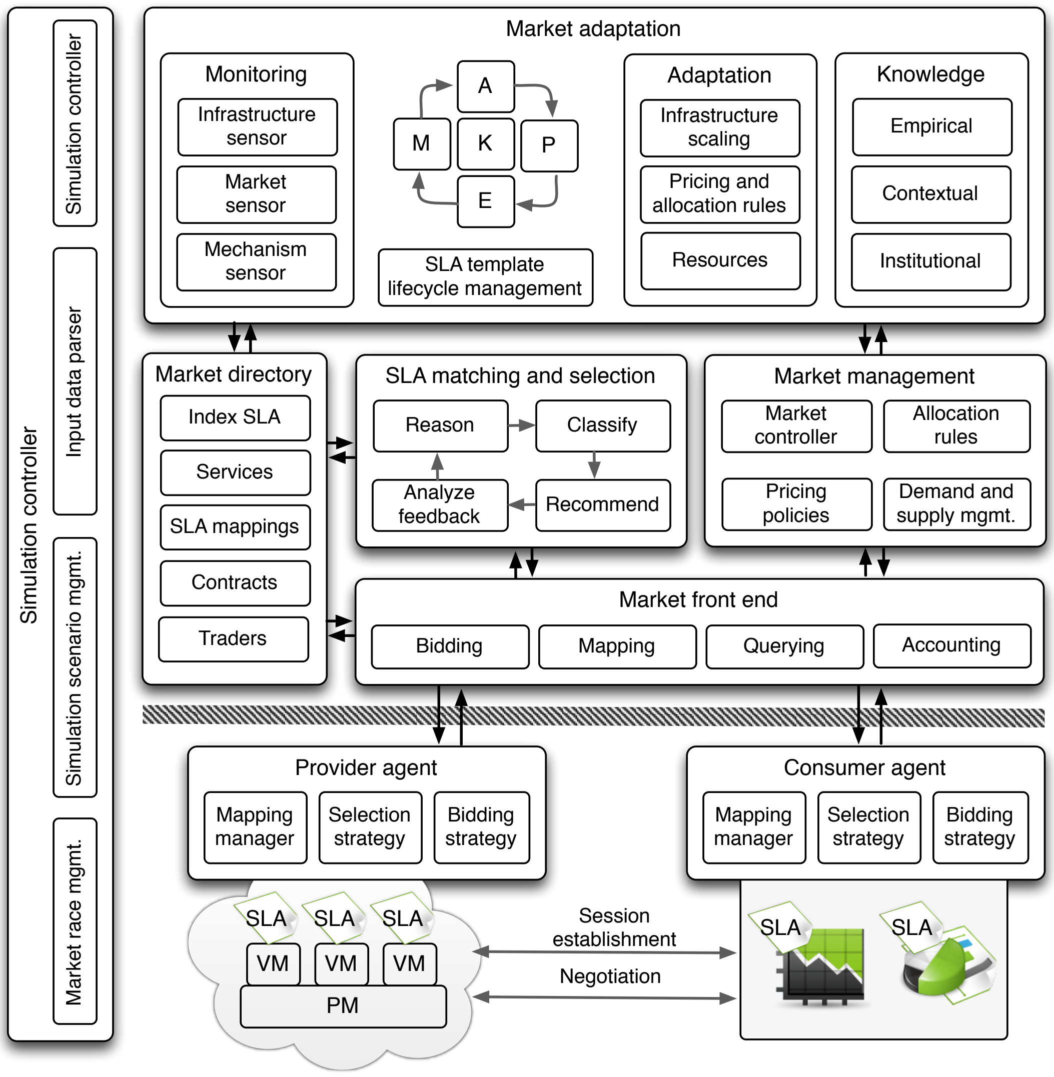


Figure 6.2: Architecture overview of our market simulator

needed for the execution and management of the market. Information stored in the market directory includes the index SLA template, specifications of (standardized) resource types that are to be traded in the market (i.e., public SLA templates), information about the traders (incl. submitted requirements and offers in forms of private SLA templates), and SLA mappings of all market participants that have been submitted to bridge the differences between traders' private SLA templates and the public SLA templates.

Market directory is accessed continuously and excessively and can, therefore, easily be a bottleneck of the market platform. In order to prevent this, market directory must, among others, satisfy the following conditions: (1) as the cloud environment is unpredictable and dynamic, it must be closely monitored and managed; (2) in order to ensure efficient deployment on the dynamic cloud environment, it must run in a replicable setup (regardless of its size); (3) it must ensure scalability, as cloud applications are often characterized by fluctuating demand that can easily spike at any moment; (4) it must ensure high availability, which often depends on the availability of resources and the ability to dynamically provision them on the fly once a failure is identified; and (5) it must support essential attributes of cloud computing, such as multitenancy.

As our simulations have been run in the controlled environment, we have implemented market directory using an SQL-based database. However, in the real-world market platforms, market directory should be implemented using a more efficient and scalable framework such as Hadoop[2], Couchbase Server[3], Cassandra[4], or MongoDB[5].

6.3.2 Market adaptation

Market adaptation is a market layer responsible for the market's self-awareness. In particular, this layer comprises methods and tools for market monitoring, market (and service) lifecycle management, adaptation methods for scaling in or out the resources and infrastructures, and managing the knowledge needed to achieve this process. This layer represents the core of the market adaptation processes described in Chapters 3 and 4.

As discussed in these chapters, market adaptation is implemented as an autonomic MAPE-K loop. During the **(M)**onitoring phase, market properties are assessed. Decision on which properties should be monitored is made with respect to the predefined market performance goals (e.g., maintaining a certain level of market liquidity and/or volatility). During the **(A)**nalysis phase, the "low level" monitoring data is aggregated to these "high level" market goals and validated against the constraints, i.e., the previously mentioned market performance goals. Based on the results, a set of actions is derived that need to be performed in order to improve market performance. During the **(P)**lanning phase, the execution of the action steps is planned (i.e., the actions are mapped to step-by-step execution directives specific for the given market setting). Finally, during the **(E)**xecution phase, these directives are executed.

Each of the autonomic loop phases is implemented as a separate component within the market adaptation layer. These components are:

Monitoring sensors. Similar to our extensions to GridSim, which were introduced in Chapter 3, monitoring component of our market simulator comprises three sensors. The *infrastructure sensor* observes the infrastructure properties of the market (e.g., CPU performance and memory utilization). The *market sensor* observes market-based properties that are not related to the allocation and pricing mechanisms (e.g., number of active service providers, consumers, and standardized services). Finally, the *mechanism sensors* observes the information related to the efficiency of a market mechanism (e.g., market liquidity, size of an order book, and the average matching price). In our simulator, infrastructure monitoring is performed using the interfaces defined by the `java.lang.Management` package,

[2]Amazon Hadoop, http://hadoop.apache.org/, Last accessed: May 2013
[3]Couchbase Server, http://www.couchbase.com/couchbase-server/, Last accessed: May 2013
[4]The Apache Cassandra Project, http://cassandra.apache.org/, Last accessed: May 2013
[5]MongoDB, http://www.mongodb.org/, Last accessed: May 2013

which is a management interface for monitoring and management of the Java virtual machine as well as the host operating system. The remaining two monitoring sensors are developed using the metric-mapping technique as described in Chapter 3.

Adaptation tools. We differentiate three forms of market adaptation: (1) scaling up or down the market infrastructure, (2) adapting or, in extreme situations, replacing market mechanisms, which mostly includes modification of pricing and allocation rules, and (3) adaptation and management of cloud resources (i.e., standardized services). Finding which of these options is the most fitting is not trivial. Autonomic adaptation of infrastructure properties has already been discussed in a large body of literature, as detailed in Chapter 2. This, however, is not the case for the *institutional adaptation*, i.e., modification of market mechanisms. To facilitate institutional adaptation, we need to understand what different market configurations mean for the fulfillment of a given set of goals, which can be achieved through simulation to enable the analysis of what-if scenarios to determine and assess adaptation options. In our current version of the market simulator, the execution steps of the first two types of market adaptations are not fully implemented, but the simulator is designed in a form that this can be easily achieved once the adaptation methods are available. On the contrary, the last type of market adaptation - resource standardization - is fully implemented as discussed in Chapter 4. This means that the adaptation component tools apply clustering algorithms and adaptation methods to create and manage standardized services.

Knowledge manager (KM). The KM component serves as a mechanism for storing, analyzing and managing knowledge for reasoning on market performance and adaptation. For example, it holds information about the goals that the market should achieve (e.g., keeping a certain level of market liquidity and/or volatility) and how these goals can be assessed (i.e., mappings from the low-level monitoring metrics to the high-level market goals, as described in Chapter 3). Additionally, the KM component holds the knowledge on market adaptation rules, i.e., set of actions that can be performed in the market (e.g., the adaptation of standardized resources) in order to achieve these goals. Among other, this includes the statement on which clustering algorithms, adaptation methods, and SLA similarity prediction functions should be used in this process. Finally, this component also holds the knowledge necessary for the automatic SLA matching and service selection (e.g., relations between semantically equal, but syntactically differing SLA parameters), which is performed by the SLA matching and selection tool. The knowledge stored in the KM component follows the classification introduced in Chapter 3 and can, therefore, be empirical, contextual or institutional.

6.3.3 SLA matching and selection

The SLA matching and selection tool offers a support to service providers and service consumers when entering the market by offering recommendations of the SLA mappings to the index SLA template and recommendations of the best-fitting public SLA templates. The goal of this component is to reduce the cost of market participation for market participants. For the automatic reasoning on SLA similarity, this tool applies the methods described in Chapter 5. These methods include various learning techniques, algebra solvers, reasoning techniques, and classification tools that improve their efficiency based on the users' feedback on recommendation correctness. The process of applying these methods is implemented as a loop with the following phases. First, based on the existing knowledge and previous cases, the tool reasons about the equality of each SLA parameter separately from the currently observed private SLA template and the index SLA template. In case a difference in the specifications of the parameters exist, the tool creates the necessary SLA mappings. In the second step, the tools applies classification algorithm on the information on SLA parameter equality to select the best-fitting public SLA template. In the third step, the best public SLA template and the SLA mappings are recommended to the user. Finally, in the last phase, users' feedback is analyzed and the knowledge updated.

6.3.4 Market management

Market management is a market layer that comprises main marketplace mechanisms for trading resources in the market. This layer includes implementations of market operations such as allocation mechanisms, pricing schemes, and methods for controlling the trade. Currently, the simulator implements several well-known clearing (i.e., allocation) mechanisms: Continuous Double, English, Japanese, and Vickrey auctions. The simulator also offers several pricing methods, such as taking the arithmetic average between the provider's offered price and the consumer's requested price, and taking the nth highest requested (bid) price. More importantly, the simulator offers an extendable layered architecture and a well-defined set of interfaces that allow simple implementation of additional mechanisms.

The control and management of the ongoing auctions and trades is controlled over event-based messaging systems. Every user is allowed to start an auction (in case of single-sided auctions) and/or participate in the ongoing auctions by submitting their bids. Each bid contains the id of the requested service, the quantity of services, and the minimum offering price (in case of providers) or the maximum price the user is willing to pay for the service (in case of consumers). In each time iteration, market management component sends notifications to all interested parties about the current mechanism status. If the mechanism allows it, the users may then replace their bids with the new ones, in case their existing offers have expired. Once a match occurs, the users are notified, an SLA is created, and the trade executed. Note that the market platform does not mediate the trading process.

Currently, the simulator allows the trade of adaptable standardized services, but also provides the option to implement and trade an arbitrary form of resources.

6.3.5 Market front-end

The marketplace is accessible over the set of front-end services that provide support for searching and selecting public standardized services, submitting SLA mappings, placing bids for resources, and managing properties of their accounts, i.e., their participation properties. The front-end services provide simple platform-independent interfaces that allow usage of the market platform from an arbitrary platform.

6.3.6 Trading agents

Service provider/consumer layer represents users (agents) who describe their services using private SLA templates and offer/request services on the market via market front-end. Each agent is responsible for: (1) creating SLA mappings to the index SLA template, (2) selecting best-fitting public SLA templates that maximize utilities of the market users, and (3) placing bids for the required services. To perform these actions, the agents apply their bidding strategies. The simulator currently support the *zero-intelligence plus* (ZIP) strategy [67], but provides a set of interfaces that allow a simple implementation of additional strategies. ZIP traders are minimally simple software agents with almost (but not actually) zero intelligence. Basically, they are simple stochastic agents that adapt over time using an elementary form of machine learning. As demonstrated in previous works [68, 69], ZIP is a simple method that provides a human-like trading behavior with a significant precision. Our implementation of the ZIP traders follows the one described originally in [66].

6.3.7 Simulation control

Running, controlling and reviewing simulations is performed by the simulation controller. The simulation controller starts a simulation either by randomly creating input data or by taking traces of previous simulation runs. The data that should be created or loaded includes: number and properties of market participants, their private SLA templates, initial public SLA templates, and the prices for which they are willing to sell or purchase services. In Chapter 7 when we discuss several simulation scenarios and evaluation results, we will describe how each of the input data is initially created.

In order to observe changes in the market and assess whether it is possible to detect and adapt to unexpected user behavior and actions, we implement certain simulation scenarios. In particular, the simulation controller offers a set of interfaces for modifying market user base, the actions they perform, and other environmental factors on the fly in order to observe market change. One such scenario is evolution of a cartel: at a certain point in time, the simulation controller groups several service providers into an artificial cartel that dictates their pricing schemes. This, naturally, affects the trade in the market and may have numerous interesting outcomes, which are detected by our monitoring sensors.

The initial simulation properties are specified in form of Java property files that are sent to the simulation controller as input values. The properties that can be specified in a property file include: simulation duration, number of market participants (sellers and buyers), number of requested/offered resources by a participant, minimum and maximum bid prices, the average validity of a bid (as a number of time iterations), allocation and pricing methods that should be used, user agent strategies that should be applied, simulation scenarios and their properties (e.g, how many users should be affected, at what time iteration and for how long, and, finally, what change they should implement), market properties that should be observed, and goals that the market should achieve.

CHAPTER 7

Evaluation

In this chapter, we present the evaluations of the contributions that address the challenges based in this thesis. In Section 7.1, we evaluate our monitoring framework for self-aware cloud markets. In particular, we present a simulation testbed implemented within GridSim that we have extended by our monitoring sensors, give comments on simulation results, and present the lessons learned from this process that were the main motivations for building our agent-based market simulator. In Section 7.2, we use VieSLAf [30, 31] and the simulation framework introduced in Chapter 6 to evaluate the approach of standardizing computational services. In particular, we present several simulation scenarios and discuss the benefits and shortcomings of our approach, as well as possibilities for computation of the "optimal market setting". In Section 7.3, we evaluate our approach of automatic SLA matching and selection and discuss the savings in market participation costs that this approach brings. Finally, in Section 7.4 we summarize evaluations of all contributions and make a final conclusions about their benefits with respect to the research questions listed in this thesis.

7.1 Identifying market inefficiencies with the monitoring methodology

In this section, we present the evaluation of our monitoring methodology for self-aware cloud markets described in Chapter 3. In particular, we build a realistic market scenario within the GridSim simulator and apply our monitoring sensors to observe market performance and identify any peculiarities and inefficiencies withing the platform. By analyzing usefulness of our methodology, we address Research question 2 presented in Chapter 1 and demonstrate the effects of our study to the vision of autonomic markets, as questioned by Research question 1.

7.1.1 Use case

As already explained in Chapter 3, a monitoring methodology permits the assessment of a market platform's performance and its underlying infrastructure. The monitoring data, i.e., the market performance metrics presented in Chapter 3, are, therefore, important for the identification of (un)expected market behavior. For example, a monitoring methodology should be able to recognize a (sudden) drop in market liquidity or change in market revenue. Showing that a monitoring methodology is able to cope with this task is only possible through simulation of various trading scenarios. In this section, we present one such scenario (i.e, a case study) serving as a fitting context to observe inefficiencies and peculiarities within a cloud market.

In our case study, we adopt the Continuous Double Auction (CDA) as the mechanism for matching sellers and buyers of a particular good and for determining the prices at which trades are executed. In CDA, bids and asks may be placed at any point in time. A seller's ask specifies the good to be sold and its ask price. Similarly, a buyer's bid states the good to be purchased and a willingness to pay or bidding price. The orders are maintained in an order book in bid and ask priority queues, which are ordered by price: ascending for bids and descending for asks, with bids and asks with equal prices ordered by the time of submission. When a new bid is received, it is compared with the first ask of the order list. Similarly, if the new-coming order is an ask, it is compared with the first bid of the bid queue. A trade is executed if the price in the ask is lower than or equal to the bid's value. Otherwise, the order is added to the order book. Upon the execution of a trade, participants are informed of the result and the orders deleted from the order book. The trading price is calculated as $(ask + bid)/2$. Transactions continue in this manner until no more matches can be found.

In our case study, we consider a trade of one resource type (for simplicity) traded between 200 buyers and 100 sellers. At first sight, it seems that this experimental scenario creates a demand higher than the supply, which somewhat contradicts the infinite elasticity assumption of the cloud paradigm. However, it is important to note that each buyer generates only 100 jobs uniformly distributed over the simulation time. On the other hand, providers have a fixed amounts of resources that are constantly (and infinitely) offered to the buyers. This scenario has a peculiar outcome: after a certain point in time, most of the buyers will receive all services that they required and will stop sending new bids to the market. This result raises many new questions. For example, how will this affect the market? Can a marketplace work efficiently with the lack of requests for services? How will sellers react to the sudden drop in demand? And, most importantly, is our monitoring methodology and goal-based performance assessment able to detect this anomaly such that autonomic market adaptation could at a later date be applied?

In order to answer these questions, we implement the presented case study in GridSim and monitor the market infrastructure and market mechanism in time frames of 5 seconds. The

Table 7.1: Simulation settings

Parameter	**Setting**
Policy	CDA
Number of buyers	200
Number of sellers	100
Number of jobs per buyer	100
Number of resource types	1
Time interval for monitoring intervals	5 seconds

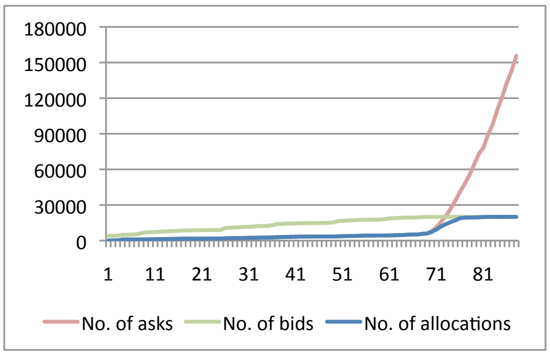
auction process is conducted as follows. Initially, a user (a provider or a consumer) submits requests to a broker. The broker is responsible for submitting and monitoring requests on the user's behalf, as well as managing the CDA instance and setting additional parameters, such as request length, quantity of auction rounds, reservation price, and policy (i.e., market mechanism) to be used. Since the broker is also the auctioneer, they inform the bidders that a CDA is about to start, create a call for proposals (CFP), set its initial price, and broadcast the CFP to all bidders. Resource providers set their bids for offering a service for a given period of time. When bids or asks are placed, the auctioneer clears the auction according to the CDA definition and broadcasts matches to the market participants. Table 7.1 summarizes key simulation settings.

As the behavior of participants is an important detail in the simulation, we utilize the Zero Intelligence Plus (ZIP) [67] bidding strategy. We selected ZIP for two reasons: Firstly, it is a well known and commonly applied bidding strategy that is often used as a benchmark for evaluating bidding strategies. Secondly, ZIP is already implemented within GridSim, which is not the case for other well know benchmark strategies. ZIP works by drawing a random price from a minimum and maximum value, but observes the success the previous bidding action: if successful, ZIP will select a new random price that is lower for consumers/higher for providers. If unsuccessful, ZIP will select a new random price that is higher for consumers/lower for providers. Although a simple strategy, ZIP has a well established history and frequently used in the literature.

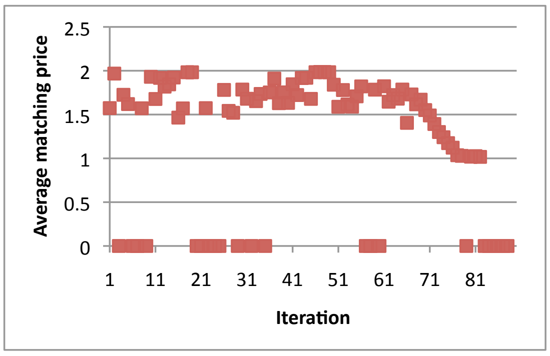
7.1.2 Simulation results

We begin our discussion on the simulation of the previously presented case study by analyzing the simulated activity of market participants. In particular, we discuss Figure 7.1a that depicts the total (cumulative) number of buyers' bids and sellers' asks, as well as the total number of allocations (i.e., matches between bids and asks) during the simulation. Figure 7.1a demonstrates the nature of the created market scenario: at the beginning, the marketplace behaves like the traditional buyers' market with the demand significantly larger than the supply. Consequently, the number of allocations is in line with the number of providers' ask. However, towards the end of the simulation process, the market suddenly looses stability. At this point in time, all buyers have received a match for their requirements and stopped sending new bids to the market. The market scenario suddenly changes: providers start acting more aggressively by increasing their number of asks since the competition increases and they no longer receive matches. Therefore, supply instantly becomes larger than demand and results in the change to the providers' market.

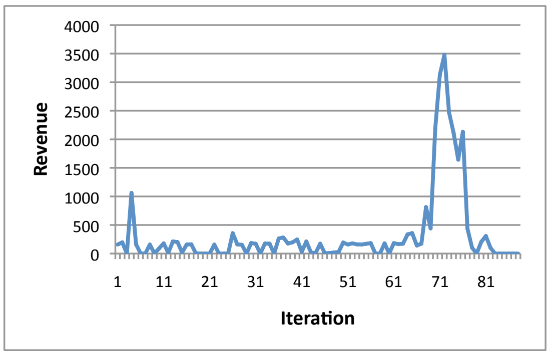
The demonstrated change in market stability caused by the lack of active buyers can be seen by inspecting several other monitoring metrics. Figure 7.1b depicts the average matching price created by the CDA in a given time frame. During the time of market stability, i.e., when the number of active traders is sufficient and stable, matching price is relatively firm. However, at the point when the buyers' bids stop coming in and the number of new sellers' asks dramatically rises, the average matching price suddenly falls by almost 50%. This is the indirect effect of the low demand and high supply, i.e., the low number of bids and a large number of asks in the order book. The relation between the number of asks and the average price is caused by the nature of the continuous double auction, which allocates highest bids with the lowest asks. Since high



(a) Total number of bids, asks, and allocations



(b) Average price



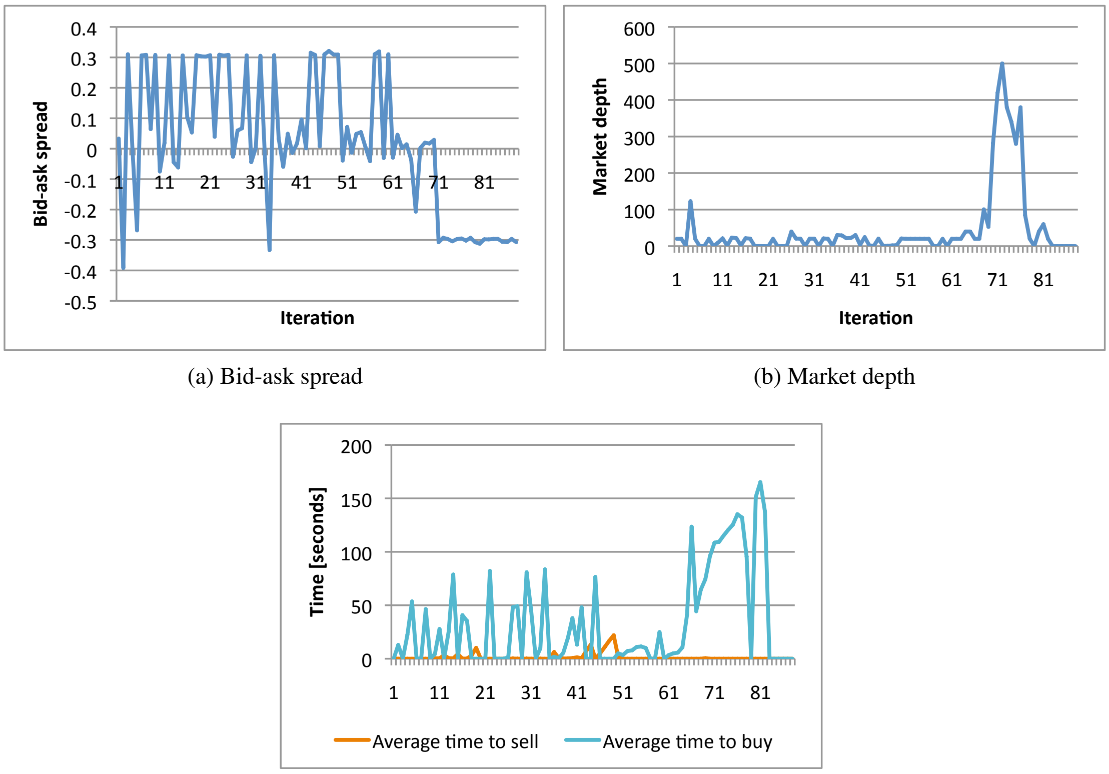
(c) Market revenue

Figure 7.1: Simulation results: CDA performance

supply results in a large number of low-priced asks in the market, the allocations with low prices are more numerous and the average price sinks.

The increase in supply also causes a sudden, but short-termed increase in number of resource allocations (Figure 7.1a and 7.2b). Due to the large number of providers' asks in the order book, buyers have a higher change to find a resource for a lower price. Therefore, at the point when users' bids stop coming in and when the number of sellers' asks increases, all buyers that were not able to find resources due to their low bid prices are suddenly allocated. This, of course, results in a big increase in the number of allocations which drops equally fast as all the buyers get the resources they have been looking for. Consequentially, market revenue also rises, despite lower prices of resources (Figure 7.1c). However, as well as the number of allocations, this effect is short-termed.

Figure 7.2 depicts the values of the three measures for market liquidity: bid-ask spread (Figure 7.2a), market depth (Figure 7.2b), and immediacy of matching (Figure 7.2c). During the period of market stability, i.e., while there are sufficiently many buyers bidding in the market, the value of the bid-ask spread is positive. This indicates a liquid market: average buyers' bid price is larger than the average sellers' ask price which results in the high purchasing possibilities of buyers and, therefore, a large number of resource allocations. Peak values of the bid-ask spread



(a) Bid-ask spread

(b) Market depth

(c) Immediacy of matching

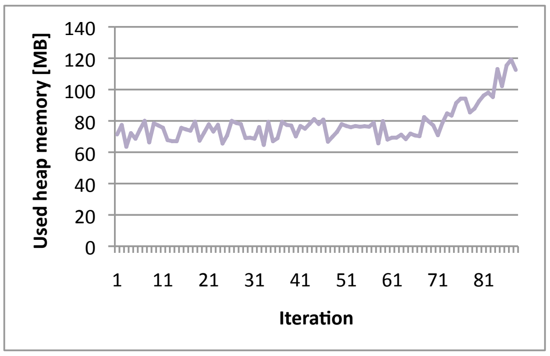
Figure 7.2: Simulation results: market liquidity

are common due to the random nature of the bidding strategies used by the market participants in the case study. However, towards the end of the simulation process, the bid-ask spread suddenly sinks below zero. This result can be explained by observing Figure 7.2a: due to the lack of buyer bids in the market, the bid-ask spread represents only the average price of the sellers' asks. Since there are no buyers for their resources, the market liquidity suddenly sinks.

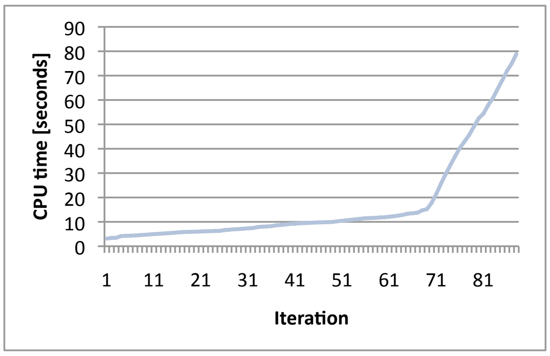
Figure 7.2b presents another perspective on market liquidity: the market depth. Market depth measures the number of matches between buyers' bids and sellers' asks in every iteration (in our simulation set to 5 seconds). Therefore, the results presented in Figure 7.2b do not significantly differ from those presented in Figure 7.1a. As already explained, due to the large number of sellers' asks at the end of the simulation process, all buyers that have not already received a match for their requirements suddenly have a significantly improved matching chance. Therefore, after the buyers' bids stop arriving and when the providers start a strong competition by submitting a large number of asks, the number of resource allocations significantly rises. This effect is again only short-termed, since all buyer bids that remain in the order book are quickly matched to the incoming seller asks.

Another interesting view on market liquidity is presented in Figure 7.2c. The figure depicts immediacy of matching (i.e., the average time between submitting an ask or a bid and it being matched to a bid or ask respectively) for buyers and sellers. Since the demand is significantly higher than the supply, the buyers must wait longer than the providers to receive an allocation. This points to an obvious conclusion: in the simulated trading scenario, selling is remarkably easier than purchasing. This situation dramatically changes towards the end of simulation. First, when sellers start sending a large number of lower-priced bids, the buyers that have been waiting for a long time due to their low requested prices suddenly get the requested resources. Since they were waiting for a long time, buyers' immediacy of matching results in a very high value. However, after all buyers have received an allocation, sellers continue submitting asks for their offers. Since there are no buyers left to purchase their resources, they never receive an allocation and the immediacy cannot be computed. Therefore, the zero value of providers' immediacy of matching at the end of the simulation does not mean that they receive their allocations immediately. On the contrary, the immediacy of matching jumps to the positive infinity instead.

From the infrastructure perspective, the market instability causes high utilization of hardware resources. Sudden growth of supply causes an increase in CPU and heap memory usage, due to the high number of sellers' asks in the order book and their exponential growth relating to constant reordering of the priority queue (Figure 7.3a and 7.3b). Note that the graph depicted in 7.3b may also represent the platform execution cost, assuming a fixed cost for platform resources (see the metric mapping in Figure 3.3). Namely, although the expression unit would be different and the values multiplied by a constant, the shape of the graph would be the same. Therefore, we can conclude that the market inefficiency and instability caused by the sudden cease in number of active participants reduces the profit to the market platform in general.



(a) Used memory



(b) Computation time

Figure 7.3: Simulation results: infrastructure performance

7.1.3 Discussion of market goals

We now turn our discussion to the market goals presented in Chapter 3. At this stage, we do not focus on how market adaptation should be undertaken, or the threshold values for each goal, but rather highlight how each goal that we have defined acts as an indicator in our case study. Interestingly, but not surprisingly, they are not unanimous in their categorization of a potentially catastrophic scenario, which strongly reinforces the need for a sound analysis approach when considering both market performance and adaptation options.

First, consider market liquidity. Liquidity is loosely defined as the ease of which a tradable artifact can be bought or sold without significant changes to price or value. Therefore, a market goal would be to retain a “good” or high level of liquidity. It is clear from Figure 7.2 that with even this simple definition the market initially has a “reasonable” liquidity, as it is easy to sell, but not that easy to buy. When our scenario begins, a notion of illiquidity should arise, which is clearly demonstrated by the sharp drop in the bid-ask-spread, average price and immediacy. From this observation, even without explicit thresholds for a desired liquidity value, we can deduce that this part of our monitoring methodology can adequately highlight sudden or extreme changes.

Second, consider the goals: provider revenue, transaction volume, platform profit and the number of allocations. When we observe increases in the values that constitute these goals, a market can be considered healthy, or at the very least has a certain element of growth. Therefore, the goal of the market corresponds to these values either retaining a specific minimum value or increasing over time at a minimum rate. Let us consider the simple case of market revenue completely independently of the market’s applied business model, as regardless of the applied business model if revenue increases so too does its potential for profit. Figure 7.1c illustrates a fairly steady revenue stream that culminates in a huge spike. Therefore, initially these metrics indicate that performance is improving (i.e., as short term gain), but is subsequently followed by an equally large reduction in performance. In our simulation, time is relatively undefined - it is nothing more than the computational runtime of the simulation. If instead it were measured in months or years, then the false positives stemming from the first half of the spike could have dramatic consequences. Therefore, the initial contradictions of these four metrics to the liquidity metrics act as early warning signs that show the importance of not considering goals or sets of goals in isolation.

Finally, consider the remaining two goals: low or predictable platform execution costs, and a minimum or balanced number of active traders such that the market receives a steady number of transactions per time period. Interestingly these two metrics although tied to different goals are in fact two very clear indicators in our given scenario. If we assume that the cost of a single platform resource instance remains constant, for example through an EC2 prefixed reservation price for a given number of virtual machines within a framework contract, then we can say (according to the mapping in Figure 3.3) that the platform costs are directly proportional to compute time. Based upon the observations from our scenario, we can tentatively highlight that both of these metrics and ultimately their goals, can due to their simplicity and ease of observation act as initial indicators when something is a miss. Similarly, when we consider the initial contradiction between liquidity and goals like market growth these two metrics act as good tie breakers.

7.1.4 Lessons learned

In the course of this thesis, we have taken the first steps towards the vision of an autonomic cloud marketplace. To ease the implementation process we selected GridSim as a means to rapidly develop a prototypical simulation framework for a computing on demand marketplace.

We did this as a means to evaluate the feasibility of our monitoring methodology with a simple calibration scenario. It has been easy to simulate a simple case study, but it is not, or at least should not be, a valid scenario for a cloud marketplace. For our purposes GridSim was adequate. However, its primary purpose is to simulate grid environments and not marketplaces. This means that moving towards more realistic scenarios (such as the introduction of new trading artifacts, disruptive technologies or service types; different participant types, e.g., market makers, speculators, monopolists, prosumers, brokers; changes in participant behavior, market growth or contraction, etc.) would mean taking GridSim further away from its initial use case, making it harder to control and extend. We also note that making the marketplace more dynamic in terms participation causes problems in the GridSim runtime, which assumes a certain element of static participation, e.g., the number of users, as well as their demand remains fixed. These results have been the main motivation for implementing a more flexible simulation environment that we have described in Chapter 6.

We have also seen that the interplay of goals when considering system-wide market performance is almost more important than the individual goal settings. Our scenario was very extreme and, therefore, lent itself nicely to illustrate this. However, more realistic scenarios may not result in such extreme changes in observed performance. Therefore, the autonomic management methodology of the marketplace (Chapter 3) demands significant studies, especially for the analysis and knowledge components, using real world data from existing markets or from human-based lab experiments from the domain of experimental economics [189, 190].

7.2 Impact of adaptive standardized cloud resources on market performance

In this section, we present the evaluation of our methods for standardization of computational resource specifications, as described in Chapter 4. In particular, we analyze the differences between differentiated and standardized goods markets in terms of market liquidity, users' utility and cost of market participation. By evaluating the standardization approach, we address Research question 3 presented in Chapter 1. Besides evaluating the usefulness of the this approach, we investigate maximization of its benefits by comparing different approaches for its realization.

7.2.1 Simulation testbed

For the evaluation of our approach of computational resource standardization, we set up a testbed in the market simulator described in Chapter 6. In Figure 7.4, we present another view of the market platform framework, i.e., its components that are relevant to the simulation scenario that will be discussed here. The simulation framework, as depicted in this figure, comprises a cloud marketplace with a service directory that stores information about resources available in the market (i.e., public SLA templates), the index SLA template, and the SLA mappings submitted by the users. The market is accessed using the front-end services for administration (e.g., creation of SLA templates), accounting (e.g., creation of user accounts), querying (e.g., search for an appropriate SLA template), and management of SLA mappings (e.g., definition of mappings for a user). Users (i.e., service consumers and providers) utilize the SLA mapping middleware to create and manage their SLA mappings to the index SLA template. The market self-adaptation cycle constantly performs monitoring of market- performance (from both the economical and infrastructural perspectives) and modifies standardized products or derives new standardized products.

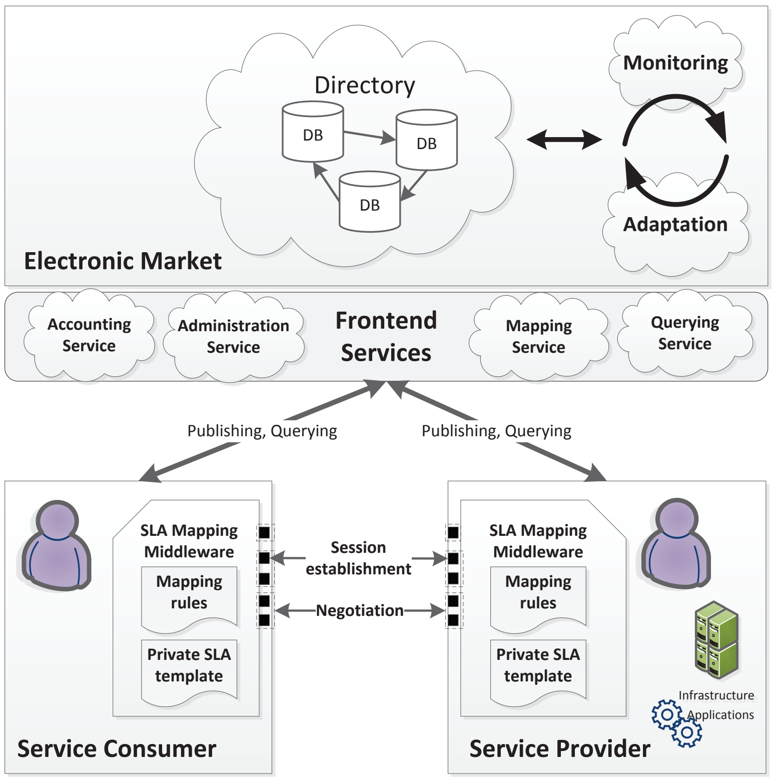


Figure 7.4: Simulation environment

Figure 7.5 depicts the simulation process as assumed in this section. Initially, the market holds: (1) a public SLA template that is based on a randomly created SLA template or a randomly picked private SLA template of a service consumer or a service provider, and (2) the index SLA template that contains only the parameters from the initial public SLA templates. A number of service providers and service consumers randomly generate private SLA templates that contain a number of SLA parameters and associated SLA metrics and service level objectives (SLOs). In the users' private SLA templates, the desired parameter values specified in users' SLOs are given in form of ranges of real numbers. For example, an SLO value for an SLA parameter $ErrorRate$ may be [0,1]%, stating that any value between 0% and 1% is acceptable for the user. The values of SLOs are created randomly, but with a predefined width of the value range, which is given as a percentage of the maximum possible SLO value range. The number of market participants (providers and consumers), SLA parameters per private SLA template, and predefined SLO width value are specific for every simulation scenario and will be explicitly stated in each of the following sections.

The simulation process is conducted in several steps, as depicted in Figure 7.5. In the first step, market participants fetch the index SLA template and create the SLA mappings to bridge the differences in SLA parameter specifications. In this process, a market participant can execute one of the following actions:

1. create ad-hoc SLA mapping(s) between an SLA parameter that exists in both their private SLA template and the index SLA template, if the specification of the SLA parameter differs in the two SLA templates,
2. create a future SLA mapping, i.e., add the parameter to the index SLA template, if the SLA parameter exists in the participant's private SLA template, but does not exist in the index SLA template, and
3. do nothing, in case the SLA parameter exists in both the participant's private SLA template and the index SLA template with the matching specification details.

The creation of redundant SLA mappings (i.e., mappings that were already created by another participants) is solved by applying the methods described in Chapter 5 and will be evaluated in Section 7.3.

After all users have submitted their SLA mappings, the adaptation process is started (step 2 in Figure 7.5): a clustering algorithm is applied to group similar private SLA templates and an adaptation method utilized to select the specification details of the new public SLA template for a given cluster of private templates. This step results in a set of new public SLA templates that replace the initial SLA template. In the following step (step 3 in Figure 7.5), participants' SLA mappings are modified as described in Section 4.5.4.

In the following steps, market participants "manually" select the public SLA templates that are the most similar to their private SLA templates (step 4 in Figure 7.5) and the providers' service offerings and consumers' service requirements are matched (step 5 in Figure 7.5). In order to simplify our simulation model, we assume a very simple trading strategies applied by traders' agents in the market and a simple allocation mechanism. Furthermore, we do not consider any pricing methods or price-based service allocations. These assumptions mean the following.

- Market participants only specify a service that they offer or require. This means that they do not specify the quantity of services that they wish to buy or sell (i.e., they always require only one service) and are not interested in the prices of the services.
- Since SLOs are in public SLA templates specified as single numerical values, a private SLA template and a public SLA template can be matched if the SLO values of all SLA parameters from the public SLA template are inside the value ranges specified by the participant's private SLA template. The participants select the public SLA template that (a) may be matched with their private SLA template and (b) has the minimum distance from their private SLA templates, which is calculated using Equation (4.9).
- Market participants are equally satisfied with all service requirements (in case of service providers) and service offerings (in case of service consumers) as long as their requirements are satisfied. This means that as long as a participant's private SLA template

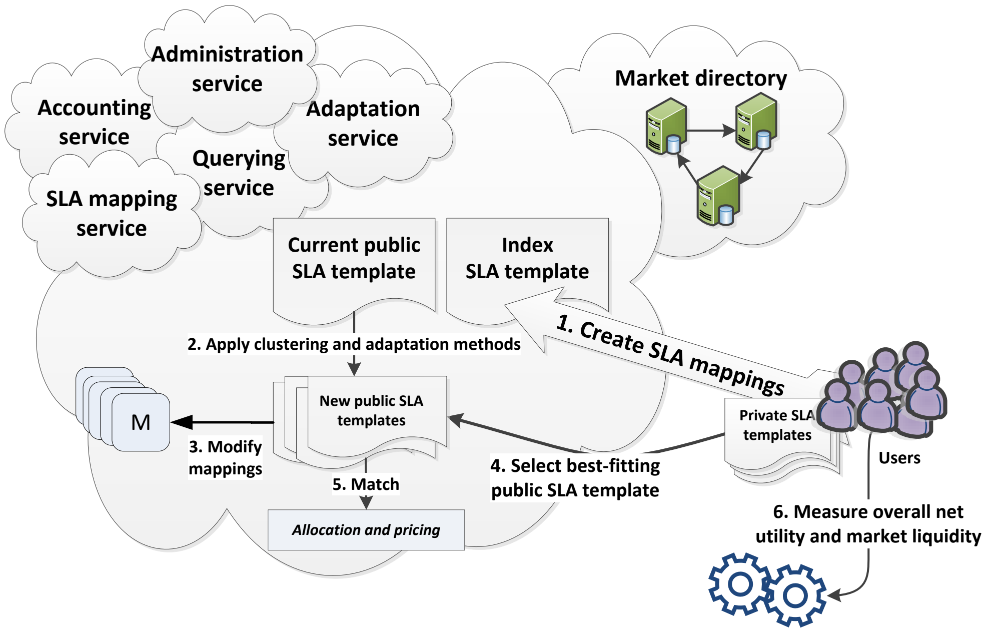


Figure 7.5: Simulation testbed for the standardized goods market

matches with a public SLA template, the user will select any provider's offering that is also associated with the public SLA template.

Note that only those participants who opt for a public SLA template are able to be part of a trade. Also, note that finding a matching public SLA template does not guarantee an allocation with another trading party, as it may happen that the demand and the supply are not balanced (i.e., there are more providers than consumers or vice versa).

In order to compare our approach of trading standardized cloud services, we also simulate a trade of numerous differentiated resources. Similar to the process of trading standardized resources, in the differentiated goods market the participants first fetch the index SLA template and submit their SLA mappings. In the second step, the trade is executed. In this process as well the participants do not specify service prices nor they differentiate between offers/requirements of different service providers/consumers as long as their specifications match. In this case, a match is made between two private SLA templates and is ensured if the intersections of the SLO value ranges of all SLA parameters from the two SLA templates is not an empty set. The participants, therefore, iterate through all service offerings (in case of consumers) or requirements (in case of providers) and select the first service that can be matched with their requirements.

In the following sections, we evaluate two aspects of our approach. First, we use the utility and cost models described in Section 4.6.2 to assess the benefits and performance of the clustering algorithms and the adaptation methods as means to create standardized resources. In particular, we will used the notion of utility and cost to determine how well the public SLA templates represent the needs of market participants' requirements and determine which combination of clustering algorithms and adaptation methods fits the most to the given scenario. This evaluation will be detailed in Section 7.2.2. Additionally, we use the utility and cost models to evaluate the benefits brought by the SLA mapping adaptation algorithm detailed in Section 4.5.4, which will be discussed in Section 7.2.3.

On the other hand, we will use the measures of market liquidity presented in Section 4.6.1 to discuss whether and to what extend the approach of resource standardization improves market liquidity when compared to the market with numerous differentiated resources. Furthermore, using the measure of market liquidity, we will determine the market setting (i.e., the number of standardized resources) that is needed in the market in order to maximize market activity. These discussions will be presented in Sections 7.2.4 and 7.2.5, respectively.

7.2.2 Creation and management of public SLA templates

In this section, we analyze the fitness of automatically created standardized SLA templates to the requirements of market participants. Herewith we consider a simulation scenario with the initial public SLA template containing 8 SLA parameters. Users' private SLA templates are generated randomly at the beginning of the evaluation process and can contain up to 11 SLA parameters, where users can choose between 5 predefined parameter names and 4 metrics for each SLA parameter. Table 7.2 summarizes the simulation settings. To assess the benefits of grouping similar supply and demand, we additionally run a simulation without applying clustering algorithms. In this scenario, adaptation methods are applied to create one single public SLA template. For the cost-benefit evaluation of our approach, we use the overall net utility as defined by Equation (4.19).

Table 7.2: Simulation settings

Parameter	**Value**
Number of service users (consumers and providers)	$100 \leq n \leq 10000$
Number of initial (currently existing) SLA templates	1
Number of parameters in initial SLA template	8
Number of parameters in private SLA templates	$\leq$11
Number of different parameters considered at most	11
Size of the set of possible parameter names per SLA parameter	5
Size of the set of possible metrics per SLA parameter	4

We begin our discussion by considering Figure 7.6. In this figure, the first bar of each of the three sets of bars represents the overall net utility achieved for the case that only a single new public template using the basic SLA mapping (i.e., without applying the algorithm presented in Section 4.5.4) has been used. The other three bars represent the overall net utility achieved by utilizing the three clustering algorithms (i.e., DBSCAN, k-means with rule-of-thumb, and k-means with Hartigan's index). As the comparison shows, the overall net utility, which is obtained by generating only one public SLA template for all market users with the basic mapping, is significantly lower than the overall net utility obtained with the clustering algorithms. This is due to the fact that many new public SLA templates are created that differ less from the users' private templates, reflecting users' needs more precisely.

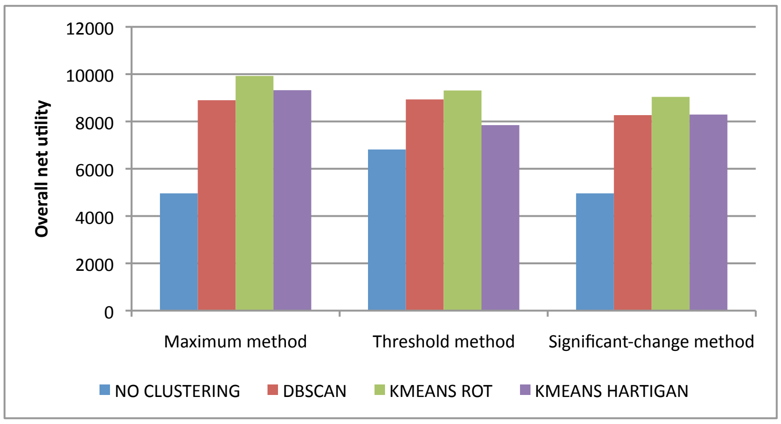


Figure 7.6: Overall net utility (without automatically modifying users' SLA mappings)

When comparing the overall net utility gained by utilizing different adaptation methods, we can conclude that the best results are achieved by using the maximum method. Although the threshold method causes significantly lower number of changes (due to the high threshold) and, therefore, very low cost, the maximum method adapts the public templates more frequently and, therefore, achieves a high utility. The highest overall net utility is achieved by the k-means clustering algorithm with the rule-of-thumb method, due to the creation of a large number of very specific clusters.

Concluding, the utility rate highly depends on the number of generated clusters. The more clusters are created, the higher the utility will be. This relation can also be seen when comparing the net utility shown in Figure 7.6 with the actual number of clusters. Table 7.3 shows the number of clusters per clustering algorithm, depending on the number of users creating SLA mappings. The maximum rate of overall net utility for n users can be achieved by creating n clusters, i.e., by generating one public SLA template per user. In this case, the public SLA templates would be equal to the users' private templates. The utility rate would in this case be maximum and the cost would be equal to 0. However, by raising the number of products on the market, i.e., the number of new public SLA templates, market liquidity is at the lowest point. Therefore, it is necessary to find a balance between keeping low cost of creating new SLA mappings and ensuring high liquidity of market goods. The problem of finding the optimal number of new public SLA templates will be discussed in detail in Section 7.2.5.

Not only the number of generated clusters influences the overall net utility. Namely, the dataset properties also determine the quality of newly generated public SLA templates. For this reason, it is not sufficient to investigate only the utility. Instead, the properties of the simulation dataset that may have influenced the results of the utility measurements. For this purpose, we introduce the measures of compactness and isolation.

To introduce the measures of isolation and compactness, we define the following variables: C represents a set of clusters, where $C_j \in C$ is a cluster and C_{jc} is its centroid (i.e., the public SLA template). R_i is a clustering item and $D(R_i, R_k)$ is the distance between two clustering items as defined by Equation (4.9). $|C|$ represents the total number of clusters and $|C_i|$ represents

Table 7.3: Number of generated clusters per algorithm and per number of users.

Users	DBSCAN	k-means (RoT)	k-means (H)
100	5	7	3
200	5	10	3
300	6	12	3
500	6	15	5
1000	7	22	6
2000	7	31	6
5000	8	50	6
10000	10	69	10

a number of items in a cluster C_i.

Compactness is reciprocal of the average distance between users' private SLA templates within clusters, and is formally defined as follows:

$$\left(\frac{1}{|C|}\sum_{Cj\in C}\left(\frac{1}{|C_j|(|C_j|-1)}\sum_{R_i\in C_j}\sum_{R_k\in C_j}D(R_i,R_k)\right)\right)^{-1} \tag{7.1}$$

Since a cluster represents a group of users with similar requirements, high compactness implies a well-formed market. Figure 7.7 depicts the rates of compactness achieved by those three clustering algorithms and the maximum method. For this purpose, we have simulated SLA mappings specified by a varying number of market users, ranging from 100 to 10000 users. The adaptation method used for the evaluation is the maximum method.

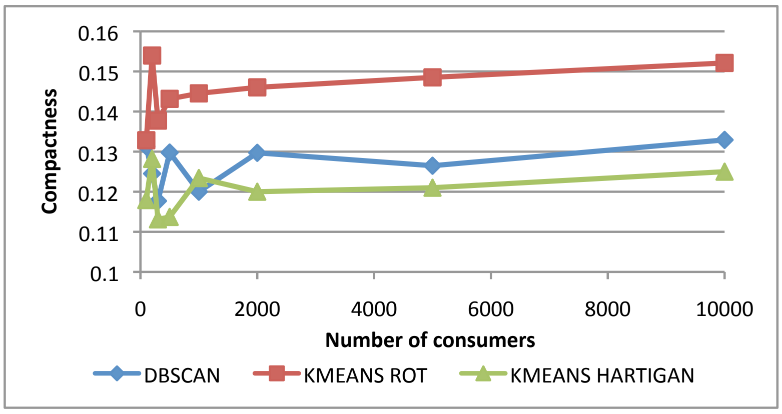


Figure 7.7: Compactness of the new public SLA templates

High utility is common for compact clusters, since all members of the cluster have similar properties, and the cluster centroid (i.e., newly generated public SLA template) reflects those properties more precisely. Items of a low compact cluster differ to a larger extent, and it is probable that the utility achieved by creating a new public SLA template will be lower. High compactness can be achieved by creating a large number of smaller clusters and is maximized when there is one cluster per item.

As depicted in Figure 7.7, the k-means algorithm with the Hartigan's index achieves lowest rate of compactness. This is due to creating small number of general clusters, where items mutually differ to a large extent. Therefore, this algorithm results in low overall net utility. The k-means algorithm with the-rule-of-thumb achieves the largest rate of compactness, and consequently a larger overall net utility, due to a large number of clusters.

Isolation is an average distance between newly generated public SLA templates and is defined as follows.

$$\frac{1}{|C|(|C|-1)} \sum_{C_i \in C} \sum_{C_k \in C} D(C_{ic}, C_{kc}) \tag{7.2}$$

Isolation represents the measure of how different newly generated public SLA templates are. The larger the distance, the more distinct the templates are. We imply that higher isolation of public templates ensures more distinct products, and therefore provides better chances for creating product niches. Figure 7.8 presents the isolation rates achieved using the maximum method.

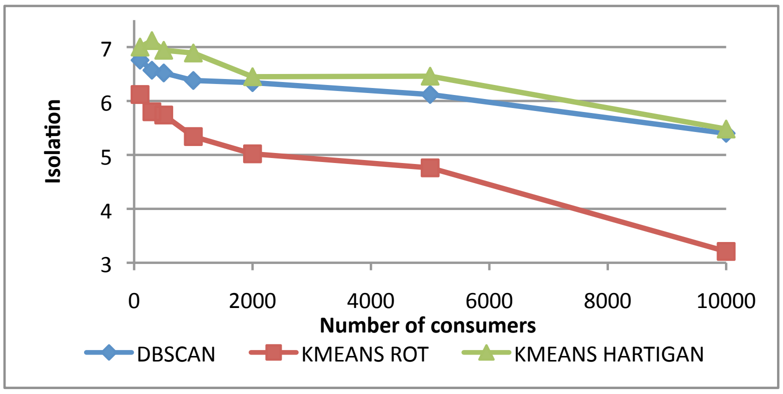


Figure 7.8: Isolation of the new public SLA templates

High isolation can be achieved by creating a small number of very distinct clusters. However, by doing so, we are reducing the compactness of the clusters, since they will contain larger number of items with broader variety of preferences. Therefore, we must find a balance between creating more general clusters with high isolation (i.e., distinct market products) and well-formed clusters with respect to their specificity and compactness.

As it can be seen in Figure 7.8, k-means algorithm with the Hartigan's index achieves a high rate of isolation. This result is not surprising, since this algorithm creates a small number of clusters and, as presented in Figure 7.7, achieves the lowest rate of compactness. Additional to that, this algorithm achieves the smallest rate of the overall net utility. DBSCAN also creates relatively small number of new public templates and creates well-isolated clusters. Finally, the k-means with the rule-of-thumb creates clusters with low isolation, due to creating a large number of clusters.

To conclude, considering the overall net utility, the compactness, and the isolation, the best results are achieved by the k-means clustering algorithm with the rule-of-thumb method for determining the number of clusters. As depicted in Figure 7.6, this algorithm achieves the highest net utility and compactness for all adaptation methods. This means that the newly created standardized products are created with the highest similarity to the user requirements. However, this comes with the cost of creating a large number of public SLA templates, which results in lower market liquidity. When compared with the k-means clustering algorithm with the Hartigan's index, it is more cost efficient, since it's computing complexity is significantly lower. In particular, unlike Hartigan's index, which requires several iterations of the k-means algorithm before finding the optimal number k, the rule-of-thumb method determines the number of clusters a priori. When compared to DBSCAN, it achieves significantly higher rate of the overall net utility. Having said that, we conclude that the k-means algorithm with the rule-of-thumb is the most appropriate method within the ones discussed for creating standardized services.

Scalability

For cloud environments, it is of great importance that resource markets are highly scalable. In order to assess the scalability of our approach, we measure the execution time of the clustering process for different number of users creating SLA mappings, varying from 100 to 10000. Results are depicted in Figure 7.9.

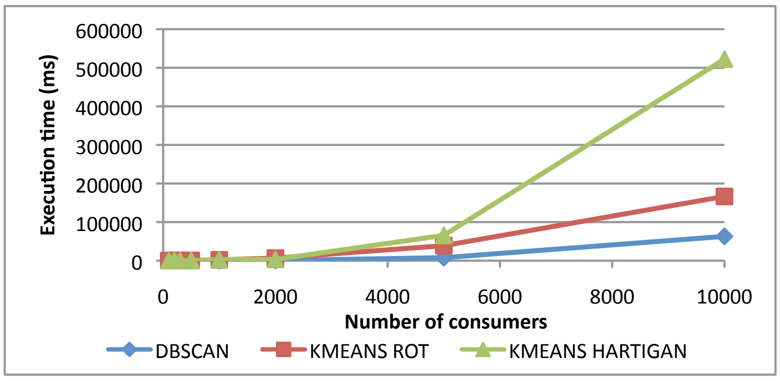


Figure 7.9: Simulation time

As shown in the figure, the k-means algorithm with the Hartigan's index takes most time to compute groups of SLA mappings. This is due to the nature of the Hartigan's index method, in which the clustering algorithm runs in several iterations and computes different number of clusters before it finds the optimal number k. The best performance is achieved by the DBSCAN algorithm. This result is also not surprising, since unlike the k-means algorithms, it creates a low number of clusters and does not compute cluster centroids, which incurs cost.

The results show that the approach presented in this thesis is highly scalable. Moreover, the execution time of k-means algorithm with the Hartigan's index, which is far the highest, can be improved by introducing heuristics for determining the initial number k, which we will consider in our future work.

7.2.3 Automatic management of SLA mappings

By not considering the cost of creating and adapting SLA mappings for the users, the cost of the basic SLA mapping approach is strongly reduced. In this section, in particular, we assess the benefits of the automation and compare it with the results presented in the previous section.

Figure 7.10 illustrates the overall net utility achieved by each of the adaptation methods with and without applying the algorithm for the automatic management of SLA mappings. For this purpose, we utilize only the k-means clustering algorithm with the rule-of-thumb, due to the best performance when compared to other clustering algorithms (Section 7.2.2). As also depicted, by applying the automation algorithm, the overall net utility is significantly higher than the overall net utility for the case when users must manually create new SLA mappings. Note, the overall utility does not differ for the two approaches, as the newly created public SLA templates are equal. The difference in the overall net utility comes from the reduction of the cost for creating SLA mappings.

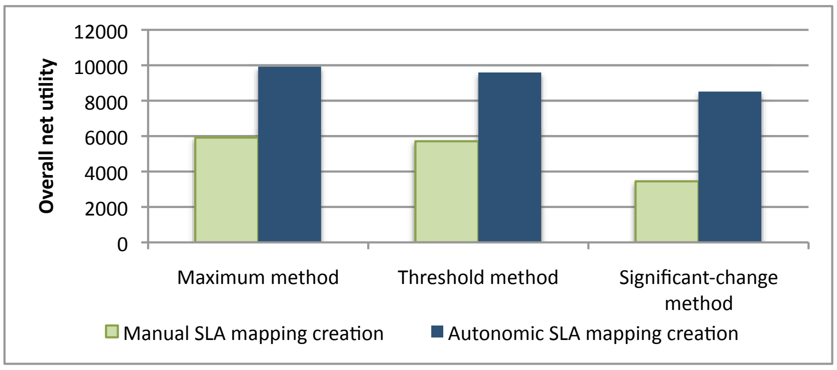


Figure 7.10: Comparison of the overall net utility of the k-means Rule-of-Thumb clustering algorithm with and without automatic modification of users' SLA mappings

The difference between costs incurred by the approaches is depicted in Figure 7.11. Since by utilizing the automatic mapping modification algorithm the users are not required to create any new SLA mappings, the overall cost for users is reduced almost to 0. The cost still exists since not all of the autonomically created SLA mappings fully satisfy users' needs. In particular, as explained in Section 4.5.4, the adaptation algorithm might not correctly recognize a newly added parameter's equivalent in a user's private SLA template. This, of course, incurs cost, since the user must rectify created mappings. However, note that the new SLA parameters are not added in public templates on a regular basis, but rather seldom, in case of a large change in user requirements. Therefore, the overall cost for users is in most cases of the adaptation process equal to 0.

It is important to note that the cost for creating SLA mappings has not vanished, but is carried by the marketplace instead. However, through this approach, the human interaction is significantly reduced and the cost for creating SLA mappings becomes negligible. Additionally, since the mappings are immediately updated, initial public templates can be deleted and replaced by newly created SLA templates. This was not possible in without the SLA mapping adaptation algorithm, due to the necessity to create new mappings before utilizing newly created public templates. This was sometimes hard to achieve, since the users preferred keeping the old public templates so to avoid additional cost of creating new SLA mappings. By dynamically replacing public SLA templates, the cost for maintenance and storage of public SLA templates is reduced.

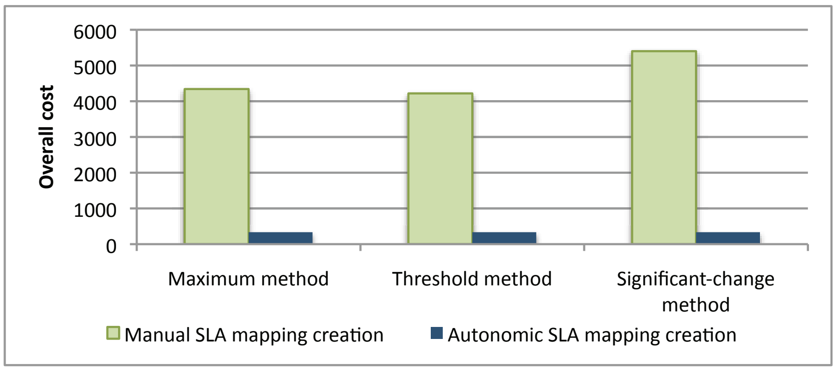


Figure 7.11: Comparison of the overall cost of the k-means Rule-of-Thumb clustering algorithm with and without automatic modification of users' SLA mappings

7.2.4 Effects of standardization on market liquidity

To assess the benefits of our approach to liquidity of the simulated market platform, we simulate both the trade of numerous differentiated and standardized computational resources and examine liquidity of the market in the given trading scenarios. For the sake of simplicity, each market participant can have either the role of a buyer or a seller, but not both at the same time. Furthermore, each participant wishes to sell or purchase only one service during the whole simulation period. Therefore, once an allocation occurs (i.e., when a match between a seller's offer and a buyer's requirement is found), both the buyer and the seller are removed from the list of users who have not yet received an allocation. Although the latter limitation may contradict the scalability requirements of the cloud computing paradigm that promise virtually unlimited resources, it helps in simplification of the simulation model without affecting the approach of standardizing computational resources.

In both the differentiated and the standardized approaches, the overall market depth is measured as the accumulative trading volume, i.e., the total number of buyers and sellers who have received an allocation. On the other hand, search cost is measured as the number of comparisons between SLA templates that the users have to perform in order to find a suiting trading partner. These are the comparisons between the buyers' and the sellers' private SLA templates in case of the differentiated approach, and between users' (buyers' and sellers') private SLA templates and the public SLA templates in case of the standardized approach.

To begin our analysis of the impact of standardized products on market liquidity, we simulate a trade of exclusively differentiated products as well as a trade of standardized products with the varying number of market participants (ranging from 200 to 15000), out of which we here specifically discuss three: with 600, 4000, and 10000 market participants trading. In each of the scenarios, demand and supply are evenly distributed, i.e., 50% of traders are buyers and the remaining 50% sellers. Note that the latter assumption does not hold in the real-world scenarios as cloud consumers currently significantly outnumber cloud providers. However, this assumption increases market activity and improves visibility of effects of resource standardization on market liquidity. The settings of the simulations discussed in this section are presented in Table 7.4.

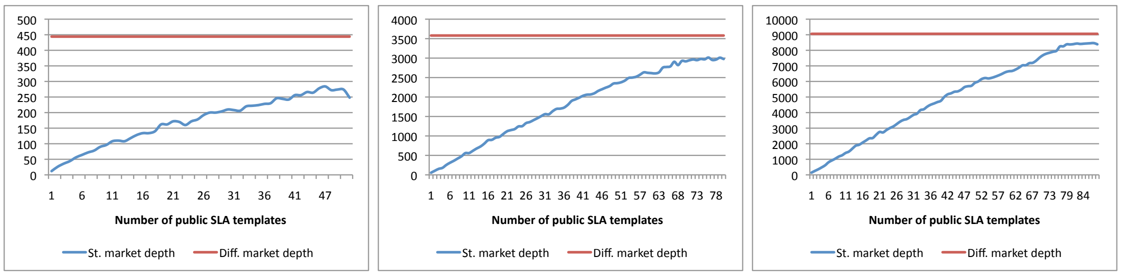
Table 7.4: Simulation settings

Parameter	**Value**
No. of market users	$200 \le n \le 15000$
Portion of buyers in the number of users	50%
Portion of sellers in the number of users	50%
No. of parameters in SLA templates	4
Width of the SLO value range	10%
Method to cluster users' preferences	k-means
Method to adapt standardized services	Maximum method
No. of services required/offered by one user	1

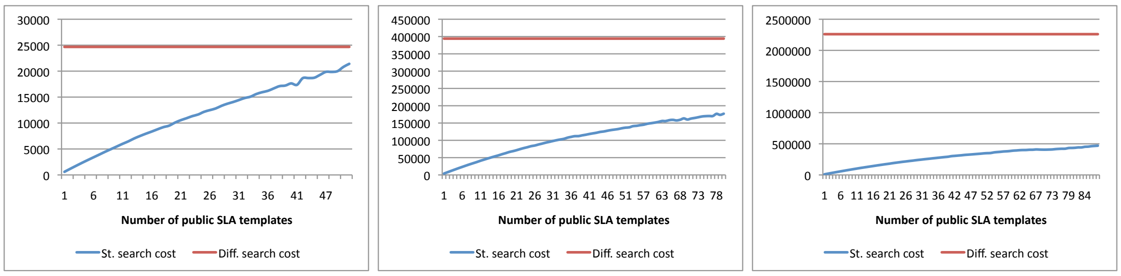
Figure 7.12 presents the simulation results. It contains 9 graphs arranged in 3 columns and 3 rows. Each of the rows presents one of the liquidity measures: Figure 7.12a depicts the overall market depth, Figure 7.12b depicts the search cost, and Figure 7.12c depicts their relative difference (also called "the aggregate liquidity measure"). In each of the rows, the left-hand graph presents the trade of 600 market participants, the middle graph the trade of 4000 participants, and the right-hand graph the trade of 10000 market participants. The horizontal axis depicts the number of created standardized products (i.e., public SLA templates) and the vertical axis depicts the result values of liquidity measures. Note that the change in the number of standardized products is the only change in the market condition that we consider. This change, of course, does not effect the trade of the differentiated goods. Nevertheless, in order to simplify the comparison between the two trading approaches, Figure 7.12 depicts the continuous but constant values of liquidity measures for the "differentiated approach".

Overall market depth

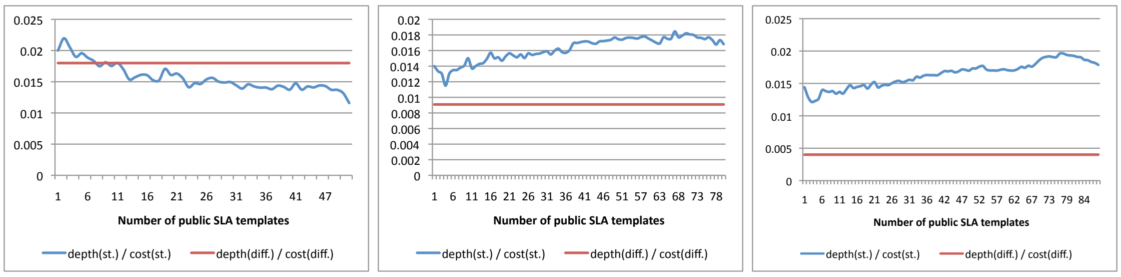
We begin our discussion on simulation results by considering overall market depth depicted in Figure 7.12a. The graphs in this figure present the expected dominance of the differentiated products over the standardized products in terms of the number of successful allocations of requirements and offers for services. Naturally, due to the high variety in resource types when trading differentiated products, probability to find an offer similar to a user's requirement is significantly higher. Regarding the standardized approach, the graphs obviously demonstrate that the overall market depth grows with the increasing number of standardized resource types in the market. In order to achieve the same amount of overall market depth as with the differentiated resources, the standardized approach should create a very large number of standardized resources. This, however, means that each standardized resource would be approximately equal to one private SLA template in the market and would only slightly differ from the differentiated approach. Moreover, since the number of standardized resources cannot be larger than the num-



(a) Overall market depth



(b) Search cost



(c) Relative difference between overall market depth and search cost (i.e., aggregate liquidity measure)

Figure 7.12: Simulation results for 600, 4000, and 10000 market participants

ber of differentiated resources, we conclude that the standardized approach will always achieve a lower or equal value of the overall market depth when compared to the differentiated approach.

When comparing the overall market depth for the differentiated and the standardized products, an interesting result occurs. As shown in Figure 7.12a, the depth for the standardized approach, although always lower than for the differentiated approach, significantly rises with the number of market traders. Considering the values of the left-hand graph, we conclude that the standardized approach achieves the maximum of 57.6% of the depth value of the differentiated approach when there are 600 active traders in the market. This value is achieved with 41 standardized products, which is 7.3 times less than the number of products in the differentiated market. When the number of market participants is increased to 10000 (the right-hand graph), the standardized approach achieves up to 93.4% of the overall market depth of the differentiated approach, although the number of standardized resources increased only slightly. In this case, when the maximum depth is achieved there are 86 standardized resources in the market which is almost 58 times less than the number of differentiated products. The main reason for this behavior is diversity of resource types which is reasonably constant, no matter the number of active traders. Namely, with only a small number of traders, the number of different resource types in the market is large and the number of users requiring or offering one resource type is low. On the contrary, with a sufficiently large number of traders, the number of different resource types is only moderately higher, but with more users requiring a same resource type. This means that the number of standardized products needed to keep the overall market depth stable grows slowly with the number of market participants.

The discovery of the relatively constant diversity in resource types may lead to a conclusion that a fixed amount of standardized products is needed to achieve a certain level of market efficiency with the number of traders playing no role in its determination. This is, however, not the case for several reasons. First, as depicted in Figure 7.12a, after a certain amount of standardized products is created, the growth of the overall market depth with every new standardized products decelerates because the currently existing standardized products already reflect the needs of most of the traders. Secondly, as it will be soon explained, this growth is not sufficiently high to cover the expenses of the introduction of more standardized products.

Search cost

Figure 7.12b presents the effort needed to find a trading partner in the simulated market environment. On the contrary to the perspective of overall market depth, the differentiated approach is significantly inferior to the standardized approach when considering the search cost. For the differentiated goods, buyers must iterate through active sellers' offerings until they find a matching service. This means that the maximum search cost is

$$cost_{max}^{diff.} = no.\,buyers \times no.\,sellers. \tag{7.3}$$

In the standardized approach, all users (buyers and sellers) iterate through public SLA templates, which means that the maximum search cost is

$$cost_{max}^{st.} = (no.\,buyers + no.\,sellers) \times no.\,st.\,resources. \tag{7.4}$$

Since the number of buyers and sellers is always remarkably larger than the number of standardized resources, the effort needed in the differentiated approach is always larger when compared to the standardized approach. However, the realistic cost values are usually notably lower than the theoretical maximum in both the differentiated and the standardized approach for two reasons. Firstly, users usually find required services before iterating through the whole list of available resources. Secondly, in the differentiated approach, once an allocation occurs, the buyer's request and the seller's offer are not considered in future iterations, which means that the number of active (i.e., non-allocated) users is reduced. Nevertheless, search cost grows with the number of market participants and the quantity of standardized products. Since sellers appear in the market much faster and in larger quantities than the standardized resources, the positive effects of the standardization becomes greatly obvious with the increasing number of market participants.

In the differentiated approach, the growth of the search cost slows down with the increase in the number of active market traders. For instance, when there are 600 traders in the market, the measured cost is 27% of the maximum possible cost (Equation (7.3)). On the contrary, when there are 10000 market traders, the value of the search cost is only 9% of the maximum cost. This is due to a large number of traders, which means that they can easily find a trading partner (due to the diversity in offerings) and have to perform a smaller number of search iterations. Similarly, the increase in the number of standardized products slows down the rise of the search cost in the standardized approach since users can easily find an appropriate service.

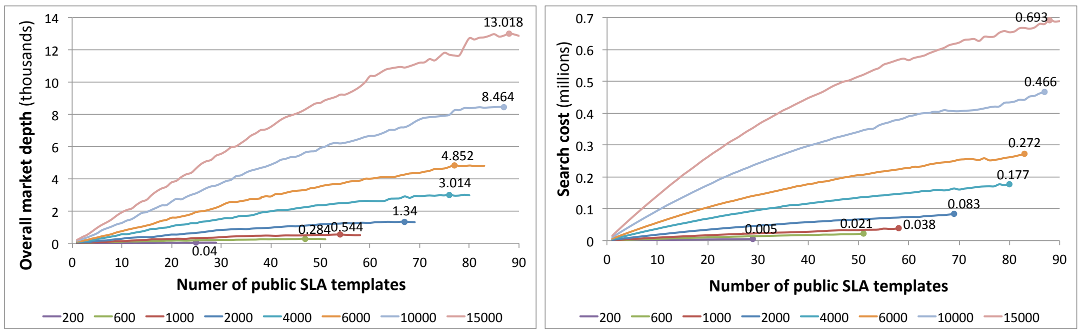
Aggregate liquidity measure

The conclusion that can be drawn from the previous analysis of standardized vs. differentiated approach in terms of overall market depth and search cost is that the standardization of services facilitates the search for the appropriate services (i.e., it minimizes the cost necessary to perform that action), but provides a lower chance to find a match for a certain service requirement or an offering. It stays unclear, however, which of the two approaches creates a better trading environment, as each of them provides a better performance from one of the two aspects of liquidity. The doubt that remains is, therefore, whether the lower overall market depth of the standardized approach compensates for the low search cost and whether the standardization improves the overall market liquidity.

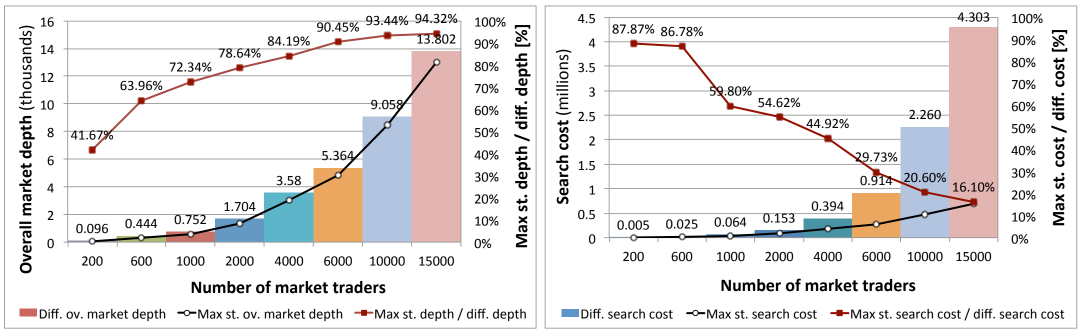
To answer this question, we shortly revisit and recapitulate simulation results from the previous sections. We visualize the comparison of the two market types in Figure 7.13 and present summarized simulation results of the overall market depth and search cost for the differentiated and standardized goods markets. Figure 7.13a depicts the simulation results for the standardized goods market with various number of market participants: the left-hand graph depicts the overall net utility and the right-hand graph the search cost. The graphs also emphasize the points in which the liquidity values reach the maximum values. Note that these maximum values can be even higher if more standardized resources were created. Nevertheless, as it will be soon discussed, we are not interested in these scenarios as they deviate from the "optimal setting".

The problem of finding the optimal market setting, i.e., the number and structure of SLA templates that maximizes the "overall" market liquidity, becomes evident in Figure 7.13b, which represents the simulation results of liquidity measures for the differentiated goods market (the values depicted as bars) and compares these results to the ones retrieved from the standardized goods market (the red line). The graph also depicts the maximum liquidity values in the standardized goods market (the black line), which are the same values emphasized in Figure 7.13a. The left-hand graph clearly demonstrates that the differentiated goods market always outperforms the standardized goods market in terms of overall market depth. This means that the differentiated goods market always offer a higher probability of finding a match. On the other hand, the right-hand graph shows that the cost of finding a match is much higher. However, as already discussed, a well-formed market is expected to fulfill both requirements: a solid market depth and a low search cost. Selecting a better trading environment between the differentiated and the standardized goods market is, therefore, not easy.

However, as it can be seen in Figure 7.13b, an increase in the number of market participants implies a significant reduction in the difference of the overall market depth, and a significant decrease in the difference of the search cost between the markets. In particular, the standardized goods market achieves 41.67% of the differentiated goods market depth with 200 active traders



(a) Market liquidity in the standardized goods market



(b) Market liquidity in the differentiated goods market

Figure 7.13: Summarized simulation results of market liquidity

in the market and a much higher value of 94.3% with 15000 traders, although the number of standardized resources increased only slightly. Similarly, the search cost goes down from 88% of the total search cost in the differentiated goods market for 200 market traders to only 16% for 15000 market traders. A conclusion that can be drawn from this behavior is that the increase in the number of market traders increases the justification of our approach.

To facilitate this discussion, we use the aggregate liquidity measure (Equation (4.16)). Figure 7.12c depicts the values of the aggregate liquidity measure and shows that the positive effects of the standardization become present only with a sufficiently high number of market participants. In a market with a limited amount of traders, a high diversity in resource types is distributed among a low number of market users and, as already explained, many standardized products are needed to achieve even a moderate overall market depth. At the same time, a large amount of standardized products increases the cost of searching for the most appropriate service offering. With the increase in the number of buyers and sellers, every new standardized product brings more to the overall market depth, but increases the search cost only slightly. Therefore, the more participants are in the market, the more benefits the standardization brings.

The question of how large the positive effect of the standardization to the market liquidity is can be answered by observing the results depicted in Figure 7.14. The figure compares the two approaches by looking at the relative difference between the aggregate liquidity measures of the

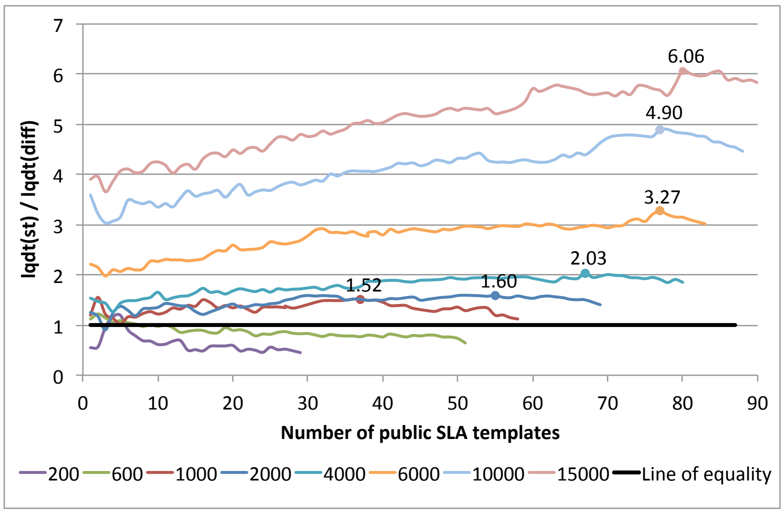


Figure 7.14: Standardized vs. differentiated approach in terms of aggregate liquidity measure

approaches, i.e.,

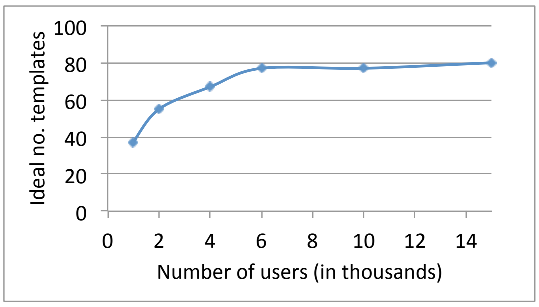
$$l = \frac{depth^{st.}/cost^{st.}}{depth^{diff.}/cost^{diff.}} = \frac{depth^{st.}}{depth^{diff.}} \times \frac{cost^{diff.}}{cost^{st.}}. \tag{7.5}$$

Essentially, Figure 7.14 shows how many times the value of the aggregate liquidity measure in the standardized approach is higher than the value in the differentiated approach. The graph presents the measured values for the various number of market participants (in the legend depicted in the form of *no. buyers* + *no. sellers*) and the "line of equality", which represents the value in which the two trading approach behave equally in terms of market liquidity (i.e., when the value of the aggregate liquidity measure is 1).

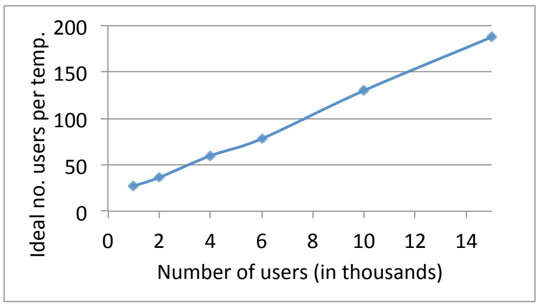
Figure 7.14 emphasizes the limited performance of the standardized approach with the low number of traders, but also its great outperformance when the number of traders is sufficiently high. In the demonstrated scenario, the standardized approach achieves up to 6 times higher amount of "aggregate liquidity" with 15000 market participants, which is achieved by creating only 80 standardized products. This result is important and noteworthy as we demonstrated, using a simulation scenario, that the standardization of goods in small markets may even hurt the market efficiency and stability. On the contrary, it brings enormous savings and benefits in a market where the demand and the supply are sufficiently high.

7.2.5 Using liquidity to determine the "optimal" market setting

In the previous section, we demonstrated the positive effects of the standardization of computational resources on market liquidity in (simulated) electronic markets. In this section, we continue this study and analyze the possibilities of using methods for approximation of liquidity to maximize the benefits of the standardized approach. In particular, we look into automatic estimation of the number of standardized products that, when introduced, increase the aggregate liquidity measure to its maximum point. Since market liquidity is the main performance



(a) “Ideal” number of SLA templates



(b) “Ideal” number of users per SLA template

Figure 7.15: Estimating the “ideal” number of standardized goods

indicator for the “benefit”, the number of products is “ideal” when the liquidity is maximized. Finding this number is, therefore, a matter of finding the market setting in which a single liquidity measure achieves its maximal value. However, as already described in Chapter 4, there is no universal measure of market liquidity. Overall market depth and search cost cannot be taken as the only measures of liquidity as they both must be considered in order to balance low search cost and high overall market depth. The aggregate liquidity measure (i.e., the relative difference between the depth and the cost) may help to find the “optimal” point, but cannot be taken into consideration individually either. To demonstrate this with an example, in the left-hand graph of Figure 7.12c it seems that, considering the value of the aggregate liquidity measure, the standardized approach outperforms the differentiated approach when there are only a few standardized products in the market. This, however, does not hold, as a user’s probability to find a trading partner at this point is almost nonexistent. A market with such a low matching probability has almost no benefits for buyers and sellers who would, in this case, almost certainly leave the market. Therefore, despite extremely low search costs that increase the aggregate liquidity measure, these results cannot be taken into consideration. However, as the aggregate liquidity measure is the closest we can get to the universal indicator of market liquidity, we address this issue by taking only those scenarios into consideration in which the overall market depth of the standardized approach reaches at least 50% of the value achieved by the differentiated approach. This step ensures at least a moderately satisfying outcome to the users. For the cases in which overall market depth is lower than this predefined threshold, we conclude that the standardization does not pay off and that only differentiated products should be traded.

Figure 7.15a depicts the number of standardized products created for a certain number of buyers and sellers in the market when the aggregate liquidity measure is maximized. Due to considering only the values where the overall market depth is sufficiently high, the scenarios with the lower number are not represented. The figure presents the market behavior already discussed in the previous: the “ideal” number of standardized products increases with a significantly slower pace than the number of traders. Moreover, the growth slows down with the increase in the number of traders. Therefore, after a certain number of standardized products are created, there are not many benefits of introducing additional products, no matter the number of market participants.

The usefulness of the market behavior depicted in Figure 7.15a is limited, as it can only be used to estimate the "ideal" number of standardized products in a low number of situations: when the number of sellers and buyers in the market is high enough, the introduction of additional products does not have a large effect on market liquidity. As already discussed, in this case the "ideal" number of standardized products stays constant. To avoid this limitation, we consider the same results from a different perspective. Figure 7.15b depicts the number of buyers and sellers per one standardized product in the market. The linear behavior of the graph shows that the diversity of resource types rises only slowly with the number of market participants, but the number of users using the same resource type increases. Therefore, in a market with more traders, the number of standardized products is not high, but the number of users per one product is. This measure does not only illustrate market behavior but may also help to estimate the "ideal" number of standardized products. In particular, the linear growth presented in Figure 7.15b can be estimated with the following equation:

$$no.\ users\, per\, resource = 0.12 \times no.\ users + 13.17 \tag{7.6}$$

The prediction probability is larger than 99% (i.e., with an R^2 value larger than 0.99). In statistics, the coefficient of determination R^2 is the proportion of variability in a data set that is accounted for by the statistical model ([191], pp. 187, 287). It provides a measure of how well future outcomes are likely to be predicted by the model. Note that in the given equation, $no.\ users$ represents the sum of the number of buyers and the number of sellers in the market.

With respect to Equation (7.6), the "ideal" number of standardized resources can be estimated using the following equation:

$$no. resources = \left\lfloor \frac{no. users}{0.12 \times no. users + 13.17} \right\rfloor \tag{7.7}$$

Note, however, that this result is valid only for the given trading scenario. The estimation function and the certainty depend on the demand and supply, i.e., the diversity in resource types in the market, and differ in other (real-case) scenarios. However, having a (reasonable) assumption that the diversity of users' requirements is relatively limited even in the real implementations of electronic markets, the same estimation method may be used in those environments, but with Equation (7.6) adapted to the observed demand and supply.

If properly used, the presented method, i.e., the appropriately adapted Equation (7.7), can be used to efficiently estimate the "ideal" number of standardized resources for every market situation. Namely, after having enough data to build the estimation function with a sufficiently high prediction probability, it is not necessary to check the varying number of standardized resources in order to find the "ideal" number, but it is possible to use the equation to quickly compute it. In order to allow a particular certainty, another method can be applied to confirm that indeed the number with the maximum aggregate liquidity measure has been selected. Namely, as the simulation scenario presented in the previous section has demonstrated, after the maximal aggregate liquidity value has been reached, the dynamics of the growth of overall market depth suddenly change: even if a larger quantity of standardized products is introduced, the depth stops rising or even loses its value (cf. Figure 7.12a). Therefore, it is possible to check whether any close number of standardized resources larger than the chosen "ideal" number increases the overall market depth and, thus, validate the estimated quantity.

7.3 Automatic service discovery and matching

In this section, we present the evaluation of our methods for automatic matching of SLA elements and selection of best-fitting SLA templates, as described in Chapter 5. In particular, we discuss how this approach may help to additionally decrease (or even completely eliminate) costs for market users remained after application of the standardization approach. Through this discussion we address Research question 4 presented in Chapter 1.

7.3.1 Simulation environment

In Chapter 6, we presented our framework for simulation of cloud marketplaces and SLA management. For the evaluation purposes, we design a testbed using this framework, which we shortly describe here. Figure 7.16 depicts another perspective of our simulator with the focus on the methods for SLA discovery and selection. From this point of view, our comprising two core components: a *cloud market platform* and a *simulation engine*.

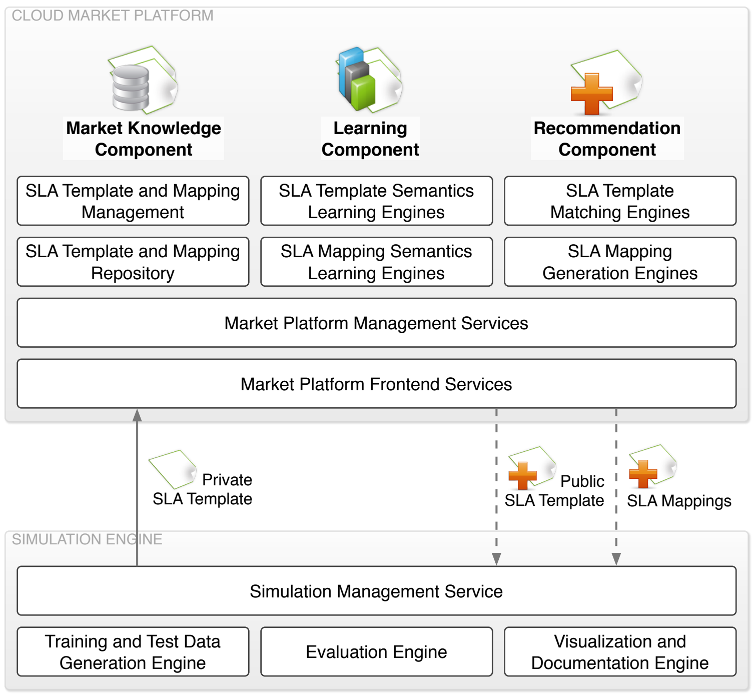

CLOUD MARKET PLATFORM
Market Knowledge Component
Learning Component
Recommendation Component
SLA Template and Mapping Management
SLA Template Semantics Learning Engines
SLA Template Matching Engines
SLA Template and Mapping Repository
SLA Mapping Semantics Learning Engines
SLA Mapping Generation Engines
Market Platform Management Services
Market Platform Frontend Services
Private SLA Template
Public SLA Template
SLA Mappings
SIMULATION ENGINE
Simulation Management Service
Training and Test Data Generation Engine
Evaluation Engine
Visualization and Documentation Engine

Figure 7.16: Simulation environment

The **cloud market platform** represents the basic infrastructure for autonomic management of the cloud market and integrates the actual implementation of our framework for autonomic creation and management of SLA mappings. In particular, it comprises: (1) frontend services, which form the basic interface for submission of users' requirements and offers for services to the market; (2) market platform management services responsible for management of supply and demand in the market (including pricing and allocation mechanisms); (3) a market knowledge component responsible for storing and managing public and private SLA templates as well as SLA mappings; (4) a learning component responsible for capturing characteristics of the data and learning to automatically recognize complex patterns and make intelligent decisions based on the data concerning the semantics of SLA elements and SLA templates; and (5) a recommendation component responsible for making final decisions on SLA matching and provider selection and recommendation of equal SLA elements, SLA mappings, and SLA templates.

The **simulation engine** (i.e., simulation controller) is responsible for automating the simulation process. In particular, it comprises: (1) a training and test data generation engine, implementing a configurable interface for semi-automatically generation of training and test data sets in compliance with predefined generation policies; (2) an evaluation engine, providing methods for assessing the results of the simulation by using an adequate cost model; and (3) a visualization and documentation engine, facilitating graphical visualization of the evaluation results as well as documentation for later analysis.

7.3.2 Simulation process

The simulation is conducted in three main steps (Figure 7.17): generation of demand and supply in the market, recommendation of the optimal offerings on the market to the service buyers (including the recommendation of SLA mappings between the differing SLA templates), and the evaluation of the recommendation results based on the simulated user feedback. In the first step, the training and the test data is randomly created in a semi-automatically fashion so that it meets requirements specified in data generation policies. These policies are forwarded to the data generation engine to ensure automatic generation of training and test datasets that meet specific properties of real datasets such as a clear separation between semantically different and semantically equal SLA templates or data distributions within the datasets. Moreover, the data generation engine contains a set of rules for creating instances of SLA elements based on characteristics of real-world examples as well as rules for modeling syntax differences between semantically equal SLA elements that can be found in real-world scenarios. The result of this process is the collection of triples containing a public SLA template from the set of all public SLA templates, a private SLA template from the set of all private SLA templates that is semantically equal to the associated public SLA template, and the pre-calculated SLA mappings between the two templates.

The second step of the simulation process involves submission of the previously created SLA templates and SLA mappings to the market. Namely, for each triple, the simulation engine submits the private SLA template to the market platform prototype. The prototype's recommendation component then automatically analyzes the submitted private SLA template and tries to match it with an existing public SLA template stored in the market repository contained by the market knowledge component. As already described in Chapter 5, this process finds the best matching public SLA template and returns it back to the simulation engine. Together with the public SLA template, the recommendation returns the SLA mappings between the discovered equivalent elements of the two SLA templates, as described in Chapter 5.

For the evaluation of the recommendation algorithm, the simulation engine analyzes correctness of the recommendation by comparing it to the precalculated public SLA template and SLA

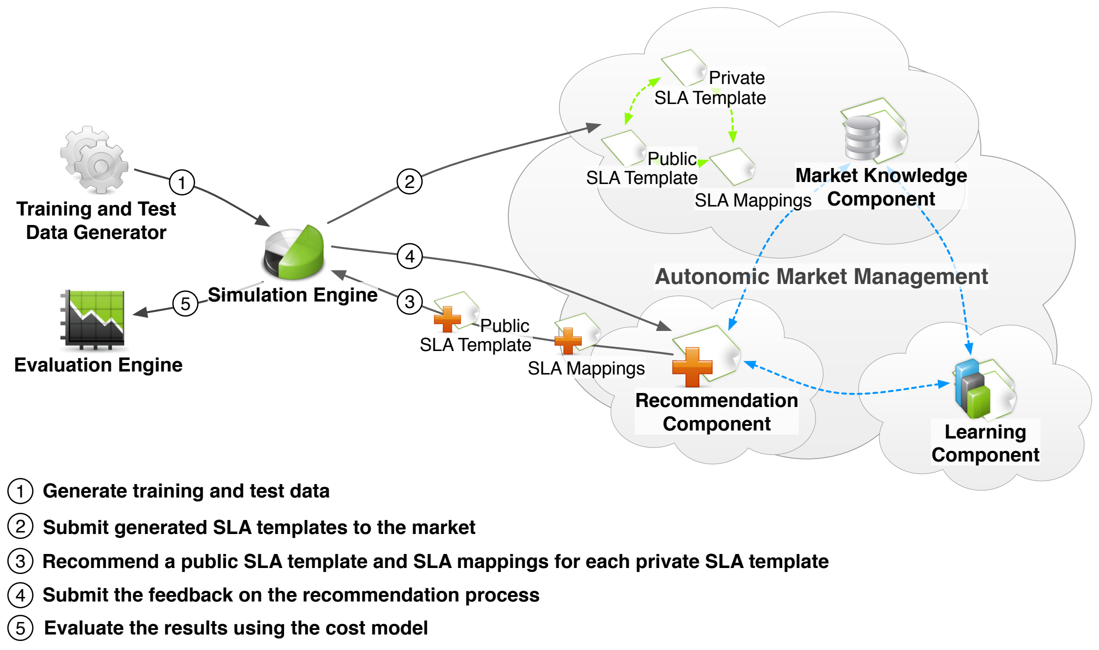

Training and Test Data Generator
Simulation Engine
Evaluation Engine
Private SLA Template
Public SLA Template
SLA Mappings
Market Knowledge Component
Autonomic Market Management
Public SLA Template
SLA Mappings
Recommendation Component
Learning Component
1 Generate training and test data
2 Submit generated SLA templates to the market
3 Recommend a public SLA template and SLA mappings for each private SLA template
4 Submit the feedback on the recommendation process
5 Evaluate the results using the cost model

Figure 7.17: Simulation testbed

mappings. If the recommended public SLA template is equal to the precalculated public template, the recommendation has been correct; otherwise an incorrect recommendation has been detected. By analogy, the correctness of the recommended SLA mappings is checked. In case of an incorrect recommendation, the framework reports the mistake to the market platform's recommendation component, thus simulating the negative user feedback. This feedback is then automatically forwarded to the learning component.

7.3.3 Simulation setup

For the evaluation of our approach, we define two simulation setups: for evaluating the process of provider selection (i.e., public SLA template recommendation) and for evaluating the methods for SLA mapping recommendation. Table 7.5 summarizes the simulation settings.

To evaluate the provider recommendation process, we generate a set of 100 public SLA templates and 3 semantically equal private SLA templates per one public SLA template. SLA templates contain between 5 and 7 SLOs as well between 5 and 7 SLA parameters. The number of SLA metrics per SLA template is in range of 7 to 10. Note that the number of SLA metrics is larger than the number of SLOs in oder to allow nesting of SLA metrics.

For evaluation of the SLA mapping recommendation process, the focus is on the syntax differences between elements of private SLA templates and semantically equal elements of corresponding public SLA templates. Therefore, only a small number of public SLA templates is necessary, but with a large variety of corresponding private SLA templates. For this purpose, we generate 3 public SLA templates with 100 semantically equal private SLA templates for each of the public templates. The settings for the structure of individual SLA templates is equivalent to the setup for public SLA template recommendation.

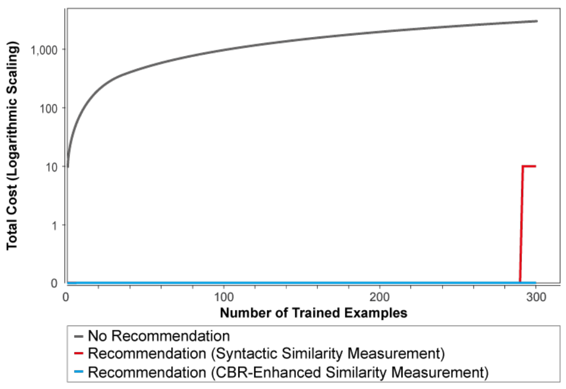
7.3.4 Evaluation results

In this section, we evaluate our approach by measuring the cost for market participants of adapting incorrect recommendations of public SLA templates and SLA mappings and compare it to the situation in which they have to search for optimal services and create SLA mappings manually. Furthermore, we mutually compare two recommendation strategies based on different learning methods: classification with input features based only on string similarity metrics and classification with input features based on string similarity metrics as well as those based on CBR knowledge.

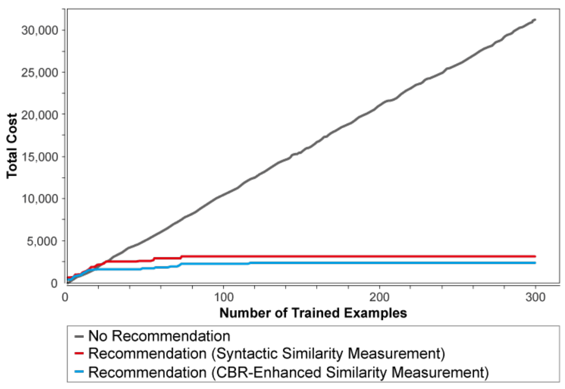
Figure 7.18a illustrates the evaluation results for automatic provider selection, i.e., recommendation of public SLA templates. It depicts the total cost in logarithmic scale of manual selection of public SLA templates achieved by different numbers of trained examples. As it can be seen, the usage of learning methods for matching SLA elements and for automatically associating and recommending public SLA templates significantly reduces the cost of manual association of public SLA templates by market participants, even when using less advanced learning methods, such as the classification method with input features based only on string similarity metrics. However, when comparing this "simple" classification method to the one with additional input features based on CBR knowledge, we can notice only a slight improvement in the cost reduction when compared to the latter case. This is due to the very small number of equivalent SLA elements that could not be matched by less advanced learning methods in comparison to the number of elements that could be matched by such methods, resulting in only a minor influence on the actual value of the overall equivalence probability between two SLA templates. Hence, even less advanced learning methods are able to show good results for automatic provider selection since not every single semantically equal pair of SLA elements between a public and a private SLA templates must be identified correctly, but even an approximation of the average similarity probability over all SLA elements of both templates is able to give enough information for making correct decisions about their semantic equivalence in most cases, espe-

Table 7.5: Simulation settings

Parameter	**Value**
Number of public SLA templates	
- Public SLA template recommendation	100
- SLA mapping recommendation	3
Number of private SLA templates per public SLA template	
- Public SLA template recommendation	3
- SLA mapping recommendation	100
Number of SLOs per SLA template	5-7
Number of SLA parameters per SLA template	5-7
Number of SLA metrics per SLA template	7-10
Maximal hierarchy level for nested SLA metrics	5



(a) SLA template recommendation (total cost development)



(b) SLA mappings recommendation (total cost development)

Figure 7.18: Evaluation results

cially when the number of pairs of semantically equivalent SLA elements with complex syntax differences is small in comparison to the total number of SLA elements within both templates.

Figure 7.18b illustrates the evaluation results for SLA mapping recommendation. It depicts the total cost of manual creation of SLA mappings achieved by different numbers of trained examples. The simulation starts with a high number of incorrect recommendations during the first 5 to 10 recommendation iterations. This is due to the initialization of SVM with random weights at the beginning of the simulation process and adjusting its weight function automatically over the following training iterations. After the SVM reaches a good approximation for the weight function, its predictions are relatively stable. The evaluation result shows that the usage of learning methods for matching SLA elements and for automatic generation and recommendation of SLA mappings significantly reduces the cost of manual creation of SLA mappings by market participants. When comparing the classification method with the input features based only on string similarity metrics to the one with additional input features based on CBR knowledge, we see that the latter can even reach a significantly lower cost than the former. This is due to the fact that CBR is able to detect complex difference patterns such as synonyms or abbreviations and therefore is able to create correct SLA mappings in such cases, while string similarity metrics are only able to detect small differences in a small number of characters and consider all other cases as semantically different. In our simulation we could show that the reductions in cost reached through the use of CBR are up to 30% after 20 iterations in comparison to the overall cost incurred when applying only string similarity metrics. However the concrete difference in cost between both methods ultimately depends on the degree of reoccurring patterns found in real-world scenarios, which may vary in different application domains.

Summarizing, as shown in the evaluation results, our approach for autonomic management of SLA mappings is able to significantly reduce market participants' cost of manual provider selection and manual creation of SLA mappings. The scope of reductions in these costs between the initial situation without recommendation and the approach proposed in Chapter 5 depends on the used SLA model, the assumptions made on the characteristics of SLA element specifications, and on the learning and reasoning strategies used. The assumptions we made on the characteristics of SLA element specifications are based on a theoretical research on possible characteristics of syntax differences between semantically equivalent SLA elements. These characteristics are limited on the one hand by the SLA model, which predefines the general structure for creating instances of SLAs, and on the other hand by the users' cognition of concepts that have the same meaning (linguistically and logically).

The evaluation has shown that even less advanced learning strategies are able to facilitate good results in automatic provider selection, while automatic creation of SLA mappings requires more sophisticated learning techniques since any inaccurate reasoning directly leads to a wrong recommendation and thus incurs unwanted cost for market participants. For manual provider selection, our approach was able to reduce market participants' cost by nearly 100% for all examples in comparison to its original amount without automatic recommendation. In other words, the automatically recommended provider is in almost all cases the correct one, i.e., the same provider that market participants would have chosen by manual selection. For automatic creation of SLA mappings, after a certain training period, our approach was able to reduce the market participants' cost by nearly 100% for examples that reused already trained knowledge in comparison to its original amount without automatic recommendation. As it could be seen in the evaluation results, the learning strategy based on CBR needs extensive training to be able to turn its advantages to account, but outperforms the less advanced learning strategy that we have tested in our evaluation since it is able to identify complex patterns and changes in SLA element and SLA mapping specifications.

7.4 Conclusion

In this chapter, we have presented the evaluations of the contributions that address the challenges based in this thesis. We have detailed the simulation environments within which evaluations were executed and demonstrated benefits that our approaches may bring. These benefits can be summarized as follows.

- As demonstrated in Section 7.1, our monitoring methodology is capable of detecting various market phenomena and can in future help to identify and react to sudden changes in the performance of cloud markets such that we can begin to give these platforms autonomic capabilities and enable them to steer away from and avoid negative market outcomes. Our simple evaluation scenario (a sudden cease in demand) illustrated how important it is to be able to adequately monitor a market where sudden changes can lead to painful consequences. Our scenario was very contrived, but aimed to test the feasibility of market monitoring. It temporarily affected the performance of all market goals both positively and negatively. In a real deployment, such a blip in performance could have resulted in excessive and costly utilization of arguably unneeded hardware infrastructure.

- As demonstrated in Section 7.2, our approach of computational resource standardization significantly improves market performance in terms of market liquidity and users' utility when sufficiently many users join the market. On the contrary, we have demonstrated that the standardization may hurt market liquidity if the number of market users is too low. Furthermore, we have identified the combination of k-means algorithm and the "maximum" adaptation method as the best performing for creating standardized resources in terms of several evaluation criteria, such as cluster isolation and compactness. We have also demonstrated that our algorithm for automated adaptation of SLA mappings significantly reduces the cost of market participation for market users. Finally, we have demonstrated that the measure of market liquidity may be successfully and simply used to create the "optimal market setting", i.e., to select the number of standardized resources that maximize market liquidity, due to the linear increase of "ideal" number of market users per a standardized SLA templates with the growth in the users population.

- In Section 7.3, we have demonstrated that our approach of automatic SLA matching and selection significantly reduces users' cost of participation in the market by recommending best-fitting public SLA templates to the market users, as well as SLA mappings to the index SLA template.

Regarding the credibility of the simulated approach with respect to the random creation of SLAs, note that the real-world "production" SLAs are currently very limited: they are used only for describing infrastructure services (i.e., in the infrastructure as a service (IaaS) business model), while for the other models such as platform as a service (PaaS) and software as a service (SaaS) they are still not utilized. The SLA parameters contained by the SLAs used in our experiments are simple modifications of common production SLAs used in the IaaS model. Nevertheless, it is noteworthy that our scenario differs from the real world in three factors:

1. Real-world SLAs are more complex than our assumed SLAs since they contain more SLA parameters and more differences in their definitions. However, the motivation for our approach of standardizing computational resources increases its significance with the additional complexity of users' SLAs. Therefore, we believe that our approach would demonstrate even better results with the real-world production SLAs than in the assumed scenario.

2. Unlike in our scenario, where the number of consumers and suppliers is equal, in the real-world cloud market, consumers currently significantly outnumber cloud providers.

3. In our scenario, we have considered equally distributed demand and supply as this increases market activity and improves visibility of effects of resource standardization on market liquidity. However, we believe that it does not affect the tendencies of the evaluation results as any other distribution will only impact the intensity of the market activity.

CHAPTER 8

Conclusion

In this chapter, we conclude this thesis by summarizing its contributions and their implications to the advancement of market-oriented resource allocation in utility and cloud computing.

8.1 Summary

One of the major challenges facing the cloud paradigm is the emergence of suitable economic platforms for the trading of cloud services. Today, many researchers investigate how specific cloud market platforms can be conceived and in some cases implemented. However, such endeavors consider only specific types of actors, business models, or cloud abstractions. We argue that market platforms for the cloud paradigm cannot (yet) be rigidly defined, and require the ability to progress and evolve with the paradigm. In this thesis, we have discussed an alternative approach: autonomic markets. Autonomic markets automatically adapt to changed environmental conditions based upon a given concept of "performance". We have described the autonomic MAPE-K loop in the context of electronic markets and considered the types of a knowledge produced and required for decision making.

As the entry point for enabling market adaptation, we have presented a novel methodology for the monitoring in cloud markets. Our methodology is built on the basis of identifying monitoring data that is available and useful for autonomic markets, and transforming this data into indicators for a given set of market goals. We have extended GridSim with appropriate market and mechanisms sensors, as well as simple infrastructure sensors. Our methodology included a series of realistic market goals, the sets of extractable metrics from a market platform, and rules for combining these metrics and transforming them to access goal performance. Our simple evaluation scenario (a sudden cease in demand) illustrated how important it is to be able to adequately monitor a market where sudden changes can lead to painful consequences. Our scenarios was very contrived, but aimed to test the feasibility of market monitoring. It temporarily affected the performance of all market goals both positively and negatively. In a real deployment, such a blip in performance could have resulted in excessive and costly utilization of arguably unneeded hardware infrastructure. We have shown that such phenomena can be detected by our monitoring model, which may in the future help to identify and react to sudden changes in the performance of cloud markets such that we can begin to give these platforms autonomic capabilities and enable them to steer away from and avoid negative market outcomes. We note and stress, however, that much research is still needed in order to explore how more subtle scenarios affect a given set of market goals. Nevertheless, we conclude that based upon the monitoring metrics of the market (which are translated from the low-level infrastructure measurements), our monitoring model can sense dynamic changes in market behavior, which is the first step towards establishing self-aware autonomic market platforms.

In this thesis, we have also discussed another perspective of market dynamism and heterogeneity and their effects on resource management and allocation. Using a simulation-based environment, we have demonstrated that a high resource diversity may hurt market performance in terms of market liquidity, users' utility, and transaction costs. In order to do that, we have first derived a set of assessment measures for monitoring market liquidity in cloud markets based on the literature study on liquidity in financial markets.

To address the problem of market diversity, we have introduced another form of market adaptation: creation and adaptation of standardized services. In this approach, demand and supply are channeled into a limited number of standardized services that maximize users' utility and market liquidity. To create standardized services, we have applied clustering algorithms to group similar service requirements and offerings, and adaptation methods to create one standardized service for the given group of users' SLAs. Instead of searching through the complete database of differentiated and fragmented providers' offerings, service consumers can now search only through a limited number of standardized services. Knowing the standardization process, users know what to expect in the market and can modify their service requirements. Although their satisfaction is never as high as in the differentiated goods market, the search cost is almost incomparably lower and market liquidity higher. However, this is not always the case. Using a simulation-based evaluation, we have demonstrated that standardization of computational goods in small markets (i.e., small number of market participants) may hurt market efficiency, but it brings enormous savings and benefits in the markets where the demand and the supply of partly substitutable goods are sufficiently high.

To additionally make the market more attractive to potential consumers and providers, we have also presented methods for automated service discovery and selection. When done manually, these processes take big effort and are often time-consuming. In this thesis, we apply several feedback-oriented machine learning methods to automatically find matching SLA specifications of services and select the best fitting service. Trading in the autonomic market becomes, therefore, virtually effortless.

Finally, in order to test our methods and hypotheses presented in this thesis, we have developed and herewith discussed conceptual and implementation details of an open-source agent-based market simulation framework. This platform provides capabilities for running flexible and complex simulations in dynamic, but highly controllable market scenarios, which is necessary for testing different aspects for the study of autonomic markets.

8.2 Constraints on thesis contributions

In this section, we describe the limitations of the research contributions achieved within this thesis. These issues are important for proper understanding of the proposed solutions and they highlight observations that are out of scope in our considerations.

- Our vision of autonomic markets discussed in Chapter 3 is presented only on a conceptual level. We have identified market functioning and adaptation steps, but have not yet proven that their construction and real-world implementation would be viable, nor how it could be performed. However, through our discussion of adaptation steps, we have noted the complexity of this process and identified that much research is still needed in order to fulfill the idea of autonomic online adaptation of market configuration.

- The monitoring methodology presented in this thesis does not fully capture market's economical footprint. In particular, the list of monitoring units and market performance metrics presented in Chapter 3 is not complete and is, therefore, not able to address all aspects of market performance. In order to achieve this goal, it is necessary to consider several other important metrics, such as volatility and solvency. Additionally, as already discussed in Chapter 7, our evaluation scenario (a sudden cease in demand) is contrived and simple. Despite this fact, it has helped us illustrate how important it is to be able to adequately monitor the market. However, in order to fully evaluate our monitoring framework, it is necessary to design, simulate and evaluate more complex and realistic market scenarios. For example, one could investigate cartel formation in the market. Namely, it could be observed whether the monitoring framework is able of detecting market peculiarities such as this one and, more importantly, if it is able of distinguishing providers joining a cartel and, therefore, and ensuring market competitiveness.

- In this thesis, public SLAs and users' SLA templates have been created by applying simple modifications to the real-world "production" SLAs used in the infrastructure as a service (IaaS) model. To facilitate our simulations, we have simplified the real-world SLAs by reducing the number of SLA parameters and the differences in their definitions. However, we believe that the benefits of our approach increase with the complexity of the underlying properties such as SLAs and the number of market participants, as discussed in Chapter 7. Nevertheless, more realistic SLAs should be used and this assumption should be evaluated. As another limitation, in our simulations we have used balanced demand and supply, i.e., a market with the equal number of service providers and service consumers. This does not realistically describe the current market landscape and does not comply with the idea of indefinitely large resources of cloud providers, but it has helped us in simplification of our simulation process and understanding evaluation results. We believe that the benefits of our approach are not affected by this market property. Nevertheless, other trading scenarios and demand-supply ratios should be considered as well. Finally, it should be considered how the adaptive standardized services could be adapted to the existing market models and how they can be traded and allocated through common market mechanisms, such as well-known auction-based mechanisms (e.g., continuous double auction and English auction).

- In the course of the thesis, we have discussed several steps of the SLA management process, which include standardization (Chapter 4) and SLA matching (Chapter 5). Throughout this process, we have mostly (and sometimes solely) considered functional parameters of computational resources without taking their non-functional properties into account. Non-functional properties (NFPs) are many and varied and include overall description of quality of a delivered service. The examples of such NFPs in cloud services are privacy, reliability, security, supportability, and trust. We acknowledge the importance of NFPs in SLA management, and especially in service selection. However, NFPs are hardly quantifiable and measurable and often not very well understood and interpreted. Assessment of these properties relies on previous user experiences, user perceptions and opinions, and cannot be easily captured from market monitoring and observations. In the work presented in this thesis, SLA management actions (especially SLA matching) does not provide any recommendation systems and tools for matching and advocating these properties. Instead, only functional elements are automatically matched, and the final matching and selection (which includes NFPs) is left to the user. In the future, these properties will have to be considered as part of the automatic SLA matching as well.

8.3 Future work

As discussed in the previous section, it can be observed that some important issues are out of scope regarding our proposed solution in this thesis. These issues imply open research challenges in this area. In this section, we discuss these challenges and identify several possible research directions.

- As already mentioned in the previous section, SLAs used in our simulations are simplified: they contain less SLA parameters and a smaller variety in their definitions. Furthermore, methods for managing these SLA elements (e.g., the SLA matching and recommendation tools) consider only functional parameters and neglect non-functional properties of SLAs. In future, it is necessary to consider more realistic and, therefore, more complex and diverse SLAs and to gather, learn and process knowledge from these SLAs so that the recommendation systems can finally consider non-functional service properties as well.

- As previously mentioned, the idea of autonomic market platform is currently portrayed only on a conceptual level and much research is still needed in order to bring it closer to the reality. This includes further development of the monitoring sensors (e.g., inclusion of more detailed market performance metrics and tuning through more realistic simulation scenarios), building and managing different type of market knowledge and information, and identifying (and executing) potential adaptation steps in the market setting. The latter task is particularly complex and requires deeper knowledge and understanding of market functioning and environment, as well as further development of the market simulators for testing novel hypotheses and methods.

- As a first step towards broader utilization and implementation of our experimental market platform, it is necessary to build a well-designed benchmark scenario to test its implementation and correctness. For example, numerous theoretical and practical works in the field of economics have discussed and demonstrated through empirical studies how users behave in certain market situations and how this behavior affects the market behavior. One such study has been introduced in [99]. In this paper, the authors report market experiments in which human traders are replaced by "zero-intelligence" agents and discussed allocative efficiency of a double auction. The results presented in this paper have been revisited in [67], where the authors presented a simulation of the approach that achieved the same results. Therefore, since both the empirical and simulation studies have demonstrated the same results, this study can be rerun in our simulation scenario in order to benchmark its performance.

- Besides already implemented clearing and bidding mechanisms, as well as simulation scenarios and agents' bidding strategies, one could implement additional mechanisms that can be used in different scenarios in order to compare their performance in different setting. For example, one could conclude that certain allocation mechanisms behave differently when the demand is high and the supply low. Furthermore, one could conclude that the properties of a certain allocation mechanism are not optimal for a given simulation scenario. Every quantitative analysis and comparison of different mechanisms and methods represent a further step towards the vision of autonomic market and implementation of its most challenging loop phase - the adaptation.

Bibliography

[1] S. Abdelwahed, Jia Bai, Rong Su, and N. Kandasamy. On the application of predictive control techniques for adaptive performance management of computing systems. *IEEE Transactions on Network and Service Management*, 6(4):212 –225, 2009.

[2] Hal Abelson. *Architects of the information society: Thirty-five years of the laboratory for computer science at MIT*. MIT Press, 1999.

[3] Jörn Altmann, Costas Courcoubetis, and Marcel Risch. A marketplace and its market mechanism for trading commoditized computing resources. *Annals of Telecommunications*, 65:653–667, 2010.

[4] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/ec2/, Last accessed: May 2013.

[5] Amazon simple storage service (amazon s3). http://aws.amazon.com/s3/, Last accessed: May 2013.

[6] Yakov Amihud and Haim Mendelson. Asset pricing and the bid-ask spread. *Journal of Financial Economics*, 17(2):223 – 249, 1986.

[7] Mauro Andreolini, Michele Colajanni, and Riccardo Lancellotti. Assessing the overhead and scalability of system monitors for large data centers. In *Proceedings of the First International Workshop on Cloud Computing Platforms*, CloudCP '11, pages 3:1–3:7, New York, NY, USA, 2011. ACM.

[8] Danilo Ardagna, Gabriele Giunta, Nunzio Ingraffia, Raffaela Mir, and Barbara Pernici. Qos-driven web services selection in autonomic grid environments. In *In OTM Conferences*, pages 1273–1289, 2006.

[9] Danilo Ardagna, Barbara Panicucci, and Mauro Passacantando. A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 177–186, New York, NY, USA, 2011. ACM.

[10] Marcos Dias De Assunçao and Rajkumar Buyya. An evaluation of communication demand of auction protocols in grid environments. Technical report, Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2006.

[11] Aws elastic beanstalk. http://aws.amazon.com/elasticbeanstalk/, Last accessed: May 2013.

[12] Mark Baker and Garry Smith. Gridrm: A resource monitoring architecture for the grid. *Grid Computing—GRID 2002*, pages 268–273, 2002.

[13] Jean-Pierre Ban,tre and Daniel Le MÈtayer. Programming by multiset transformation. *Commun. ACM*, 36(1):98 – 111, 1993.

[14] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *IEEE International Conference on Web Services (ICWS'06)*, pages 63 –71, 2006.

[15] 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):2637–2665, 2003.

[16] Luciano Baresi and Sam Guinea. Towards dynamic monitoring of ws-bpel processes. In Boualem Benatallah, Fabio Casati, and Paolo Traverso, editors, *Service-Oriented Computing - ICSOC 2005*, volume 3826 of *Lecture Notes in Computer Science*, pages 269–282. Springer Berlin / Heidelberg, 2005.

[17] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[18] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt. Vge - a service-oriented grid environment for on-demand supercomputing. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 11–18, 2004.

[19] Hendrik Bessembinder and Herbert M Kaufman. A comparison of trade execution costs for nyse and nasdaq-listed stocks. *Journal of Financial and Quantitative Analysis*, 32(3), 1997.

[20] Harold Bierman. Measuring financial liquidity. *The Accounting Review*, 35(4):628–632, 1960.

[21] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive Name Matching in Information Integration. *Intelligent Systems, IEEE*, 18(5):16–23, 2003.

[22] M. Bilenko and R.J. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48. ACM, 2003.

[23] Ekkehart Boehmer, Gideon Saar, and Lei Yu. Lifting the veil: An analysis of pre-trade transparency at the nyse. *Journal of Finance*, 60(2):783–815, April 2005.

[24] Irena Bojanova and Augustine Samba. Analysis of cloud computing delivery architecture models. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 453–458. IEEE, 2011.

[25] S. Boßung, J. Grundy, and J. Hosking. Semi-independent Discovery of Mapping Rules to Match XML Schemas. *Department of Computer Science, The University of Auckland*, page 71pp, 2003.

[26] Harley Boughton, Pat Martin, Wendy Powley, and Randy Horman. Workload Class Importance Policy in Autonomic Database Management Systems. In *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 13–22. IEEE Computer Society, 2006.

[27] A. Boukottaya, C. Vanoirbeek, F. Paganelli, and O.A. Khaled. Automating XML Documents Transformations: A Conceptual Modelling Based Approach. In *Proceedings of the First Asian-Pacific Conference on Conceptual Modelling*, volume 31, pages 81–90. Australian Computer Society, 2004.

[28] I. Brandic, D. Music, S. Dustdar, S. Venugopal, and R. Buyya. Advanced QoS Methods for Grid Workflows based on Meta-Negotiations and SLA-Mappings. In *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, pages 1–10. IEEE, 2008.

[29] Ivona Brandic, Dejan Music, and Schahram Dustdar. Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services. In *Grids meet Autonomic Computing Workshop, in conjunction with the 6th International Conference on Autonomic Computing and Communications*, pages 1–8. ACM, 2009.

[30] Ivona Brandic, Dejan Music, and Schahram Dustdar. VieSLAF framework: Facilitating negotiations in clouds by applying service mediation and negotiation bootstraping, 2010.

[31] Ivona Brandic, Dejan Music, Philipp Leitner, and Schahram Dustdar. VieSLAF framework: Enabling adaptive and versatile SLA-management. In *Grid Economics and Business Models*, volume 5745 of *Lecture Notes in Computer Science*, pages 60–73. Springer Berlin / Heidelberg, 2009.

[32] Jim Brandt, Ann Gentile, Jackson Mayo, Philippe Pebay, Diana Roe, David Thompson, and Matthew Wong. Resource monitoring and management with ovis to enable hpc in cloud computing environments. In *Parallel &amp; Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[33] Ivan Breskovic, Jörn Altmann, and Ivona Brandic. Creating standardized products for electronic markets. *Future Generation Computer Systems*, 2012.

[34] Ivan Breskovic, Ivona Brandic, and Jörn Altmann. Maximizing Liquidity in Cloud Markets through Standardization of Computational Resources. In *Proceedings of the 7th International Symposium on Service Oriented System Engineering*, SOSE '13. IEEE Computer Society, 2013.

[35] Ivan Breskovic, Christian Haas, Simon Caton, and Ivona Brandic. Towards self-awareness in cloud markets: A monitoring methodology. In *Proceedings of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 81–88, Washington, DC, USA, 2011. IEEE Computer Society.

[36] Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. Towards autonomic market management in cloud computing infrastructures. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science*, CLOSER'11, pages 24–34, 2011.

[37] Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. Achieving market liquidity through autonomic cloud market management. In Ivan Ivanov, Marten van Sinderen, and Boris Shishkov, editors, *Cloud Computing and Services Science*, Service Science: Research and Innovations in the Service Economy, pages 91–107. Springer New York, 2012.

[38] Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Schahram Dustdar. Cost-efficient utilization of public sla templates in autonomic cloud markets. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing*, UCC '11, pages 229–236, Washington DC, USA, 2011. IEEE Computer Society.

[39] R. Buyya, D. Abramson, and J. Giddy. A Case for Economy Grid Architecture for Service-Oriented Grid Computing. In *10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In Conjunction with IPDPS*, 2001.

[40] R Buyya, D Abramson, and S Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[41] R. Buyya and A. Sulistio. Service and utility oriented computing systems: Challenges and opportunities for modeling and simulation communities. In *Annual Simulation Symposium*, pages 68 –81, 2008.

[42] R. Buyya, C.S. Yeo, and S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *The 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13. IEEE, 2008.

[43] Rajkumar Buyya. Gridsim: A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing. http://www.buyya.com/gridsim/, Last accessed: May 2013, 2013.

[44] Rajkumar Buyya, David Abramson, and Jonathan Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, volume 1, pages 283–289. IEEE, 2000.

[45] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13):1175–1220, 2002.

[46] Rajkumar Buyya, Manzur Murshed, David Abramson, and Srikumar Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost–time optimization algorithm. *Software: Practice and Experience*, 35(5):491–512, 2005.

[47] Rajkumar Buyya and Sudharshan Vazhkudai. Compute power market: Towards a market-oriented grid. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 574–581. IEEE, 2001.

[48] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599 – 616, 2009.

[49] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software - Practice and Experience*, 2010.

[50] Radu Calinescu and Marta Kwiatkowska. Using quantitative analysis to implement autonomic it systems. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 100–110, Washington, DC, USA, 2009. IEEE Computer Society.

[51] R.D. Callaway, M. Devetsikiotis, Y. Viniotis, and A. Rodriguez. An autonomic service delivery platform for service-oriented network environments. *IEEE Transactions on Services Computing*, 3(2):327 –331, 2010.

[52] Simon Caton, Ivan Breskovic, and Ivona Brandic. A conceptual framework for simulating autonomic cloud markets. In *Proceedings of the 3rd International Conference on Cloud Computing*, CloudComp '11, Washington DC, USA, 2012. IEEE Computer Society.

[53] Simon Caton and Omer Rana. Towards Autonomic Management for Cloud Services based upon Volunteered Resources. *Concurrency and Computation: Practice and Experience*, 23, 2011.

[54] Sivadon Chaisiri, Rakpong Kaewpuang, Bu-Sung Lee, and Dusit Niyato. Cost minimization for provisioning virtual servers in amazon elastic compute cloud. In *Proceedings of the 2011 IEEE 19th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '11, pages 85–95, Washington, DC, USA, 2011. IEEE Computer Society.

[55] George C. Chako, Jakub W. Jurek, and Erik Stafford. The price of immediacy. *The Journal of Finance*, 63(3):1253–1290, 2008.

[56] Clovis Chapman, Wolfgang Emmerich, Fermın Galán Marquez, Stuart Clayman, and Alex Galis. Elastic service definition in computational clouds. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 327–334. IEEE, 2010.

[57] Kyle Chard, Kris Bubendorfer, Simon Caton, and Omer Rana. Social cloud computing: A vision for socially motivated resource sharing. *IEEE Transactions on Services Computing*, 99(PrePrints), 2011.

[58] G. Cheliotis and C. Kenyon. Autonomic economics. In *IEEE International Conference on Electronic Commerce*, pages 120 – 127, june 2003.

[59] Wai-Khuen Cheng, Boon-Yaik Ooi, and Huah-Yong Chan. Resource federation in grid using automated intelligent agent negotiation. *Future Generation Computer Systems*, 26(8):1116–1126, October 2010.

[60] Yu Cheng, A. Leon-Garcia, and I. Foster. Toward an autonomic service management framework: A holistic vision of SOA, AON, and autonomic computing. *IEEE Communications*, 46(5):138 –146, 2008.

[61] Lskrao Chimakurthi and Madhu Kumar SD. Power efficient resource allocation for clouds using ant colony framework. *CoRR*, abs/1102.2608, 2011.

[62] P. Christen. Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 151–159. ACM, 2008.

[63] N. Chudasma and S. Chaudhary. Service Composition Using Service Selection with WS-Agreement. In *Proceedings of the 2nd Bangalore Annual Compute Conference*, page 21. ACM, 2009.

[64] Li Chunlin and Li Layuan. A distributed utility-based two level market solution for optimal resource scheduling in computational grid. *Parallel Computing*, 31(3):332–351, 2005.

[65] Kassidy P. Clark, Martijn Warnier, and Frances M. T. Brazier. An intelligent cloud resource allocation service - agent-based automated cloud resource allocation using micro-agreement. In Frank Leymann, Ivan Ivanov, Marten van Sinderen, and Tony Shan, editors, *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, pages 37–45. SciTePress, 2012.

[66] Dave Cli. Minimal-intelligence agents for bargaining behaviors in market-based environments. *Hewlett-Packard Labs Technical Reports*, 1997.

[67] D.; Cliff and J. Bruten. Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets. Technical Report HPL-97-141, Hewelett-Packard Laboratories, Bristol, UK., 1997.

[68] Dave Cliff and Janet Bruten. More than zero intelligence needed for continuous double-auction trading. *HP LABORATORIES TECHNICAL REPORT HPL*, 1997.

[69] Dave Cliff and Janet Bruten. Simple bargaining agents for decentralized market-based control. *HP LABORATORIES TECHNICAL REPORT HPL*, 1998.

[70] Cloud foundry. http://www.cloudfoundry.com/, Last accessed: May 2013.

[71] W.W. Cohen, P. Ravikumar, and S. Fienberg. A Comparison of String Metrics for Matching Names and Records. In *KDD Workshop on Data Cleaning and Object Consolidation*, volume 3, pages 73–78. Citeseer, 2003.

[72] Committee on the Global Financial System. How should we design deep and liquid markets? The case of government securities. In *CGFS Working Group Reports*, October 1999.

[73] Marco Comuzzi, Constantinos Kotsokalis, George Spanoudakis, and Ramin Yahyapour. Establishing and monitoring slas in complex service based systems. In *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, pages 783–790, Washington, DC, USA, 2009. IEEE Computer Society.

[74] Costas Courcoubetis, Manos Dramitinos, Thierry Rayna, Sergios Soursos, and George Stamoulis. Market mechanisms for trading grid resources. *Grid Economics and Business Models*, pages 58–72, 2008.

[75] Marcos de Assunção, Werner Streitberger, Torsten Eymann, and Rajkumar Buyya. Enabling the simulation of service-oriented computing and provisioning policies for autonomic utility grids. In *4th International Workshop on Grid Economics and Business Models*, pages 136–149, 2007.

[76] R.L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M.L. Maher, M.T. Cox, K. Forbus, et al. Retrieval, Reuse, Revision and Retention in Case-Based Reasoning. *Knowledge Engineering Review*, 20(3):215, 2005.

[77] A.H. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to Match Ontologies on the Semantic Web. *The VLDB Journal*, 12(4):303–319, 2003.

[78] Glen Dobson and Alfonso Sanchez-Macian. Towards unified QoS/SLA ontologies. In *Services Computing Workshops, 2006. SCW '06. IEEE*, pages 169 –174, September 2006.

[79] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. Duplicate Record Detection: A Survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[80] Vincent C Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. Low level metrics to high level slas-lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 48–54. IEEE, 2010.

[81] Vincent C Emeakaroha, Rodrigo N Calheiros, Marco AS Netto, Ivona Brandic, and César AF De Rose. Desvi: An architecture for detecting sla violations in cloud computing infrastructures. In *Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp'10)*, 2010.

[82] Vincent C Emeakaroha, Marco AS Netto, Rodrigo N Calheiros, Ivona Brandic, Rajkumar Buyya, and César AF De Rose. Towards autonomic detection of sla violations in cloud infrastructures. *Future Generation Computer Systems*, 2011.

[83] Enomaly: Elastic cloud computing platform. http://www.enomaly.com/, Last accessed: May 2013.

[84] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

[85] Ettk (emerging technologies toolkit) project by ibm. https://www.ibm.com/developerworks/wikis/display/ettk/ETTK/, Last accessed: May 2013.

[86] European energy exchange. http://www.eex.com/en/, Last accessed: May 2013.

[87] K. Fakhfakh, S. Tazi, K. Drira, T. Chaari, and M. Jmaiel. Implementing and Testing a Semantic-Driven Approach Towards a Better Comprehension Between Service Consumers and Providers. In *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, pages 183–188. IEEE, 2010.

[88] Michael Fleming. Measuring treasury market liquidity. *FRB of New York Staff Report*, 2001.

[89] Fluid operations. http://www.fluidops.com/, Last accessed: May 2013.

[90] Howard Foster and George Spanoudakis. Advanced service monitoring configurations with sla decomposition and selection. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1582–1589. ACM, 2011.

[91] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[92] Wei Fu and Qian Huang. Grideye: A service-oriented grid monitoring system with improved forecasting algorithm. In *Grid and Cooperative Computing Workshops, 2006. GCCW '06. Fifth International Conference on*, pages 5–12, 2006.

[93] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: An architecture for secure resource peering. In *ACM SIGOPS Operating Systems Review*, pages 133–148. ACM, 2003.

[94] Saurabh Kumar Garg, Rajkumar Buyya, and Howard Jay Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, 26(8):1344–1355, October 2010.

[95] Frank E Gillett, EG Brown, J Staten, and C Lee. Future view: the new tech ecosystems of cloud, cloud services, and cloud computing. *Forrester Research Paper*, 2008.

[96] F. Giunchiglia and P. Shvaiko. Semantic Matching. *The Knowledge Engineering Review*, 18(03):265–280, 2003.

[97] F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic Matching: Algorithms and Implementation. In *Journal on Data Semantics IX*, pages 1–38. Springer, 2007.

[98] Lawrence R. Glosten and Paul R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. Discussion Papers 570, Northwestern University, Center for Mathematical Studies in Economics and Management Science, August 1983.

[99] Dhananjay K Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–37, February 1993.

[100] David Goldreich, Bernd Hanke, and Purnendu Nath. The price of future liquidity: Time-varying liquidity in the us treasury market. *Review of Finance*, 9(1):1–32, 2005.

[101] Google apps. http://www.google.com/apps/, Last accessed: May 2013.

[102] Google compute engine.

[103] Anastasios Gounaris, Christos Yfoulis, Rizos Sakellariou, and Marios D. Dikaiakos. A control theoretical approach to self-optimizing block transfer in web service grids. *ACM Transactions on Autonomous and Adaptive Systems*, 3(2):6:1–6:30, 2008.

[104] Les Green. Service level agreements: an ontological approach. In *8th international conference on Electronic commerce*, ICEC '06, pages 185–194. ACM, 2006.

[105] Grid enabled performance analysis using stochastic logics. http://aesop.doc.ic.ac.uk/projects/grail/, Last accessed: May 2013.

[106] Grid markets project. http://www.lesc.ic.ac.uk/markets/, Last accessed: May 2013.

[107] Sanford J. Grossman and Merton H. Miller. Liquidity and market structure. *The Journal of Finance*, 43(3):pp. 617–633, 1988.

[108] Dan Gunter, Brian Tierney, Brian Crowley, Mason Holding, and Jason Lee. Netlogger: A toolkit for distributed system performance analysis. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*, pages 267–273. IEEE, 2000.

[109] Larry Harris. *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press, 2002.

[110] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons Inc, 1975.

[111] Joel Hasbrouck. The summary informativeness of stock trades: An econometric analysis. *Review of Financial Studies*, 4(3):571–595, 1991.

[112] Joel Hasbrouck. Assessing the quality of a security market: A new approach to transaction-cost measurement. *Review of Financial Studies*, 6(1):191–212, 1993.

[113] Joel Hasbrouck. Measuring the information content of stock trades. *The Journal of Finance*, 46(1):179–207, 2012.

[114] Linli He and Thomas R. Ioerger. Forming resource-sharing coalitions: a distributed resource allocation mechanism for self-interested agents in computational grids. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 84–91, New York, NY, USA, 2005. ACM.

[115] Terrence Hendershott, Charles M. Jones, and Albert J. Menkveld. Does algorithmic trading improve liquidity? *The Journal of Finance*, 66(1):1–33, 2011.

[116] Thomas A Henzinger, Anmol V Singh, Vasu Singh, Thomas Wies, and Damien Zufferey. A marketplace for cloud resources. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 1–8. ACM, 2010.

[117] Heroku. http://www.heroku.com/, Last accessed: May 2013.

[118] Fred Howell and Ross McNab. A discrete event simulation package for java with applications in computer systems modelling. In *1st International Conference on Web-based Modelling and Simulation*, January 1998.

[119] Hp cloud services. https://www.hpcloud.com/, Last accessed: May 2013.

[120] He Huang and Liqiang Wang. P&amp;p: A combined push-pull model for resource monitoring in cloud computing environment. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 260–267. IEEE, 2010.

[121] Roger D. Huang and Hans R. Stoll. Dealer versus auction markets: A paired comparison of execution costs on nasdaq and the nyse. *Journal of Financial Economics*, 41(3):313–357, July 1996.

[122] David Irwin, Jeffrey Chase, Laura Grit, Aydan Yumerefendi, David Becker, and Kenneth G Yocum. Sharing networked resources with brokered leases. In *Proceedings of the USENIX Technical Conference*, pages 199–212, 2006.

[123] Laxmikant V. Kale, Sameer Kumar, Mani Potnuru, Jayant DeSouza, and Sindhura Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *Proceedings of the 2004 International Conference on Parallel Processing*, ICPP '04, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.

[124] P. Karänke and S. Kirn. Service Level Agreements: An Evaluation from a Business Application Perspective. In *Proceedings of eChallenges*. Citeseer, 2007.

[125] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36:41–50, January 2003.

[126] Bastian Koller and Lutz Schubert. Towards autonomous SLA management using a proxy-like approach. *Multiagent Grid Syst.*, 3:313–325, August 2007.

[127] H. Köpcke, A. Thor, and E. Rahm. Evaluation of Entity Resolution Approaches on Real-World Match Problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493, 2010.

[128] K. Kritikos and D. Plexousakis. Requirements for QoS-Based Web Service Description and Discovery. *Services Computing, IEEE Transactions on*, 2(4):320–337, 2009.

[129] J.J. Laffont and D. Martimort. *The theory of incentives: the principal-agent model*. Princeton paperbacks. Princeton Univ. Press, 2002.

[130] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182, 2005.

[131] Tobias Langenberg. *Standardization and Expectations*. Lecture Notes in Economics and Mathematical Systems. Springer Verlag, 2005.

[132] G. Le Mahec, F. Desprez, D. Loureiro, and A. Muresan. Cloud computing resource management through a grid middleware: A case study with diet and eucalyptus. In *IEEE International Conference onCloud Computing (CLOUD '09)*, pages 151–154, Sept.

[133] Bu Sung Lee, Shixing Yan, Ding Ma, and Guopeng Zhao. Aggregating iaas service. In *SRII Global Conference (SRII), 2011 Annual*, pages 335–338. IEEE, 2011.

[134] Chonho Lee and J. Suzuki. An autonomic adaptation mechanism for decentralized grid applications. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 583 – 589, January 2006.

[135] Kevin Lee, Norman W. Paton, Rizos Sakellariou, and Alvaro A. A. Fernandes. Utility functions for adaptively executing concurrent workflows. *Concurrency and Computation: Practice and Experience*, 23(6):646–666, 2011.

[136] Haifei Li and Jun-Jang Jeng. Ccmarketplace: a marketplace model for a hybrid cloud. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, pages 174–183. IBM Corp., 2010.

[137] D. Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the 15th International Conference on Machine Learning*, volume 1, pages 296–304. San Francisco, 1998.

[138] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. Technical report, IBM Corporation, 2003. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[139] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[140] Zaigham Mahmood. Cloud computing: Characteristics and deployment approaches. In *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on*, pages 121–126. IEEE, 2011.

[141] Kanti V. Mardia and J. T. Kent. *Multivariate Analysis*. Academic Press, 1980.

[142] Market models for grid computing. http://aesop.doc.ic.ac.uk/projects/grid-market/, Last accessed: May 2013.

[143] Marketo cloud. http://www.marketo.com/, Last accessed: May 2013.

[144] Keith Ansel Marzullo. Maintaining the time in a distributed system: an example of a loosely-coupled distributed service (synchronization, fault-tolerance, debugging). PhD thesis, Stanford University, 1984. AAI8506272.

[145] Michael Maurer, Ivona Brandic, Vincent Chimaobi Emeakaroha, and Schahram Dustdar. Towards knowledge management in self-adaptable clouds. In *6th World Congress on Services (SERVICES-1)*, pages 527 –534, July 2010.

[146] Michael Maurer, Ivona Brandic, and Rizos Sakellariou. Simulating autonomic sla enactment in clouds using case based reasoning. In *ServiceWave 2010*, volume 6481 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2010.

[147] Michael Maurer, Ivona Brandic, and Rizos Sakellariou. Enacting SLAs in clouds using rules. In *Proceedings of the 17th International Conference on Parallel Processing*, Euro-Par'11, pages 455–466, Berlin, Heidelberg, 2011. Springer-Verlag.

[148] Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. Cost-benefit analysis of an SLA mapping approach for defining standardized cloud computing goods. *Future Generation Computing Systems*, 28(1):39–47, 2012.

[149] Rajat Mehrotra, Abhishek Dubey, Sherif Abdelwahed, and Weston Monceaux. Large scale monitoring and online analysis in a distributed virtualized environment. In *Engineering of Autonomic and Autonomous Systems (EASe), 2011 8th IEEE International Conference and Workshops on*, pages 1–9. IEEE, 2011.

[150] Peter Mell and Timothy Grance. The nist definition of cloud computing. *NIST special publication*, 800:145, 2011.

[151] Andreas Menychtas, Anna Gatzioura, and Theodora Varvarigou. A business resolution engine for cloud marketplaces. In *Proceedings of the Third International Conference on Cloud Computing Technology and Science*, pages 462–469, 2011.

[152] Microsoft office 365. http://www.microsoft.com/office365/, Last accessed: May 2013.

[153] Ashraf Bany Mohammed, Jön Altmann, and Junseok Hwang. Cloud computing value chains: Understanding businesses and value creation in the cloud. In Dirk Neumann, Mark Baker, Jörn Altmann, and Omer Rana, editors, *Economic Models and Algorithms for Distributed Systems*, Autonomic Systems, pages 187–208. Birkhäuser Basel, 2010.

[154] Debajyoti Mukhopadhyay, Falguni J. Chathly, and Nagesh N. Jadhav. Qos based framework for effective web services in cloud computing. *Journal of Software Engineering and Applications*, 5(11A):952–960, 2012.

[155] H. Muñoz, I. Kotsiopoulos, A. Micsik, B. Koller, and J. Mora. Flexible SLA Negotiation Using Semantic Annotations. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pages 165–175. Springer, 2010.

[156] Zsolt NÈmeth, Christian PÈrez, and Thierry Priol. Workflow Enactment Based on a Chemical Metaphor. In *SEFM*, pages 127–136, 2005.

[157] Dirk Neumann, Jochen Stösser, and Christof Weinhardt. Bridging the adoption gap-developing a roadmap for trading in grids. *Electronic Markets*, 18:65–74, February 2008.

[158] Jens Nimis, Arun Anandasivam, Nikolay Borissov, Garry Smith, Dirk Neumann, Niklas Wirström, Erel Rosenberg, and Matteo Villa. SORMA - business cases for an open grid market: Concept and implementation. In *Grid Economics and Business Models*, volume 5206 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin / Heidelberg, 2008.

[159] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009.

[160] M. O'Hara. *Market Microstructure Theory*. Wiley, Blackwell, Cambridge, 1995.

[161] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-Agreement Partner Selection. In *Proceedings of the 15th international conference on World Wide Web*, pages 697–706. ACM, 2006.

[162] Open grid forum. http://www.ogf.org/, Last accessed: May 2013.

[163] Oracle infrastructure as a service. http://www.oracle.com/us/products/engineered-systems/iaas/overview/, Last accessed: May 2013.

[164] Walamitien H. Oyenan and Scott A. Deloach. Towards a systematic approach for designing autonomic systems. *Web Intelligence and Agent Systems*, 8:79–97, January 2010.

[165] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38 –45, November 2007.

[166] David Pardoe, Peter Stone, Maytal Saar-Tsechansky, and Kerem Tomak. Adaptive mechanism design: a metalearning approach. In *Proceedings of the 8th international conference on Electronic commerce*, ICEC '06, pages 92–102. ACM, 2006.

[167] Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Cooperation in multi-organization scheduling. *Concurrency and Computation: Practice and Experrience*, 21(7):905–921, May 2009.

[168] P. Patel, A. Ranabahu, and A. Sheth. Service Level Agreement in Cloud Computing. In *Cloud Workshops at OOPSLA*, 2009.

[169] Norman W. Paton, Marcelo A. T. Aragão, Kevin Lee, Alvaro A. A. Fernandes, and Rizos Sakellariou. Optimizing utility in cloud computing through autonomic workload execution. *IEEE Data Engineering Bulletin*, 32(1):51–58, 2009.

[170] A.F. Rahimi and A.Y. Sheffrin. Effective market monitoring in deregulated electricity markets. *IEEE Transactions on Power Systems*, 18(2):486 – 493, May 2003.

[171] 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 13th IEEE/ACM International Conference on Grid Computing*, GRID '11, pages 85–94, Washington DC, USA, 2012. IEEE Computer Society.

[172] Ori Regev and Noam Nisan. The popcorn market. online markets for computational resources. *Decision Support Systems*, 28(1):177–189, 2000.

[173] Marcel Risch and Jörn Altmann. Enabling open cloud markets through WS-agreement extensions. In Philipp Wieder, Ramin Yahyapour, and Wolfgang Ziegler, editors, *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 105–117. Springer US, 2010.

[174] Marcel Risch, Jörn Altmann, Li Guo, Alan Fleming, and Costas Courcoubetis. The gridecon platform: A business scenario testbed for commercial cloud services. In *Proceedings of the 6th International Workshop on Grid Economics and Business Models*, GECON '09, pages 46–59, 2009.

[175] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio M Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Cáceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[176] Luis Rodero-Merino, Luis M. Vaquero, Victor Gil, Fermin Galan, Javier Fontan, Ruben S. Montero, and Ignacio M. Llorente. From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, 26(8):1226–1240, October 2010.

[177] Florian Rosenberg, Christian Platzer, and Schahram Dustdar. Bootstrapping performance and dependability attributes ofweb services. In *Web Services, 2006. ICWS'06. International Conference on*, pages 205–212. IEEE, 2006.

[178] Günter Rote. Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38:123–127, 1991.

[179] Andrew L Russell. Standardization in history: A review essay with an eye to the future. *The standards edge: Future generations*, pages 247–260, 2005.

[180] M. Salama, A. Shawish, A. Zeid, and M. Kouta. Integrated qos utility-based model for cloud computing service provider selection. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 45–50, 2013.

[181] Salesforce.com. http://www.salesforce.com/, Last accessed: May 2013.

[182] Paul Anthony Samuelson and William D. Nordhaus. *Economics*. McGraw-Hill/Irwin, 18. ed., internat. ed. edition, 2005.

[183] Laura Savu. Cloud computing: Deployment models, delivery models, risks and research challenges. In *Computer and Management (CAMAN), 2011 International Conference on*, pages 1–4. IEEE, 2011.

[184] Björn Schnizler, Dirk Neumann, Daniel Veit, and Christof Weinhardt. Trading grid services - a multi-attribute combinatorial approach. *European Journal of Operational Research*, 187(3):943–961, June 2008.

[185] Kwang Mong Sim. Agent-based cloud commerce. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*, pages 717–721. IEEE, 2009.

[186] Kwang Mong Sim. Towards complex negotiation for cloud economy. In *Proceedings of the 5th international conference on Advances in Grid and Pervasive Computing*, GPC'10, pages 395–406, Berlin, Heidelberg, 2010. Springer-Verlag.

[187] Gurmeet Singh, Carl Kesselman, and Ewa Deelman. Application-level resource provisioning on the grid. In *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, E-SCIENCE '06, pages 83–, Washington, DC, USA, 2006. IEEE Computer Society.

[188] SLA@SOI. http://sla-at-soi.eu/, Last accessed: May 2013.

[189] Vernon Smith. Microeconomic systems as an experimental science. *The American Economic Review*, 72(5):923–955, 1982.

[190] Vernon L. Smith. Experimental economics: Induced value theory. *The American Economic Review*, 66(2):274–279, 1976.

[191] R. G. D. Steel and J. H. Torrie. *Principles and Procedures of Statistics*. McGraw-Hill New York, 1960.

[192] Werner Streitberger and Torsten Eymann. A simulation of an economic, self-organising resource allocation approach for application layer networks. *Computer Networks*, 53:1760–1770, July 2009.

[193] Gunther Stuer, Kurt Vanmechelen, and Jan Broeckhove. A commodity market algorithm for pricing substitutable grid resources. *Future Generation Computer Systems*, 23(5):688–701, 2007.

[194] Anthony Sulistio, Uros Cibej, Srikumar Venugopal, Borut Robic, and Rajkumar Buyya. A toolkit for modelling and simulating data grids: an extension to gridsim. *Concurr. Comput. : Pract. Exper.*, 20:1591–1609, 2008.

[195] Symja - a java computer algebra system. http://code.google.com/p/symja/, Last accessed: May 2013.

[196] Leigh Tesfatsion. Agent-based computational economics: Growing economies from the bottom u. *Artificial Life*, 8 (1):55–82, 2002.

[197] Douglas Thain and Miron Livny. Building reliable clients and servers. In Ian Foster and Carl Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[198] The rackspace cloud. http://www.rackspace.com/, Last accessed: May 2013.

[199] H. Tianfield. Towards agent based grid resource management. In *IEEE International Symposium on Cluster Computing and the Grid*, volume 1, pages 590–597, May 2005.

[200] Brian Tierney, William Johnston, Brian Crowley, Gary Hoo, Chris Brooks, and Dan Gunter. The netlogger methodology for high performance distributed systems performance analysis. In *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, pages 260–267. IEEE, 1998.

[201] Tradecard. http://www.tradecard.com/, Last accessed: May 2013.

[202] V.X. Tran and H. Tsuji. A Survey and Analysis on Semantics in QoS for Web Services. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 379–385. IEEE, 2009.

[203] Tsunamic tech. inc. http://www.clusterondemand.com/, Last accessed: May 2013.

[204] Kurt Vanmechelen, Wim Depoorter, and Jan Broeckhove. Market-based grid resource co-allocation and reservation for applications with hard deadlines. *Concurr. Comput. : Pract. Exper.*, 21(18):2270–2297, December 2009.

[205] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[206] 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.

[207] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *Software Engineering, IEEE Transactions on*, 18(2):103–117, 1992.

[208] Web service level agreements (wsla) project. http://www.research.ibm.com/wsla/, Last accessed: May 2013.

[209] Guiyi Wei, Athanasios V. Vasilakos, Yao Zheng, and Naixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, 54(2):252–269, November 2010.

[210] Christof Weinhardt, Arun Anandasivam, Benjamin Blau, Nikolay Borissov, Thomas Meinl, Wibke Michalk, and Jochen Stößer. Cloud computing - a classification, business models, and research directions. *Business & Information Systems Engineering*, 1(5):391–399, 2009.

[211] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and analyzing influential factors of business process performance. In *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International*, pages 141 –150, sept. 2009.

[212] Joseph S. Wholey and Harry P. Hatry. The case for performance monitoring. *Public Administration Review*, 52(6):pp. 604–610, 1992.

[213] Windows azure. http://www.windowsazure.com/, Last accessed: May 2013.

[214] Richard Wolski, James S Plank, Todd Bryan, and John Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 8–pp. IEEE, 2001.

[215] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proc. NSDI*, volume 7, pages 11–13, 2007.

[216] Ws-agreement. http://www.ofg.org/documents/GFD.107.pdf, Last accessed: May 2013.

[217] Xensource. http://www.xen.org/, Last accessed: May 2013.

[218] Guofu Xiang, Hai Jin, Deqing Zou, Xinwen Zhang, Sha Wen, and Feng Zhao. Vmdriver: A driver-based monitoring mechanism for virtualization. In *Proceedings of the 2010 29th IEEE Symposium on Reliable Distributed Systems*, SRDS '10, pages 72–81, Washington, DC, USA, 2010. IEEE Computer Society.

[219] Lijuan Xiao, Yanmin Zhu, Lionel M Ni, and Zhiwei Xu. Gridis: An incentive-based grid scheduling. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 65b–65b. IEEE, 2005.

[220] Viktor Yarmolenko and Rizos Sakellariou. Towards increased expressiveness in service level agreements: Research articles. *Concurrency and Computation: Practice and Experience*, 19:1975–1990, September 2007.

[221] Chee Shin Yeo and Rajkumar Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience*, 36(13):1381–1419, 2006.

[222] Gong Yong, Yao Li, Zhang Wei-ming, Sha Ji-chang, and Wang Chang-ying. Methods for resource allocation via agent coalition formation in grid computing systems. In *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, volume 1, pages 295–300, October 2003.

[223] Huiping Zhang. Measuring liquidity in emerging markets. *National University of Singapore Working Paper*, 2009.

[224] S. Sarah Zhang, Martin Wagener, Andreas Storkenmaier, and Christof Weinhardt. The quality of electronic markets. In *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, HICSS '11, pages 1–10, Washington, DC, USA, 2011. IEEE Computer Society.

[225] Shuai Zhang, Shufen Zhang, Xuebin Chen, and Xiuzhen Huo. Cloud computing research and development trend. In *Second international conference on future networks*, pages 93–97, 2010.

[226] Gansen Zhao, Chunming Rong, Martin Gilje Jaatun, and Frode Eika Sandnes. Deployment models: Towards eliminating security concerns from cloud computing. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 189–195. IEEE, 2010.

[227] H. Zhao and S. Ram. Entity Identification for Heterogeneous Database Integration – A Multiple Classifier System Approach and Empirical Evaluation. *Information Systems*, 30(2):119–132, 2005.

[228] Jiantao Zhou, Shang Zheng, Delin Jing, and Hongji Yang. An approach of creative application evolution on cloud computing platform. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 54–58, New York, NY, USA, 2011. ACM.

[229] Zimory cloud computing. http://www.zimory.de/, Last accessed: May 2013.

APPENDIX A

Curriculum vitae

Ivan Brešković

Personal information

Date of birth	May 7th, 1986
Place of birth	Belgrade, Serbia
Citizenship	Croatian
E-mail	ivan.breskovic@tuwien.ac.at
Web	http://www.infosys.tuwien.ac.at/staff/breskovic/
Affiliation	Vienna University of Technology Distributed Systems Group Argentinierstraße 8 A-1040 Wien, Austria

Education

2010 - 2013	PhD Computer Science, Vienna University of Technology, Austria
2008 - 2010	MSc Software Engineering and Information Systems, University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia
2009	Computer Science, University of Vienna, Austria (within European Union Erasmus exchange program)
2005 - 2008	BSc Software Engineering, University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia

Employment

2010 - present	Research assistant: cloud computing, computational economics at Vienna University of Technology, Distributed Systems Group, Austria
2010	Software development: social networks and process management systems at Verein Studentensport, Austria
2008 - 2010	Research and development: multicore SIP compliant parsers and TTCN-3 script generators at Ericsson, Croatia
2007 - 2009	Security advisor within the +CERT.hr (Croatian National Computer Emergency Response) project at the Croatian Academic and Research Network (CARNet), Croatia

Honors and awards

2011	Best student paper award at the 9^{th} IEEE International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia
2011	Austrian grant for a research visit to Seoul National University from Vienna University of Technology, Institute for International Relations, Vienna, Austria
2011	International grant for a research visit to Karlsruhe Institute of Technology from Awareness Initiative: Self-Awareness in Autonomic Systems, Edinburgh, UK
2010	Best student paper award at the 33^{rd} International Convention MIPRO, Opatija, Croatia
2009	Best student project award for the work in the field of telecommunications (project: QimPrESS) from University of Zagreb in cooperation with Ericsson Nikola Tesla, Zagreb, Croatia
2009	European Union Erasmus scholarship for an exchange program at University of Vienna from European Commission, Zagreb, Croatia
2008 - 2010	National scholarship for academically outstanding students from the Croatian Ministry of Science, Education and Sports, Zagreb, Croatia
2006 - 2009	Scholarship for academically outstanding students from Ericsson Nikola Tesla, Zagreb, Croatia

Scientific activities

Research projects

- HALEY - Holistic Energy Efficient Approach for the Management of Hybrid Clouds, 2012 - present
  Vienna University of Technology, Distributed Systems Group, Austria
  http://www.infosys.tuwien.ac.at/linksites/haley/

- FoSII - Foundations of Self-governing ICT Infrastructures, 2010 - 2012
Vienna University of Technology, Distributed Systems Group, Austria
http://www.infosys.tuwien.ac.at/linksites/FOSII/
- PROTO: Multicore SIP parser, 2009
Ericsson Nikola Tesla, Croatia
- Q-ImPrESS - Quality Impact Prediction for Evolving Service-Oriented Software (EU FP7), 2008
Ericsson Nikola Tesla, Croatia
http://www.q-impress.eu/

Students advising

- Christoph Redl: *Autonomic Management of SLA Mappings in Cloud Markets* (MSc, 2012)
- Manuel Schalleck: *Cloud Market Simulator* (BSc, 2013)

Research visits

- Karlsruhe Institute of Technology, Karlsruhe Services Research Institute, Germany, July 2012
Topic: Making Self-Aware Electronic Markets a Reality: an Autonomic Methodology
- Seoul National University, College of Engineering, South Korea, December 2012
Topic: Achieving Market Liquidity through Autonomic Cloud Market Management
- Karlsruhe Institute of Technology, Karlsruhe Services Research Institute, Germany, May - June 2011
Topic: Towards Self-Awareness in Cloud Markets

Scientific talks

- *Maximizing Liquidity in Cloud Markets through Standardization of Computational Resources*. 7^{th} International Symposium on Service Oriented System Engineering (SOSE 2013), 26 Mar. 2013, San Francisco, California, USA
- *A Conceptual Framework for Simulating Autonomic Cloud Markets*. 3^{rd} International Conference on Cloud Computing (CloudComp '12), 25 Sep. 2012, Vienna, Austria
- *Towards Self-Awareness in Cloud Markets: A Monitoring Methodology*. 9^{th} IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC '11), 13 Dec. 2011, Sydney, Australia
- *Cost-Efficient Utilization of Public SLA Templates in Autonomic Cloud Markets*. 4^{th} IEEE International Conference on Utility and Cloud Computing (UCC '11), 8 Dec. 2011, Melbourne, Australia
- *Towards Autonomic Market Management in Cloud Computing Infrastructures*. 1^{st} International Conference on Cloud Computing and Services Science (CLOSER '11), 7 May 2011, Noordwijkerhout, The Netherlands

Other activities

- Program committee member: 3^{rd} *International Conference on Cloud Computing (CloudComp '12)*

- Reviewer for journals: *ACM Computing Surveys (ACM)*, *ACM Transactions on Internet Technologies (ACM)*, *Automated Software Engineering (Springer)*, *Computing in Science and Engineering (IEEE)*, *Concurrency and Computation: Practice and Experience (Wiley)*, *Future Generation Computer Systems (Elsevier)*, *IEEE Transactions on Parallel and Distributed Systems (IEEE)*, *IEEE Transactions on Services Computing (IEEE)*, *International Journal of the Physical Sciences (Academic Journals)*, *Scientific Programming (IOS Press)*

Publications

Refereed publications in conference proceedings

- **Ivan Breskovic**, Ivona Brandic, and Jörn Altmann. *Maximizing Liquidity in Cloud Markets through Standardization of Computational Resources*. 7^{th} International Symposium on Service Oriented System Engineering (SOSE 2013), 25-28 March 2013, San Francisco, USA

- Simon Caton, **Ivan Breskovic**, and Ivona Brandic. *A Conceptual Framework for Simulating Autonomic Cloud Markets*. 3^{rd} International Conference on Cloud Computing (CloudComp 2012), 24-26 September 2012, Vienna, Austria

- Christoph Redl, **Ivan Breskovic**, Ivona Brandic, and Schahram Dustdar. *Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets*. The 13^{th} ACM/IEEE International Conference on Grid Computing (Grid 2012), 20-23 September 2012, Beijing, China

- **Ivan Breskovic**, Christian Haas, Simon Caton, and Ivona Brandic. *Towards Self-Awareness in Cloud Markets: A Monitoring Methodology*. 9^{th} IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011), 12-14 December 2011, Sydney, Australia **(best student paper award)**

- **Ivan Breskovic**, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Schahram Dustdar. *Cost-Efficient Utilization of Public SLA Templates in Autonomic Cloud Markets*. 4^{th} IEEE International Conference on Utility and Cloud Computing (UCC 2011), 5-8 December 2011, Melbourne, Australia

- Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, and **Ivan Breskovic**. *SLA-Aware Application Deployment and Resource Allocation in Clouds*. The Second IEEE International Workshop on Emerging Applications for Cloud Computing (CloudApp 2011), In conjunction with the 35^{th} Annual IEEE International Computer Software and Applications Conference (COMPSAC 2011), 18-21 July 2011, Munich, Germany

- Michael Maurer, **Ivan Breskovic**, Vincent C. Emeakaroha, and Ivona Brandic. *Revealing the MAPE Loop for the Autonomic Management of Cloud Infrastructures*. Workshop on Management of Cloud Systems (MoCS 2011), in association with the IEEE Symposium on Computers and Communications (ISCC 2011), 28 June - 1 July 2011, Kerkyra (Corfu) Greece

- **Ivan Breskovic**, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. *Towards Autonomic Market Management in Cloud Computing Infrastructures*. International Conference on Cloud Computing and Services Science (CLOSER 2011), 7-9 May 2011, Noordwijkerhout, The Netherlands

- Davor Sutic, **Ivan Breskovic**, Rene Huic, and Ivan Jukic. *Automatic Evaluation of Facial Attractiveness*. Proceedings of the 33^{rd} International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2010), 24-30 May 2010, Opatija, Croatia **(best student paper award)**

Book chapters

- **Ivan Breskovic**, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Jörn Altmann. *Achieving Market Liquidity Through Autonomic Cloud Market Management*. In Cloud Computing and Services Science (Service Science: Research and Innovations in the Service Economy), pp. 91-110. Editors: Ivan Ivanov, Marten van Sinderen, Boris Shishkov. Springer, 2012
- Vincent C. Emeakaroha, Michael Maurer, **Ivan Breskovic**, Ivona Brandic, and Schahram Dustdar. *SOA and QoS Management for Cloud Computing*. In Cloud computing: methodology, system, and applications, pp. 277-300. Editors: Lizhe Wang, Rajiv Ranjan, Jinjun Chen, Boualem Benatallah. Taylor & Francis group, 2011

Technical reports

- **Ivan Breskovic**. *Towards Self-Awareness in Cloud Markets*. Aware Project - Self Awareness in Autonomic Systems. August, 2011
- Vincent C. Emeakaroha, Michael Maurer, **Ivan Breskovic**, and Ivona Brandic. *Time Shared VMs and Monitoring of Time Shared VMs*. Proceedings of the COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems, 2^{nd} Year, J. Pierson, H. Hlavacs (Ed.), COST Office, 2011, p. 47-51.
- **Ivan Breskovic**. *Extending Geospatial Database Management Systems with Moving Objects Support* (Master's thesis). University of Zagreb, Faculty of Electrical Engineering and Computing, July 2010
- **Ivan Breskovic**, Vedrana Jankovic, Zvonimir Pavlinovic, Zeljko Rumenjak. *Multicore SIP*. Ericsson Nikola Tesla, Zagreb, Croatia, 2009 **(best student paper in telecommunications)**