Hudert, Sebastian

Working Paper

The BabelNEG System - A Protocol-generic Infrastructure for Electronic SLA Negotiations in the Internet of Services

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik, No. 52

Provided in Cooperation with:
University of Bayreuth, Chair of Information Systems Management

Suggested Citation: Hudert, Sebastian (2011) : The BabelNEG System - A Protocol-generic Infrastructure for Electronic SLA Negotiations in the Internet of Services, Bayreuther Arbeitspapiere zur Wirtschaftsinformatik, No. 52, Universität Bayreuth, Lehrstuhl für Wirtschaftsinformatik, Bayreuth,
https://nbn-resolving.de/urn:nbn:de:bvb:703-opus-9287

This Version is available at:
https://hdl.handle.net/10419/73037

Lehrstuhl für
Wirtschaftsinformatik

Information Systems
Management

No. 52

October 2011

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Sebastian Hudert

The BabelNEG System - A Protocol-generic Infrastructure for Electronic SLA Negotiations in the Internet of Services

Bayreuth Reports on Information Systems Management

UNIVERSITÄT
BAYREUTH

ISSN 1864-9300

Die Arbeitspapiere des Lehrstuhls für Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The Bayreuth Reports on Information Systems Management comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Authors:

Sebastian Hudert (University of Bayreuth)

Information Systems Management Working Paper Series

Edited by:

Prof. Dr. Torsten Eymann

Contact:

Universität Bayreuth
Lehrstuhl für Wirtschaftsinformatik (BWL VII)
Prof. Dr. Torsten Eymann
Universitätsstrasse 30
95447 Bayreuth
Germany

Email: wirtschaftsinformatik@uni-bayreuth.de

ISSN 1864-9300

The BabelNEG System – A Protocol-generic Infrastructure for Electronic SLA Negotiations in the Internet of Services

Dissertation
zur Erlangung des Grades eines Doktors der Wirtschaftswissenschaften
der Rechts- und Wirtschaftswissenschaftlichen Fakultät
der Universität Bayreuth

Vorgelegt
von
Sebastian Hudert
aus
Schweinfurt

Dekan: Prof. Dr. Markus Möstl
Erstberichterstatter: Prof. Dr. Torsten Eymann
Zweitberichterstatter: Prof. Dr. Guido Wirtz
Tag der mündlichen Prüfung: 04. 10. 2011

Acknowledgments

The research presented in this doctoral thesis has been carried out during my assignment at the Department of Information Systems Management at the University of Bayreuth. Along the path towards this thesis, I was involved in several research projects and the academic life in general, providing me a very challenging but also stimulating work environment. I owe a debt of gratitude to many people who have assisted me along the way and influenced both my work and my understanding of the topic of electronic SLA management.

In particular, I wish to express my gratitude to my primary supervisor, Professor Dr. Torsten Eymann for the opportunity he gave me at his department. The discussions we had, his many conceptual and methodical suggestions and not least his constant encouragement fundamentally affected my work and crucially contributed to its success.

I am grateful to Professor Dr. Guido Wirtz from the Mobile and Distributed Systems Group at the Otto-Friedrich University Bamberg. He decisively influenced my way of assessing and designing distributed software infrastructures during his supervision of both my diploma and co-supervision of my doctoral thesis.

Furthermore I am deeply indebted to my colleagues at Bayreuth University for their relentless assistance with conceptual, technical and methodological problems along the way. Depending on their academic origin in computer science or business administration / economics they all provided me with valuable input and helped me understand the intricacies of the research area I worked in. In spite of all difficulties occurring in such a demanding project, they managed to create an almost homelike environment for which I am deeply grateful. In particular, I would like to thank Stefan König whom I had the honor to get to know at the very beginning of my undergraduate studies and work together ever since as well as Christoph Niemann for the many discussions on research methodology and conceptual help he accorded me. A special thanks goes to Axel Pürckhauer for his constant technical support of our work; me having been able to continuously work on my research is greatly owed to his efforts. I am proud to have been a member of this team and to be able call these people my friends.

Torsten Eymann always encouraged an exchange of ideas with the national and international research community. On many occasions he gave me the opportunity to visit conferences and workshops, give talks and in general meet researchers from all over the globe, working on similar topics. Among these, I owe much gratitude to Prof. Dr. Udo Krieger (Otto-Friedrich University Bamberg) for believing in me and acquainting me with Dr. Heiko Ludwig from IBM Research and the area of automated SLA negotiations in general. This opened many doors for me in the academic world. I consequently want to thank Dr. Ludwig for escorting me and my work for a long time and for the endless phone calls we had over the years, in which he consulted me in much more than just my

research topic. I am also very grateful to researchers I was lucky to meet within the collaborative projects I worked in, such as CATNETS, SORMA or eRep, as well as the ones I met at research forums, such as the Open Grid Forum or at Dagstuhl Castle. Among these I particularly owe Philipp Wieder (Georg-August University Göttingen), Dr.-Ing. Bastian Koller (High Performance Computing Center Stuttgart), Dr. Shamima Paurobally (University of Westminster), Prof. Omer F. Rana (Cardiff University), Dr.-Ing. Nicolas Repp (Technical University of Darmstadt) and Prof. Frances Brazier (Technical University of Delft) for their time and the valuable input they gave me for my work.

With all my heart I thank my wonderful family and friends. My parents Karin and Werner raised, supported, taught and loved me in a way I am but beginning to grasp and they still continue to do so. Without the unconditional help, patience and understanding received from them and my much valued brother Fabian this work would never have been possible. I am forever indebted to them for all they did for me throughout my life.

Finally my most heartfelt gratitude goes to Verena. Having met her at the beginning of my time in Bayreuth and coming to know her support, constant encouragement and her unconditional love and companionship, I consider the greatest godsend in my life.

Bayreuth, October 2011
Sebastian Hudert

Abstract

Visions of the next-generation Internet of Services are driven by digital resources traded on a global scope. For the resulting economic setting, automated on-line techniques for handling services and resources are needed, as well as for advertising and discovering or for the on-the-fly negotiation of proper terms for their use.

This thesis presents the results of my dissertation project. They comprise a service infrastructure, able to support the structured discovery and protocol-generic negotiation of electronic service level agreements (SLAs) and thus services themselves.

The need for such an artifact is deduced from a detailed scenario analysis, extrapolating past and current developments in distributed business information systems. Based on economic theory, the need for both negotiation processes as such and the possibility to adapt to different protocols at run time is inferred in a second step.

The requirements for my prototype system are then derived from the scenario model and underlying economic theories. I discuss conceptual foundations, comprising theoretical principles for the design, formulation, discovery, negotiation, and subsequent usage of (electronic) SLAs in distributed information systems.

After having presented these fundamental concepts, the actual infrastructure design, thus the proposed solution to the stated research problem, is detailed. The underlying idea is to decouple the good to be sold (the SLA) from the negotiation protocol, enabling a service provider to apply different negotiation protocols for the same service over time.

Furthermore, for the consumer side a protocol-generic negotiation component is designed, capable of automatically adapting to different protocols, as offered by the service providers. The conceptual copula between these two sides is a set of structured service description documents, defining not only the service-relevant functional and non-functional parameters, but also the applied negotiation protocol in a machine-readable way.

The last chapter presents a thorough assessment of the infrastructure design, including the actual implementation of the developed mechanisms and data structures in a Java-based simulation environment, the conceptual demonstration of the system's effectiveness (with regard to the stated requirements) as well as a simulative demonstration step proving the adaptability of the service consumer agents. With these assessment steps, the system's capability to fulfill all stated requirements could be shown, thus proving its effectiveness to solve the research problem.

Contents

List of Figures

List of Tables

Abbreviations

ACK	Acknowledgement (Message Type)
Akogrimo	Access to Knowledge through the Grid in a mobile World (Project)
AL	Application Layer
AO	Alternate Offers
ASAPM	Adaptive Service Agreement and Process Management (Project)
ASG	Application Services Grid (Project)
ASP	Application Service Provider
AssessGrid	Advanced Risk Assessment & Management for Trustable Grids (Project)
BEinGRID	Business Experiments in the GRID (Project)
BREIN	Business objective driven reliable and intelligent Grids for real Business (Project)
CATNETS	Catallaxy paradigm for decentralized operation of dynamic application networks (Project)
CC	Cloud Computing
CDA	Continuous Double Auction
CM	Call Market
COBIT	Control Objectives for Information and Related Technology
DA	Dutch Auction
DHT	Distributed Hash Table
DS	Design Science
EA	English Auction
EC	European Commission
EPR	Endpoint Reference

EST	Extended SLA Template
FIPA	Foundation for Intelligent Physical Agents
FPA	Fixed Price Auction
FPSB Auction	First-price-sealed-bid Auction
GB	Gigabyte
GC	Grid Computing
GGF	Global Grid Forum
GN	Generic Negotiator
GT	Guarantee Term
HPC4U	Highly Predictable Cluster for Internet-Grids (Project)
ID	Identifier
IETF	Internet Engineering Task Force
IL	Infrastructure Layer
IoS	Internet of Services
IS	Information Systems
ISP	Independent Service Provider
ITIL	IT Infrastructure Library
NACK	Non-Acknowledgement (Message Type)
NC	Negotiation Coordinator
NextGRID	Architecture for the Next Generation Grids (Project)
OASIS	Organization for the Advancement of Structured Information Standards
OCL	Object Constraint Language
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
OLA	Operational Level Agreement
P2P	Peer-to-Peer

QoS	Quality-of-Service
RA	Registry Agent
RDA	Reverse Dutch Auction
REA	Reverse English Auction
REPAST	Recursive Porous Agent Simulation Toolkit
RFC	Request for Comments
SC	Service Consumer
SDT	Service Description Term
SI	Service Identificator
SimIS	Simulating and Internet of Services (Toolkit)
SLA	Service Level Agreement
SLO	Service Level Objective
SLP	Service Location Protocol
SLS	Service Level Specification
SO	Service Orientation
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SORMA	Self-Organizing ICT Resource Management (Project)
SOS	Service Oriented System
SP	Service Provider
SRT	Service Reference Term
ST	Service Type
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
USD	US Dollar

WS	Web Service
WS-Agreement	Web Services Agreement
WS-Discovery	Web Services Dynamic Discovery
WSCM	Web Services Composition Management
WSLA	Web Service Level Agreement
WSOL	Web Services Offering Language
XML	Extensible Markup Language

1. Research Problem and Approach

> As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country.
>
> *(Leonard Kleinrock, UCLA press release announcing the launch of the ARPANET in 1969)*

1.1. Introduction and Motivation

Visions of 21st century's information systems (IS) show highly specialized digital services and resources, which collaborate continuously and with a global reach. Today's Internet of mainly human interactions evolves to a global, socio-technical information infrastructure, where humans as well as software agents, acting on their behalf, continuously interact to exchange data and computational resources. Possibly millions of service providers (SPs), consumers (SCs) and a multitude of intermediaries like brokers or workflow orchestrators are present, forming a global economic environment. This vision is commonly referred to as the Internet of Services (IoS) (Ruggaber 2007; Schroth and Janner 2007).

Building on currently applied computing paradigms, such as Service-oriented (SOC) (Foster 2005), Grid (GC) (Foster, Kesselman, and Tuecke 2001) or Cloud Computing (CC) (Buyya et al. 2009), the IoS vision defines highly dynamic networks of composable services, offered and consumed on demand and on a global scope. It rigorously focuses on the goal of an Internet-based service economy, similar to the real-world service sector. Digital services are offered over electronic service markets, purchased by respective customers and then combined with internal or other external services to business workflows of varying complexity. In that, it allows even very small and specialized companies to find a niche in the digital economy where they can compete with the ubiquitous international enterprises, which in turn have to face a much higher competition on the global market (Theseus 2009).

Economic success of the IoS crucially depends on new business models, as well as their supporting technical infrastructure, enabling trading processes down to the level of an individual service, and the subsequent charging based solely on its usage and delivered

quality-of-service (QoS). Such models imply the need for mechanisms guaranteeing QoS for each service invocation, even across enterprise boundaries.

Since a scenario like the IoS inherently lacks the applicability of centralized QoS management, guarantees must be obtained to this end in the form of bi- or even multi-lateral service level agreements (SLAs), assuring service quality across individual sites (Ludwig et al. 2003a).

In the following the results of my dissertation project, dealing with the automated management (more specifically, the discovery and negotiation) of electronic services and respective SLAs in the IoS, are presented.

1.1.1. Scenario Model

In this subsection a detailed scenario model, subsequently acting as the context for this thesis, is derived. For this purpose current trends in distributed IS are analyzed first. Building on those developments, a generic scenario model, anticipating the future IoS environment, can be induced in a second step.

Service Orientation and Service Oriented Architectures

In recent years a new paradigm of designing and implementing business IS has been established: Service Orientation (SO). The main idea behind this concept is that every functionality offered by humans, organizational entities or computer systems is considered an abstract *service*, each of which can again be combined with others to create more complex composite services.

Before detailing the individual assumptions and design principles present in this paradigm, a set of related key concepts have to be defined and distinguished from each other, namely SOC, Service Oriented Architectures (SOAs) and Service Oriented Systems (SOS).

Each of these concepts builds on the basic idea of SO, however each represents a different perspective on this vision. In order to distinguish these paradigms, a well-known concept in IS research can be employed, the distinction between the *task layer* and the *task operator layer* (Ferstl and Sinz 2008, pp. 2-5). The task layer comprises all abstract tasks and their combination to processes, whereas the task operator layer contains all human or automated operators present in a given enterprise system, which on their part can execute tasks assigned to them.

Applied to the SO realm, a SOA defines all services, and therefore abstract functionalities, existing in a given system, thus representing the task layer. Just as with traditional enterprise IS the operators (task operator layer) providing the individual services can be both humans[1] or computer programs (also referred to as electronic services or service instances[2]). The design paradigm concerned with the definition and implementation of such electronic services is called SOC.

[1] For the remainder of this thesis human task operators will be omitted; the primary focus of my work lies on electronic services, the automated task operators.

[2] Although technically a distinction between service (task layer) and service implementation (task operator layer) would also be appropriate here, in the majority of the scientific literature only the term

All of the abovementioned concepts apply to either the abstract vision of service-based systems or the way the implied tasks are structured. An actual service-based system is called SOS, representing the set of electronic services offering the tasks defined in a SOA and consequently its automated task operators.

According to the general agreement in the literature (see for example Papazoglou and Georgakopoulos 2003; Srinivasan and Teadwell 2005) an electronic service (instance), can thus be defined as follows:

- An electronic service is an individually addressable software component that provides some functionality to a service requester.
- Services can be accessed over an electronic network, such as an enterprise intranet.
- Individual services can be composed to higher-level, more complex services, resulting in possibly multiple levels of service complexity.
- Services only advertise details, such as their capabilities, interfaces or accepted protocols that are needed to interact with them. Technical implementation details of the service are hidden from the service requestors.
- Regarding their interaction, electronic services are loosely coupled. This means that their interactions are not hardcoded in each individual service, but every service requester discovers and binds a given other service, it interacts with, at run time.

Among others, the main advantage of the resulting SOSs is a much higher flexibility, when compared to traditional systems. Due to the loose coupling service requesters do not rely on hard-coded links, but on dynamic service discovery and invocations. Additionally, SOSs allow for the dynamic instantiation and removal of service instances to cope with load fluctuations. The new service instances just have to be registered to the discovery system and can immediately be invoked (allowing for a better distribution of incoming service requests to available instances).

Due to the prominent advantages of this concept the SO vision has broadly been adopted within the last years for whole business IS as well as within individual software systems. Significant research work has been dedicated to the definition of technical service standards (such as Web Services (WS) (Booth et al. 2004), the Universal Description, Discovery and Integration (UDDI) (Bellwood et al. 2004) and the Simple Object Access Protocol (SOAP) (Gudgin et al. 2007)) supporting the definition, description and discovery as well as the interaction with electronic services.

Grid Computing

In parallel to these developments, regularly applied within a given enterprise, a paradigm for distributed IS has emerged, which is mostly dealing with the coordination of electronic resources across organizational boundaries: GC. It has been introduced in the

service is used, referring to both concepts depending on the context. I will also proceed this way, as especially with my focus on electronic task operators this distinction is of minor importance.

early years of the new millennium as a new paradigm for distributed execution of resource-demanding computing tasks (Foster 2002; Foster and Kesselman 2004; Foster, Kesselman, and Tuecke 2001).

According to two of the most recognized Grid researchers, Ian Foster and Carl Kesselman, GC is mainly concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (Foster, Kesselman, and Tuecke 2001, p. 201), emphasizing the cross-organizational nature of Grid systems.

Grids can thus be defined as:

- Systems that "coordinate[...] resources that are not subject to centralized control", also addressing problems such as security, policy or payment occurring when resources are shared across organizational domains (Foster 2002, p. 2). Those resources range from computational, storage and network resources to code repositories (Foster, Kesselman, and Tuecke 2001).

- In doing so, Grids employ "standard, open, general-purpose protocols and interfaces" (Foster 2002, p. 2), supporting the sharing process.

- The final overall goal of GC is "to deliver nontrivial qualities of service", following the vision of the individual "utility of the combined system [being] significantly greater than that of the sum of its parts" (Foster 2002, p. 3). This last statement explicitly confirms GC's focus on the user, aiming at maximizing the performance of a user's application, run on the distributed Grid nodes (Schopf and Nitzberg 2002).

With the definition of the Open Grid Services Architecture (OGSA) (Foster et al. 2002a) the GC vision has been integrated with emerging SO principles. The traditional goal of GC, to execute individual jobs on a set of distributed resources, remained the same, however a much more precisely described architecture for those resources was given. According to the vision of SO, each Grid resource was considered to be a service. Service semantics and management interfaces for the basic building blocks in such systems, *Grid Services*, were given, thus converging the metaphors used in business transactions (i.e. electronic business relationships with external or internal SPs) and in computing technology. The central concept of a distinct service has become the glue between those worlds.

This concept did not only provide a common basis for computer scientists and potential business users but also enabled the emergence of a powerful abstraction concept in distributed computing: *Virtualization*. "Virtualization enables consistent resource access across multiple heterogeneous platforms [and] also [the] mapping of multiple logical resource instances onto the same physical resource..." (Foster et al. 2002b, p. 40). From a user perspective virtualization thus aims at completely hiding not implementation but also deployment details of a given service. Service requestors only deal with the offered interfaces and do not need any information of the actual implementation or computer hardware the service instance is deployed on.

Cloud Computing

CC as the most current development builds on GC, SO and Virtualization technologies (Foster et al. 2002b) in order to implement "parallel and distributed system[s] consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on SLAs, established through negotiation between the service provider and consumers" (Buyya et al. 2009, p. 601).

CC can thus be characterized as follows:

1. Clouds heavily build on virtualization technologies. All offered computing resources are virtualized, thus hiding the implementation details from the end user.
2. Cloud services are offered by independent and external SPs.
3. Cloud resources are dynamically provisioned on demand. Computing power is supplied that can dynamically scale up or down as the demand for the hosted services varies.
4. Each CC offering is based on possibly pre-negotiated SLAs. Therefore the main goal of each Cloud service is to meet certain QoS levels.
5. CC resources are accessed over standard Internet protocols.

The main difference between CC and GC is probably that a cloud tries to present a centralized "image" that manages and schedules its resources in the background (e.g. at commercial data centers as with Amazon[3]) as opposed to Grids which explicitly offer access to decentralized resources with local policies (Begin 2008).

Clouds introduce a new abstraction layer between the raw resources and the users: a *virtualization layer*. All resources available at a given data center are pooled as an input for the virtualization layer, which in turn offers the available resources as discrete computing blocks to the users in the form of virtual servers. By introducing this virtualization layer Clouds manage to break down the potential M:N relationships between users and resources to two sets of 1:N relationships (users to virtualization layer and virtualization layer to resources respectively), therefore reducing the complexity of the overall system. In contrast, Grid systems combine resources, located in different organizational domains, to execute resource demanding jobs in parallel. Subsequently the individual partial results are collected from the individual Grid nodes and combined to a single result of the overall job.

Common Vision: Internet of Services

In the last subsections the state of the art in distributed computing was sketched in terms of three commonly applied paradigms for system design and operation. The presented developments expose a high degree of similarity in terms of the mechanisms and technologies used.

[3] http://aws.amazon.com/

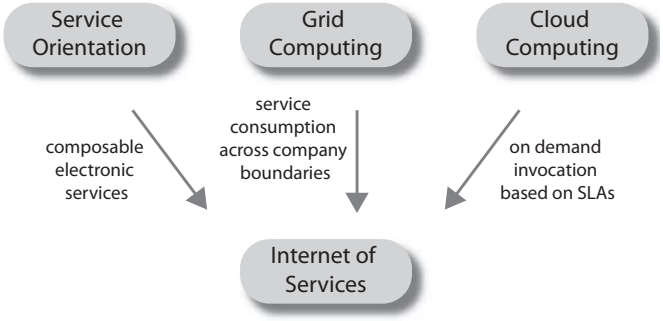


Figure 1.1.: Relation of IoS, SOC, GC and CC

Each of the depicted scenarios builds on very similar technical infrastructures, based on the Internet as a communication infrastructure. Also, the applied computational abstraction for all of those systems is always a service.

On the other hand, they differ slightly in the way the individual services are managed and used on a higher abstraction level. This is especially noticeable when looking at the applied invocation paradigms, the point of control within each invocation, the overall system configuration as well as the scope of the employed systems. However, not having different infrastructures but only using them differently should not prevent a development of consolidating and integrating computing paradigms in order to implement more powerful and efficient global systems.

Thus, many experts in IS research and industry share a common vision for the next generation Internet, based on highly dynamic networks of composable services, offered and consumed on a global scope, ultimately leading to innovative business models and supporting the transition from value chains to value nets (Blau et al. 2009; Scheithauer and Winkler 2008).

This vision builds on the aforementioned paradigms of SO, GC and CC by still following the service paradigm, the orchestration of internal and the choreography of external services as well as their on demand consumption. In addition to combining all these concepts, the IoS puts a much stronger focus on new business models and the commercial application of the SO ideas.

Summarizing the IoS scenario model results in the following set of characteristics:

- The IoS is composed of a set of electronic services.
- Services vary in complexity and therefore range from raw (hardware) resources to very complex workflows.
- They can be stand-alone (they only fail if an internal error occurs) or composite (meaning that a service depends on other service(s) which can potentially fail, ultimately also causing the composite service to fail).
- Each of the services is deployed at an abstract infrastructure node, representing a server or on a higher level a data center or organization.

- These nodes are interconnected via the Internet.
- Services and nodes can dynamically appear and disappear again, due to the IoS being an open system.
- Each of the services (more precisely, their respective management components) can adopt different roles, ranging from SPs and SCs to mediating roles such as a service broker.

For the remainder of this thesis this abstract scenario model is assumed as the problem context for my dissertation project.

1.1.2. Research Problem

The IoS scenario, as described in the last subsection, is more and more becoming reality. However, especially when employing this emerging global infrastructure for business workflows, still several serious issues remain unsolved until today.

Especially the need for guaranteed reliability and service quality becomes more prominent, as no longer the question of "who provides the service?" matters but only whether she is able to achieve the requested result. The assurance of such QoS even becomes crucial when external services are to be integrated into business critical workflows.

However, in globally distributed service systems no central QoS control can be easily implemented. Such systems inherently lack any type of control hierarchy, thus QoS management must be implemented in a decentralized way. Researchers agree that the most promising mechanisms currently available for this task are bi- or even multilateral SLAs in which the involved transaction partners assure each other certain QoS guarantees (provider side) or financial settlement (consumer side) (Keller et al. 2002a; Seidel et al. 2007). Such service contracts ensure service quality across individual sites and therefore across organizational boundaries in a decentralized way. Representing qualitative guarantees placed on services, SCs can benefit from SLAs because they make non-functional properties of services predictable. On the other hand, SLAs enable SPs to manage their capacity, knowing the expected quality levels.

A crucial phase throughout the SLA-based service management life cycle is the negotiation of the respective SLA document. A negotiation basically "constitute[s] the process of two or more parties communicating in order to proceed from some conflict situation to an agreement" (Hudert 2006, p. 13). Bichler et al. define negotiations as "the decision-making approach used to reach consensus whenever a person, organization or another entity cannot achieve its goals unilaterally" (Bichler, Kersten, and Strecker 2003, p. 312).

Based on these definitions a negotiation represents a configuration of two or more parties (SP and SC in the IoS scenario), arguing about some abstract good (a SLA, governing a respective service invocation). In such situations the interests of the involved parties collide and lead to a conflict situation concerning the subject of the negotiation (SPs will probably want to deliver a low quality for much money whereas SCs probably have quite opposing intentions).

For a long time, economic research came up with a rich set of different negotiation protocols (Ströbel and Weinhardt 2003), such as single or double sided auctions (see (Wurman, Wellman, and Walsh 2001) or (Klemperer 1999) for detailed overviews) or one-on-one bargaining protocols (see example Smith 1980), each tailored to a different negotiation setting and thus exhibiting very specific characteristics. Depending on the characteristics of the traded services, the market configuration or the context of the negotiators, a different negotiation protocol has to be used to reach the highest-overall efficiency of the service market (Buyya et al. 2009; Neumann et al. 2008; Paurobally, Tamma, and Wooldridge 2007). Common knowledge between researchers thus states that "there is no single best [solution] for all imaginable sourcing activities" (Block and Neumann 2008, p. 44) or contrarily that negotiation systems only supporting one particular protocol lack flexibility as needed in various negotiation scenarios (Benyoucef and Rinderle 2006; Kersten, Law, and Strecker 2004; Paprzycki et al. 2004; Strecker et al. 2006). Others more implicitly argue in the same direction by implementing systems capable of providing more than one protocol to the user (Wurman, Wellman, and Walsh 1998) or by defining taxonomies and ontologies used to structure currently used and newly designed protocols (Ermolayev and Keberle 2006; Lochner and Wellman 2004; Rolli et al. 2006; Tamma, Wooldridge, and Dickinson 2002; Wurman, Wellman, and Walsh 2002).

Research areas like Mechanism Design (Bichler 2001), dealing with the conceptual design of negotiation mechanisms, base their work on the very fact that there is no single perfect mechanism. Mechanism designers do so by employing both intuition and experience with market mechanisms or more formal tools like mathematical optimization, in order to design negotiation protocols satisfying a set of a priori determined requirements.

Among others, the following attributes determine the suitability of a negotiation protocol for a given market setting (Bichler, Kersten, and Strecker 2003):

- Characteristics of the negotiation object. This also includes time constraints as for example present with perishable goods. Furthermore valuation (common value vs. private value) and the uniqueness of this product is considered (commodities vs. individually tailored goods).
- Market configuration. This incorporates supply as well as demand situation as perceived by a market participant.
- Context of the negotiators. This attribute comprises for example the negotiators' sourcing objectives or potential timing constraints posed on them for making the transaction.
- Risks associated with a protocol for a given set of negotiators. A protocol's associated risk has to be compatible with the risk attitude of the involved agents and the nature of the negotiation object.

Given the global context of the envisioned scenario it is thus not likely, or even efficient, that only one central marketplace for electronic services will emerge, offering a single,

known protocol. Instead, a system of marketplaces offering different protocols is foreseen on the basis of economic theory. Each marketplace will offer an individual negotiation protocol, depending on the characteristics of its actual context. On the other hand it becomes necessary for SCs to take part in several marketplaces, even at the same time in order to fulfill ones individual service needs.

However, current systems restrict SCs in that they are only able to interact with one distinct service market they were implemented for (and are therefore only technically compatible with the applied negotiation protocol). This unnecessarily decreases the potential flexibility and efficiency of the IoS as a whole. SCs should be able to buy, and therefore negotiate about, any fitting service, regardless of the market it is offered in, and thus regardless of the protocol with which it is offered.

Finally, the usage of automated negotiation and discovery systems becomes crucial given the dynamic nature of distributed workflow execution and the increased complexity of global service selection as present in the IoS. Manual negotiations of human users would by far not be efficient enough to cope with these circumstances.

Research Question Subsequently, the research question to be answered during this dissertation project is:

How, can automated SLA discovery and negotiation mechanisms for highly distributed IoS settings be designed to support protocol-generic negotiation processes?[4]

1.2. Structure of this Thesis

On a first layer this thesis is structured into five high-level chapters, each of which is dealing with one particular aspect of my work and its scientific context.

In chapter 1 I motivate my dissertation project by producing answers for the following questions:

- What is the problem context my research is grounded in? And consequently, which environments will profit from the achieved results?
- What scientific problem arises in such settings and how can the overall research question, to be answered within this thesis, be defined accordingly?
- Which scientific method has been applied to answer the stated question?

In doing so, I have first presented a detailed scenario analysis in subsection 1.1.1, building on current developments in distributed computing and ultimately defining the context for my research. In a second step, the actual research problem has been deducted from this scenario model (section 1.1.2), followed by a precise definition of the research

[4]The discovery of electronic services and respective SLA offers is of secondary interest in this work. Hence, it will only be covered within my thesis to the extent needed for the protocol-generic negotiation phase.

question underlying my work. Section 1.3 closes this first chapter by presenting the research method applied.

The second chapter provides an overview on the scientific context of the identified research problem. The questions answered in this chapter are:

- What requirements can be identified for a potential solution of the stated research problem in the given scenario?
- Which theoretical concepts and research work provide valuable input for my work?
- Which alternative solutions to at least parts of the stated problem exist and why are they not sufficient?

Building on the scenario analysis done in subsection 1.1.1, a requirements analysis for my work is provided in section 2.1, delimiting the range of potential solutions. Theoretical foundations for my thesis are then presented in section 2.2. I close this chapter with a list of related research efforts, targeting (parts of) the identified problem in 2.3. They are structured according to the requirements derived before, thus explicitly stating which of the requirements can already be satisfied with current technology and which cannot.

The central deliverable of my work, the design of the BabelNEG system, is presented in chapter 3. In doing so, it will answer the following questions:

- What is the underlying solution idea I have applied to the stated problem?
- Which computational entities have been designed to realize this idea?
- How have these components been designed and why?

Section 3.1 presents this design idea and sections 3.2 - 3.4 provide information on the developed concepts, distinguishing between data structures, interaction patterns and functional components.

Chapter 4 subsequently addresses the evaluation of my proposed design with respect to the identified requirements. Here, also the steps to be undertaken for actually deploying, parameterizing and starting the system are described (this also includes example instantiations of the developed data structures).

- How can the proposed design be implemented in a prototypical system?
- Is such an infrastructure prototype able to fulfill all stated requirements (effectiveness)?
- How good is it able to do so (efficiency)?

To this end the prototypical implementation of my system design is presented in section 4.1 before the conceptual and simulative assessment of this prototype is shown

in sections 4.2 and 4.2.2. Chapter 4 closes with a final evaluation step, investigating the efficiency of the proposed solution (sections 4.4 and 4.5).

The fifth chapter of this thesis is finally concerned with both, a retrospective reflection on the developed system design and applied research method (section 5.2) as well as with an outlook on related further research questions that have been identified in the course of my dissertation (section 5.4). It will thus answer the questions:

- What was the essential outcome of this dissertation project?
- How can the appropriateness of the applied research method be assessed?
- How could the developed mechanism be efficiently implemented within a running commercial service infrastructure and how could it be extended in order to further increase its utility (future work)?

1.3. Research Method

In this section the research method applied within my thesis to answer this question is described. A short overview on research methods, as present in the current IS research landscape, is given first before the one chosen for this dissertation project is described in more detail.

1.3.1. General Considerations

On an abstract level, a research method describes the general course of action undertaken in any research effort. To this end, research methods regularly comprise a set of intersubjectively understandable, normative rules, which are to be used as goal-oriented directives for a research process at hand (Wilde and Hess 2007, p. 281). Since its advent as a research discipline a variety of different research methods were employed in the field of IS research, ranging from case studies, laboratory experiments and simulations to mathematical proofs (Glass, Ramesh, and Vessey 2004, p. 91) (Wilde and Hess 2007, p. 282).

When trying to further distinguish the research methods present, not only in IS research but in research in general, one can identify two main classes of research methods: such applying to *descriptive research* and such suitable for *prescriptive research*. The main goal of descriptive research is to explain certain phenomena that can be observed in nature, society etc. Research methods for this paradigm thus focus on the observation of some entity (or a system of entities) and the subsequent analysis of the resulting data. On the other hand, prescriptive research aims at solving complex problems identified by the researcher. Respective research methods therefore define steps to identify and solve such problems as well as to assess the achieved results in the light of the problem to be solved (Hevner et al. 2004). This second type of research is often called *Design Science* (DS), especially in IS and engineering disciplines.

While IS research has been dominated historically by descriptive research approaches, in recent years there is a growing trend to generate a more holistic view of research as

such. Initiated by Herbert Simon in his influential book "The Sciences of the Artificial" (Simon 1996), research efforts aiming to *improve* the environment, rather than to *explain* it became more and more prominent. Such (DS) research efforts produce artifacts that serve a distinct purpose and are subsequently evaluated regarding their utility, instead of with regard to their explanatory power, as with descriptive research. Such artifacts are defined as artificial, human-constructed entities as opposed to something that occurs naturally (Simon 1996).

March and Smith argued on the structural equivalence of both research paradigms on an abstract level: Descriptive research (as mainly present in the natural sciences) consists of two activities. A theory has to be developed (*theorize*), based on empirical observation results, and *justified*, building on a potentially huge set of empirical evidence and inductive steps (March and Smith 1995, p. 255). DS on the other hand, is based on two activities as well: A researcher has to *build* an artifact that improves the environment. Subsequently, in order to prove that the research has been effective, the artifact needs to be *evaluated* (March and Smith 1995, p. 255).

Figure 1.2 follows this rationale and tries to integrate the two research paradigms on an abstract level. In doing so it also supports Hevner's view, propagating both approaches to complement each other on a high-level perspective (Hevner et al. 2004, p. 98). The top half of the research cycle displays descriptive research, the bottom half shows DS with the corresponding steps[5].

The cycle has two potential starting points, depending on the research question to be addressed. If a researcher seeks to explain an observed phenomenon (descriptive research), the entry point is on the left hand side. She starts with an existing technology and discovers a certain new phenomenon that was not yet investigated. In the theorize phase, she develops a hypothetical theory that could explain the phenomenon, potentially based on some empirical evidence such as observation data. To obtain a useful theory, it needs to be justified. All explanations and predictions of the theory must be consistent with the empirical findings. In line with Popper's philosophy of science, even in that case the developed theory or hypothesis still remains "tentative" (Popper 2002, p. 280) because it can still be refuted by empirical findings in the future. This leads to the well-known process of theory justification or rejection within the descriptive sciences.

The second potential entry point is on the right hand side. If the research question aims to solve a problem (Hevner et al. 2004, p. 78), the research is design oriented or prescriptive. Building on the theoretical foundations that have been generated within descriptive research, a researcher implements an artifact. After the implementation, the artifact must be evaluated. Only if it provides greater utility than other, existing artifacts, it can be considered useful, and the research problem can thus be considered to be brought one step closer to an ultimate solution. To complete the cycle, an evaluated artifact can be the source of new descriptive research, as it denotes newly available technology, whose effects on existing systems can be investigated as part of a descriptive research effort.

[5]For the remainder of this section I will focus on descriptive and prescriptive research in the area of IS, although the assertions made are also applicable for other research areas.

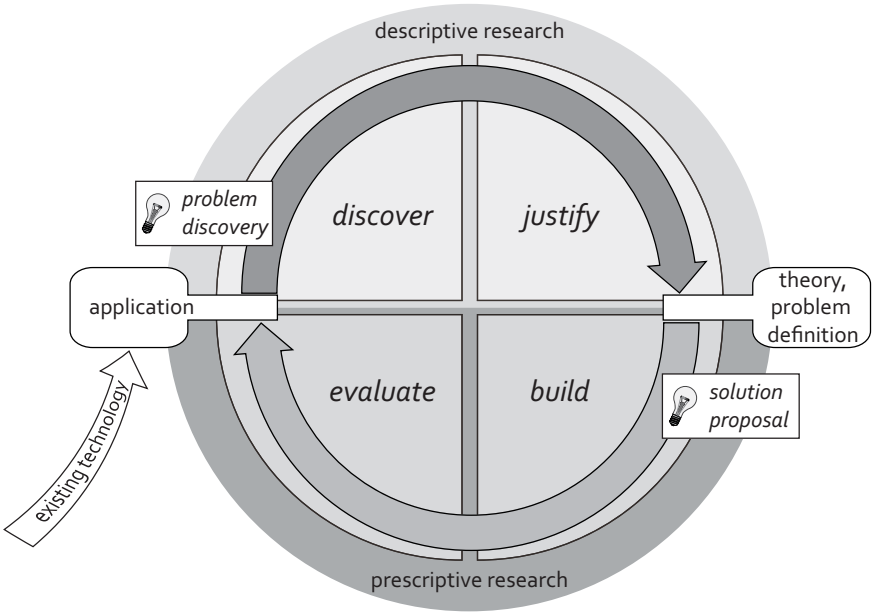


Figure 1.2.: Consolidated Research Process

The consolidated research process thus integrates descriptive as well as prescriptive research. The choice of methods is based on the research problem to be addressed: If its aim is to explain something, it starts on the left hand side of the cycle and uses the two activities of descriptive research. Research that tries to improve the environment with a novel artifact uses the DS part of the cycle. It starts on the right hand side and uses the two activities of prescriptive research.

Due to the nature of my research goal, a DS research process has been chosen accordingly. In the next subsection this lower side of the research cycle, as presented above, as well as its actual implementation within my work is described in more detail.

1.3.2. Design Science in Information Systems Research

As just described, DS is "fundamentally a problem solving [research] paradigm", originating in the engineering sciences and the science of the artificial (Hevner et al. 2004, p. 76). In doing so it "seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" (Hevner et al. 2004, p. 75).

When trying to further detail this paradigm it can be useful to distinguish two perspectives on it: the *dynamic* perspective, defining the process of designing an artifact, as opposed to the *static* perspective, describing the structure of the design knowledge (result of the DS process). This follows the rationale of Walls et al. who regard design as both a *process* (set of activities) and a *product* (designed artifact) (Walls, Widmeyer, and El Sawy 1992, p. 42).

Design Science Research Process

Up until now a significant amount of different DS research process models have been developed in science and industry; Peffers et al. (2008) give a good overview on the most prominent ones. The same authors also attempted to create a consolidated process model that is consistent with the ones found in the literature. It provides a nominal process description for DS research (as guideline for a) researchers conducting DS work as well as b) reviewers when assessing it) (Peffers et al. 2008). This model has been adopted as a basis for this thesis, as it provides the most structured and comprehensive process description for DS currently available.

It comprises six distinct activities:

1. *Problem Identification and Motivation*
2. *Objectives of a Solution*
3. *Design and Development*
4. *Demonstration*
5. *Evaluation*
6. *Communication*

Problem Identification and Motivation This step aims at "defin[ing] the specific research problem and justify[ing] the value of its solution" (Peffers et al. 2008, p. 52). This phase marks one of the most crucial steps during any DS research effort, as it identifies the goals and application context for the artifacts to be developed subsequently. In the context of IS research, a problem is considered relevant whenever it can be described as an "unsolved and important business problem..." (Hevner et al. 2004, p. 84) occurring in the context of "the interaction of people, organizations and information technology" (Hevner et al. 2004, p. 85).

DS researchers always have to make sure that their research attempt can be distinguished from routine design, and is thus worthy of a research effort. Hevner et al. present a set of characteristics a problem should exhibit for its solving being a legitimate DS process (Hevner et al. 2004, p. 81):

- unstable requirements and constraints based upon ill-defined environmental contexts.
- complex interactions among subcomponents of the problem and its solution.
- inherent flexibility to change design processes as well as design artifacts.
- a critical dependence upon human cognitive abilities (e.g. creativity) to produce effective solutions.

- a critical dependence upon human social abilities (e.g. teamwork) to produce effective solutions.

Hence, given at least one of the abovementioned requirements is fulfilled for the problem addressed in this thesis, the DS method can be considered appropriate for my dissertation project.

Objectives of a Solution In the next phase, the problem definition developed in the first step is used to derive precise requirements for the artifacts to be developed. These can either be *quantitative* or *qualitative*, the former defining measurable metrics "in which a desirable solution would be better than current ones", the latter defining how the newly designed artifact will "support solutions to problems not hitherto addressed" (Peffers et al. 2008, p. 55). Especially in case of quantitative objectives being identified, this step requires the researcher to have extensive knowledge on currently available solutions to the identified problem and their efficiency.

Design and Development This step marks the core process of all DS projects, as it comprises the actual creation of the artifact, providing the researchers with a solution to the identified research problem. This includes the determination of the "artifact's desired functionality and its architecture" (Peffers et al. 2008, p. 55).

A lot of very detailed process models for this phase have been developed, ranging from the traditional Waterfall Model for Software Engineering (Royce 1987) to the iterative Spiral Model (Boehm 1986).

Whichever approach is used, the researcher is bound to build on the common knowledge currently available in the respective research discipline (Hevner et al. 2004, p. 80), also called *Justificatory Knowledge* in (Gregor and Jones 2007, p. 322). On the other hand the researcher is urged to apply *rigorous* research methods to her work, which can be assessed based on the "applicability and generalizability of the [developed] artifact" (Hevner et al. 2004, p. 88).

Hevner et al. define such an artifact to be "created to address an important organizational problem [and it being] described effectively, enabling its implementation and application in an appropriate domain" (Hevner et al. 2004, p. 82). They also identified four different types of artifacts that can be the result of a DS activity (Hevner et al. 2004, p. 77):

- *Construct* (vocabulary and symbols)
- *Model* (abstraction and representation)
- *Method* (algorithms and practices)
- *Instantiation* (implemented and prototype systems)

Gregor and Jones categorize them into two distinct groups, namely "theories [or] immaterial artifacts" (constructs, models or methods)" and "instantiations [or] material artifacts" (instantiation) (Gregor and Jones 2007, p. 321).

Demonstration During this step the artifact's capability to solve the addressed research problem is to be shown. Most DS research models integrate this phase with the next one, as both aim at assessing the developed artifact. The difference between both is basically whether the capability of the developed artifact to solve the problem at all is assessed (demonstration) or how good it does so (evaluation). This distinction allows for a much more structured assessment, especially in cases where artifacts are designed addressing problems that were not addressed before at all. Here the first question to ask is whether the artifact fulfills its requirements and questions about the quality of the solution (evaluation) are secondary. In these cases the "research contribution lies in the novelty of the artifact and in the persuasiveness of the claims that it is effective" (March and Smith 1995, p. 260).

Evaluation After having shown that the artifact essentially fulfills its purpose, the evaluation step addresses the quality of the designed solution. According to Peffers et al. this involves the "compari[son of] the objectives of a solution to actual observed results from use of the artifact in the demonstration" (Peffers et al. 2008, p. 56). By building on a set of evaluation criteria (which should be derived from the objectives of the solution) as well as objective evaluation methods this steps basically measures how good the artifact solves the stated problem.

Following Hevner et al., all evaluation criteria for IS artifacts originate in the business environment they are supposed to be applied in; common examples for such criteria are for example completeness, functionality, consistency or performance (Hevner et al. 2004, p. 85). It is up to the actual researchers to identify the criteria appropriate for the respective work and specify the way those are calculated based on the observation made from using the artifact.

Past developments have shown that even artifacts that were worse than their competitors in the traditional metrics not only kept being used by a significant set of customers, but often even outlived their "better" predecessors. Such technology is often called *disruptive* as opposed to the *sustaining* technology it sets out to outclass (Christensen 1997). These new technologies obviously exhibited some characteristics that were not part of traditional evaluation in the respective context, but that proved to be very much important over time (a good example for this are the 3,5" and 2,5" hard discs). This phenomenon can easily be observed in DS research as well and has to be taken into account when defining the metrics for a given evaluation setting: even if a new artifact renders to be worse than present ones on some scale it might be much better in a new, not yet considered metric, which could become very important in the future. In such DS efforts it is crucial to explicitly define the characteristic(s) in which the designed artifact is varying from its competitors and why this metric can be foreseen to be of future importance. An actual assessment of this assertion can only be made over a significant period of time during which this new metric has to prove its importance (long-term evaluation of disruptive research artifacts).

Based on the structure of the artifact and the identified evaluation criteria an appropriate evaluation technique must be selected as a second step (see (Hevner et al. 2004)

or (Bucher, Riege, and Saat 2008) for examples). The most prominent ones, in the context of DS research, are probably building and applying of software prototypes in a productive environment, computer simulations, surveys or lab experiments (Bucher, Riege, and Saat 2008, p. 81) as well as analytical (if possible) or descriptive evaluations (Hevner et al. 2004, p. 86).

Communication The final step within every DS effort should be the communication of the results to fellow "researchers and other relevant audiences, such as practicing professionals" (Peffers et al. 2008, p. 56). Hevner et al. define the two classes of potential audiences as "technology-oriented as well as management audiences" (Hevner et al. 2004, p. 90), given the application domain of IS research.

The challenge occurring in this final step is a) to identify respective outlets that promise a good visibility among the addressed audiences (e.g. conferences or journals) as well as to b) identify all results interesting for these groups. This includes not only the artifact itself, but also "the [addressed] problem and its importance, [..] its utility and novelty, the rigor of its design, and its effectiveness" (Peffers et al. 2008, p. 56).

Results of Design Science Efforts: Design Theories

A question, occurring during the communication step of DS research at the latest, is how "design knowledge[, being the result of any DS effort,] can be captured, written down and communicated" (Gregor and Jones 2007, p. 313). An abstract guideline for DS researchers is needed for structuring their achieved results in a way they can communicate them. Gregor and Jones proposed such a guideline by consolidating former works on DS theories from Walls et al. (Walls, Widmeyer, and El Sawy 1992) and Dubin (Dubin 1978). The main idea is not to view the designed artifact as the primary result of a DS process, but rather that it is no more than the nucleus around which a comprehensive design theory is to be defined. The scientific findings of a DS process are expected to be generalizable (much more than the sole artifact would be) theories which should be able to act as part of the theoretical basis for future research efforts.

The proposed structure of a DS theory identifies eight different categories or abstract components that should be the result of an ideal DS process (Gregor and Jones 2007, p. 322):

- *Purpose and Scope*
- *Constructs*
- *Principles of Form and Function*
- *Artifact Mutability*
- *Testable Propositions*
- *Justificatory Knowledge*

- *Principles of Implementation*
- *Expository Instantiation*

Purpose and Scope This aspect defines the high-level goals of the developed design theory (in which the developed artifact is grounded / which was derived from the developed artifact). In doing so, the scope or boundaries of the theory, and systems for which it applies respectively, are shown. The requirements are to be stated on a "meta-level" in that the goal of a DS researcher should be to define a design "theory that is suited to a whole c1ass of artifacts that are typified by these requirements" (Gregor and Jones 2007, p. 325).

Constructs Constructs represent the basic vocabulary of any theory, consisting of its ("indicative, rather than [complete]") set of "entities of interest" (Gregor and Jones 2007, p. 325). It can thus be seen as an extended glossary of "physical phenomena or abstract theoretical terms" (Gregor and Jones 2007, p. 325) used in a given theory, such as "software agent" or "SLA".

Principles of Form and Function This component describes the "structure, [...] shape [...], properties and functions" (Gregor and Jones 2007, p. 326) of an artifact. It thus defines the "blueprint" for individual instantiations following the respective design theory.

Artifact Mutability Given the inherently changing context of any IS artifact this aspect of a design theory covers the capabilities of a developed artifact when evolving over time and adapting to new application settings and organizational environments.

Testable Propositions This component describes a set of "testable propositions or hypotheses" (Gregor and Jones 2007, p. 327) on the behavior of the developed artifact. By this means it provides the main input for assessment of the artifact, taking place in the demonstration and evaluation steps as presented above.

Justificatory Knowledge The theoretical basis on which the design theory is grounded is known as justificatory knowledge. It thus represents related artifacts and respective theories underlying the current DS effort. This concept is to be distinguished from other DS projects aiming at the same problem (regularly known as *related work* in scientific papers), as those are basically competitors to the currently designed approach whereas the justificatory knowledge only provides a common ground for all those efforts.

Principles of Implementation This aspect describes "the means by which the design is brought into being" (Gregor and Jones 2007, p. 328). An abstract guideline for implementing the design theory in an actual artifact is given, thus depicting how the results of the DS effort can be applied in real-world settings.

Expository Instantiation An actually instantiated artifact implementing the developed design theory "contributes to the identification of potential problems in a theorized design and in demonstrating that the design is worth considering" (Gregor and Jones 2007, p. 329). It therefore proofs that the theory can be instantiated and that such an artifact ultimately solves the addressed research problem (the demonstration and evaluation phases of the research process assess the developed design on the basis of such an instantiation)[6].

Any DS effort (such as the one presented in this thesis) should aim at describing the proposed problem solution from all these perspectives in order to allow for generalization and application of the results in real-world settings as well as in future scientific projects.

1.3.3. Implementation of the Design Science Paradigm

Design Science Process

This dissertation project follows the DS process model that was introduced in the last subsection except for one additional phase concerning the *Metrics Operationalization and Testbed Implementation*. Such efforts are also present in the model proposed by Peffers et al. (Peffers et al. 2008, p. 54), however, they are not explicitly presented as a distinct phase, which underrepresents their value to the overall process in my opinion.

The complete process as adopted for this thesis is shown in figure 1.3:

As a first step, the research problem has been identified (it has already been detailed in section 1.1): the lack of mechanisms for a comprehensive and fully automated discovery and flexible negotiation of SLAs in distributed settings such as the IoS. It has been deduced from a detailed scenario model for the IoS after investigating respective literature on service-based systems as well as the research problems stated within current national and international research projects.

The problem addressed in this thesis clearly fulfills at least some of the criteria Hevner et al. (Hevner et al. 2004) impose on a DS project. The complex nature of service-based settings and the intricate interactions and dependencies between the different components of the IoS can probably be seen as the most prominent one.

Next, the objectives for solution to the respective problem are stated in 2.1. They primarily originate in a thorough analysis of scientific literature about the stated scenario and research problem.

After the objectives for this DS project are defined, two basically independent phases can begin: the design and development of the actual artifact (i.e. the SLA management infrastructure) as well as the definition of the evaluation metrics and the testbed implementation. These two phases have no actual contact points, so they can be conducted independently. The only requirement regarding the overall process is that the testbed, within which the designed and implemented artifact is to be assessed, must be finished before starting the demonstration and evaluation phases. The testbed used within this

[6] Having produced an expository instantiation does not necessarily render a design effort to be valid DS, as in the case of just having such an artifact without an appropriate theory of design "the level of knowledge is that of a craft-based discipline" (Gregor and Jones 2007, p. 329).

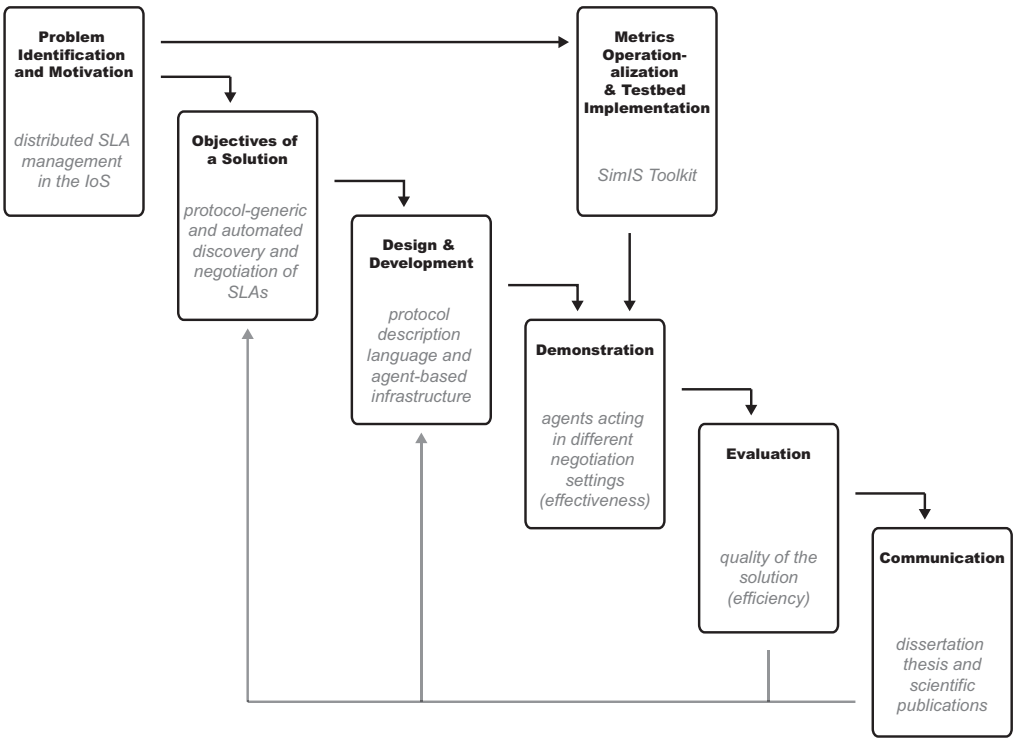


Figure 1.3.: Research Process applied for this Thesis

thesis is the IoS simulation toolkit SimIS ("Simulating an Internet of Services")[7] (König, Hudert, and Eymann 2010), which I co-developed. In parallel, the metrics to be used within the demonstration and evaluation phases, are derived on the basis of the research question and scenario definition at hand.

During the design and development of the actual artifact, the mechanisms and data structures needed for solving the stated research problem are defined and subsequently implemented in a proof-of-concept prototype.

Once both this expository instantiation as well as the testbed are finished, the demonstration and evaluation steps are undertaken. To this end, the developed software components are deployed in a simulated IoS scenario (as parameterized using SimIS) and subsequently assessed regarding the requirements defined before.

Following Bucher et al., simulation is a valid evaluation (and demonstration) technique within DS whenever the application of a prototype in a real-world experiment is impossible (see for example Bucher, Riege, and Saat 2008; Hudert, Niemann, and Eymann 2010). This can be due to *pragmatic* (e.g. investigations on fluid behavior in the core of

[7] http://sourceforge.net/projects/simis

the sun), *theoretic* (e.g. what-if questions on different values for natural constants) or *ethical* reasons (Hartmann 1996, p. 87). In the case of this project a pragmatic reason is present, as a global IoS as I envision it, is still only a vision for future systems and not already implemented reality in IS. Initial steps towards this vision can already be observed in current infrastructures (for example SAP's BusinessByDesign[8], Enomaly's SpotCloud[9] or SalesForce[10]), but the comprehensive IoS vision in its entirety is not present yet. Simulation therefore represents an appropriate evaluation tool for exploring these scenario settings, "that cannot (yet?) be investigated ... by experimental means" (Hartmann 1996, p. 87).

As a conceptually last step the results of this project are communicated in the form of scientific publications (see Hudert (2006, 2009, 2010); Hudert and Eymann (2010, 2011a,b); Hudert, Ludwig, and Wirtz (2006, 2007, 2008, 2009); Hudert et al. (2009)) in both, the computer science and business communities, contributions to conjoint research projects as well with the publication of this thesis as a whole.

Resulting Design Theory

In order to describe all relevant input and (intermediate) results of my research project, the remainder of this thesis is primarily structured according to the research process just sketched. Nevertheless, I also tried to incorporate Gregor and Jones' structure of a design theory when presenting my results. In the following I very shortly want to provide some overview information on where in this thesis which of the mentioned theory components can be found. These remarks can thus act as an alternative approach for an outline of this thesis and are supposed to guide the reader whenever she is particularly interested in a distinct aspect of the theory resulting from my work.

- The *Purpose and Scope* of my work can be found in section 1.1 and more particularly in 1.1.1, where my actual research goal is presented.
- Next, the *Constructs* used within this thesis are implicitly introduced in section 2.2. In this section also the *Justificatory Knowledge* is described as an input for my work and whenever a distinct construct, that is needed in the further thesis, is introduced, it is highlighted respectively.
- The main part of this thesis, the design of the SLA management infrastructure in chapter 3 basically comprises the *Principles of Form and Function*, whereas the *Testable Propositions* can be found in 4 as part of the demonstration and evaluation experiments.
- Also located in chapter 4 is the description of the implemented proof-of-concept prototype (*Expository Instantiation*).

[8] http://www.sap.com/germany/sme/solutions/businessmanagement/businessbydesign/index.epx
[9] http://www.spotcloud.com/
[10] http://www.salesforce.com

- The last two aspects of a design theory are concerned primarily with the time after the actual research process and are thus mentioned in chapter 5: A short guideline for how to use my system in a productive environment is given in 5.3 (*Principles of Implementation*) and finally the system's capability to cope with changing environments and potential extensions to the system to increase this capability are sketched in 5.4 (*Artifact Mutability*).

2. Objectives and Foundations

In this chapter the objectives of my work as well as the conceptual foundations upon which this thesis builds are presented.

2.1. Requirements Analysis

In investigating the identified research problem and the resulting research question, a set of requirements for the intended solution can be identified. They are structured according to the phases of the overall service life cycle (see figure 2.1) they refer to, i.e. the discovery or negotiation phase, as those are the main focus of this thesis. Within the negotiation phase requirements another level of categorization is introduced, following a well-known distinction of negotiation research: *negotiation object* (referring to the entity negotiated about), *decision making strategy* (referring to the way negotiators act during a negotiation) and *negotiation protocol* (defining the way negotiators can communicate during negotiation) (Jennings et al. 2001, pp. 200-201):

Discovery Phase: During the Discovery Phase, a system of market registries is needed to support the discovery and publication of service offerings. This ensures effective and efficient discovery processes in which potential transaction partners are made known to each other, a prerequisite for any economic transaction.

R1 Market registries should be present for the service discovery phase (Buyya et al. 2009).

Negotiation Phase - Negotiation Object: Next the requirements for the negotiation phase are presented, beginning with the ones concerning the negotiation object; in the IoS these objects are service invocations and therefore SLAs. Due to the variety of different services the complexity of such service invocations, respective SLAs of highly varying complexity must be negotiable.

R2 Service management in the IoS should be based on (electronic) SLAs of varying complexity (Barros, Dumas, and Bruza 2005; Blau, Stösser, and Block 2008; Buyya et al. 2009; Ludwig et al. 2006; Neumann, Stösser, and Weinhardt 2007; Neumann et al. 2008; Paurobally, Tamma, and Wooldridge 2007).

On the other hand, in order to delimit the potential agreement space a priori (for example as needed in fixed price protocols) a possibility to define some SLA terms to be non-negotiable is needed. These correspond to hard-coded service characteristics,

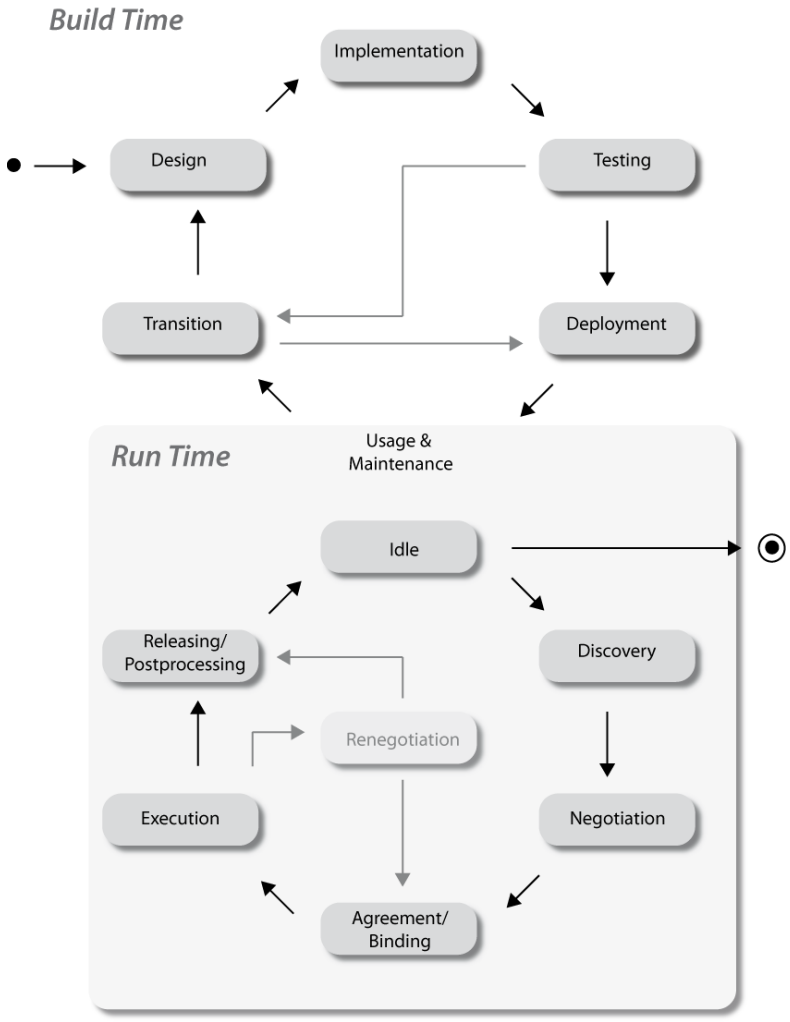


Figure 2.1.: Integrated Service Life Cycle Model for the IoS

that the offering management agent cannot (or is not willing to) change and therefore negotiate about.

R3 Possible offers should be restrictable, including non-negotiable SLA terms (Brandic et al. 2008b; Ludwig et al. 2006; Ziegler et al. 2008).

Negotiation Phase - Negotiation Protocol / Setting: In trying to cope with the multitude of possible settings in the IoS it is inappropriate to only offer one particular negotiation protocol (Neumann, Stösser, and Weinhardt 2007). Following the main motivation for this thesis, such mechanisms would not provide the needed flexibility in order to achieve efficient or even stable system states at some times. Analogous to real-world economies digital service economies offering different negotiation protocols within indi-

vidual service (micro-) markets are needed for efficient service management on a global scope. Additionally, the knowledge of the protocols to be used has to be created or discovered by the involved services during the discovery phase dynamically (this means during run time).

With regard to the involved negotiator agents, this implies that they must be able to act on different markets, thus within different protocols. Such settings are of special interest in multi-tier market infrastructures as presented within a variety of different research efforts (see for example Streitberger et al. 2008). These approaches try to cope with the huge differences in complexity between the electronic services present in the overall economy by introducing individual markets at each abstraction layer. Agents then for example buy or sell raw resources on one market whereas they buy or sell more complex services on the other one. In addition to this vertical view, agents can be in the need of acting on horizontally distributed markets, for example when different services offered on different micro-markets have to be combined to a complex workflow. Such situations will be very common in global and protocol-generic infrastructures as envisioned in this thesis.

R4 SLA negotiations should allow to change the applied negotiation protocol (protocol-generity) (Barros, Dumas, and Bruza 2005; Brandic et al. 2008b; Buyya et al. 2009; Ludwig et al. 2006; Neumann, Stösser, and Weinhardt 2007; Neumann et al. 2008; Paurobally, Tamma, and Wooldridge 2007; Resinas, Fernandez, and Corchuelo 2010).

Taking this argument one step further, a restriction of the possible negotiation protocols to a pre-defined set still does not provide the flexibility needed for the IoS scenario. Given the massively distributed and dynamic setting, this assumption is easily justifiable as very diverse negotiation situations (and thus different protocols) can be anticipated to appear therein. Restricting the supported protocols to a pre-defined sub-set (known to all involved negotiator agents at start-up) necessarily also restricts the benefits such an infrastructure could achieve (Hudert, Ludwig, and Wirtz 2009).

R5 In order to cope with the highly dynamic IoS environment the available set of protocols should not be restricted a priori.

Negotiation Phase - Negotiation Strategy / Participants: A vast majority of researchers agree that automated software agents should be employed for service management in the IoS. Software agents can bring both, the flexibility and appropriate speed as well as the required strategic intelligence, needed in such complex global systems. Additionally, they represent the right abstraction level for user input as each agent can be assigned to a human user and thus users can implement their desired negotiation behavior within their agents, which will act on their behalf further on.

R6 Software agents should be applied for service management in the IoS (Barros, Dumas, and Bruza 2005; Blau, Stösser, and Block 2008; Borissov, Neumann, and

Weinhardt 2009; Bui, Gachet, and Sebastian 2006; Hung, Li, and Jeng 2004; Ludwig et al. 2006; Neumann, Stösser, and Weinhardt 2007; Paurobally, Tamma, and Wooldridge 2007).

Other than the two main roles in each negotiation, SC and SP, additional software agents are needed to act as third party roles for supporting the actual negotiation processes. By employing such mediators, the overall complexity of the IoS can be reduced and tasks ensuring the overall stability of the system can be safely implemented, such as market makers matching bids in a two-sided auction process or negotiation brokers assisting the agents in their decisions by providing reputation information, for example.

R7 Intermediaries should be present to act as auctioneers and market makers for negotiation support (Barros, Dumas, and Bruza 2005; Buyya et al. 2009).

Aside from these conceptual requirements a SLA management infrastructure, as developed within this dissertation, should follow some fundamental technical claims, such as being service-oriented and decentralized (Foster et al. 2002b; Hung, Li, and Jeng 2004; Paurobally, Tamma, and Wooldridge 2007). This is a sine qua non for any mechanism to be applied in the IoS, itself being service-oriented and decentralized.

Also, human users defining and executing business processes should not be bothered with technical aspects of service discovery and negotiation. They should be able to pass their preferences over to the service management agents and be presented with the (hopefully positive) results of their invocations. The infrastructure should thus be transparent to the user (Bui, Gachet, and Sebastian 2006).

Finally, in order to achieve an efficient service management layer a set of concepts must be defined and commonly used among the management agents. Such concepts mainly comprise the message types and interfaces involved as well as the discovery and negotiation protocols (Brandic et al. 2008b; Buyya et al. 2009; Hung, Li, and Jeng 2004; Ziegler et al. 2008).

2.2. Conceptual Foundations

In the following two sections the scientific context of my research project is presented. On the one hand, this includes mechanisms and concepts fundamental to the addressed research problem (and thus to both my dissertation as well as related research projects) and on the other hand related research problems, dealing with similar research questions like the one stated for this thesis. The former is being discussed in this section, the latter in the next.

2.2.1. Service Level Agreements in the Internet of Services

In order to derive a consistent definition of a SLA, the core concept it relates to, the service itself, must be defined. For this thesis the following definition is employed (see for example Berger 2005, pp. 12-18):

Definition 2.1 *A service is defined as an immaterial good that a SP delivers to a SC. As opposed to material goods, the creation and consumption of such a service coincide as a service is not storable.*

Definition 2.2 *An electronic service is consequently a service, which is requested, at least partially created, delivered and otherwise communicated with by employing standardized, electronic communication channels and other software systems.*

This definition stresses the point, that an electronic service, being the fundamental concept of the IoS scenario, is defined by its capability to communicate with its consumers over standardized, electronic communications channels, such as the Internet. Also it demands the service to be at least partially created using software systems. This excludes purely human-created services, such as a service a car mechanic would provide its customer.

On the other hand, semi-automated services, such as a translation service with additional manual proof-reading, can be seen as an electronic service, as long as it is requested and its results are returned over an Internet-based communication channel.

An electronic service is thus commonly described in terms of its functionality, mostly called its offered *operations*. In addition to this "syntactical" description, a *semantic* explanation of the offered functionality is given in many cases, allowing for automated discovery and subsequent invocation. In combination these two aspects answer the question on "What can a service deliver?".

On the other hand, the service quality it is able to deliver to the SC, describes, how a service delivers its offered functionality:

Definition 2.3 *The quality a service is provided in, mostly referred to as QoS, defines all characteristics of the delivered service with regard to its capability to achieve the desired results.*

The QoS a service delivers thus defines to what extent the service was able to satisfy the SC in terms of non-functional characteristics. These non-functional aspects (as opposed to the functional properties, defining the service's operations), such as response time or availability, vary over time, even for one and the same service instance. This results from changing workloads or other external factors, such as denial-of-service attacks or maintenance operations.

Fundamentally a service always provides the same set of operations, but it varies in how good it is able to provide them. This effect is exploited when negotiating about SLAs, as the QoS-affecting circumstances, such as priorities of the incoming requests, allocated resources, schedules of maintenance operations etc., can (mostly even in an automated fashion) be altered in favor or at the expense of an incoming service request. By re-scheduling or intelligent assignment of resources to individual service instantiations a desired QoS can be assured, ultimately allowing a SP to negotiate such QoS aspects within certain boundaries. These boundaries mark the range, which it can reliably achieve with its service without risking a SLA violation.

Building on this definition as well as the IoS scenario described before, a SP can be defined.

Definition 2.4 *A Service Provider (SP) is an organizational entity, managing and offering a non-empty set of services, whose primary goal is to engage in transactions with SCs, and in doing so, maximize its utility.*

SPs are thus not necessarily actual enterprises, but also departments within such enterprises or even governmental or research institutions. Their defining characteristic is the administration of one or more services that are offered to potential customers.

Definition 2.5 *A Service Consumer (SC) is an organizational entity in need of a distinct service, it is not able or willing to provide satisfactorily itself. It thus engages in a transaction with a SP able to deliver this service, and at the same time tries to maximize its utility resulting from that transaction.*

In analogy to a SP, a SC is defined by its need for a particular service, it cannot provide itself or at least not in a satisfying fashion. Again this concept is defined as an abstract entity, explicitly allowing it to represent an internal (other department of the same organization) or external (for example an actual customer) organization[1].

Definition 2.6 *Utility is used in this context as a subjective measure for the relative preference a given entity has on a set of different aspects of a transaction.*

Subjective means that the utility a given transaction result (for example a SLA) would yield for a given entity (SC or SP) can potentially differ significantly from the utility another entity associates with the same result. Utility is thus a way to weigh different aspects of an agreement and calculate the overall valuation of it with respect to a distinct individual, defining the respective weights. One SC could for example regard an early guaranteed delivery time as the only relevant SLA guarantee and not care at all about its costs and another one could define the exact opposing weights. In the end one given SLA would thus be valuated quite differently by these two SCs. How the set of agreement elements are weighed by a particular SC or SP is defined in its individual *Utility Function*.

The arena in which the individual SPs and SCs communicate to maximize their individual utility through agreements with one another is represented by the market they take part in.

Definition 2.7 *A market is an institutional context in which SPs and SCs meet and engage in transactions.*

[1] These two abstract definitions deliberately allow one organizational entity to act as both SP and SC. This is one of the core assumptions of the IoS scenario.

On a higher abstraction level the market thus coordinates demand and supply via the price. This economic control mechanism amongst individual actors (with potentially conflicting interests and thus utility preferences) is commonly referred to as the *invisible hand* (Smith 1976).

Having presented these fundamental concepts, a definition for the term SLA itself can now be addressed. However, there exist a variety of such definitions in the scientific literature. In order to derive a well-founded definition for this thesis, the most influential ones are now shortly presented and discussed with regard to the assumed IoS scenario.

A very abstract definition of a SLA is given in (Karaenke and Kirn 2007, p. 104): "Explicit formal statements of obligations and guarantees regarding grid services in a business relationship are often referred to as service level agreements (SLAs)." Although the authors mainly focus on GC environments, they already state the main characteristics of a SLA, namely obligations and guarantees regarding the delivered service(s).

Karten defines a SLA to be a "formal negotiated agreement which helps to identify expectations, clarify responsibilities, and facilitate communication between a service provider and its customers" (Karten 2003, p. 5) The main focus of this definition therefore lies in the specification of the goals to be achieved by a SLA. No concrete description of the actual elements of the SLA documents is given.

A definition much more oriented towards electronic services is given by the Internet Engineering Task Force (IETF)[2] in (Westerinen et al. 2001, p. 13). There, a SLA is defined as "[t]he documented result of a negotiation between a customer / consumer and a provider of a service, that specifies the levels of availability, serviceability, performance, operation or other attributes of the service." Here, very specific elements of a SLA documents are presented in terms of actually measurable service attributes upon which the service level guarantees can be defined.

Similarly the Global Grid Forum (GGF), which became the Open Grid Forum (OGF)[3] in September 2006, gives a definition of a SLA in the OGSA Glossary of Terms (Treadwell 2007, p. 11): They regard a SLA as a "contract between a provider and a user that specifies the level of service that is expected during the term of the contract. SLAs are used by vendors and customers, as well as internally by IT shops and their end users. They might specify availability requirements, response times for routine and ad hoc queries, and response time for problem resolution (network down, machine failure, etc.)." Again, a set of possible SLA elements is given in terms of measurable service characteristics. However, the authors also stress a point not present in most of the other definitions: the possibility for SLA usage within a given organization. This is rather new, as it abstracts from the image of an internal SP and an external organization acting as a SC (or vice versa), but rather employs this relationship for both, internal and cross-organizational interactions.

Keller et al. give a first hint at the possibility for electronic SLAs by stating "Web Services provide the opportunity to dynamically bind to services at run time, i.e., to enter and dismiss a business relationship with a service provider on a case-by-case basis and

[2] http://www.ietf.org/
[3] http://www.ogf.org/

on-demand [...]. Electronic contracts specify the way how these interactions are carried out and which contractual parties are involved. An important aspect of a contract for IT services is the set of Quality of Service (QoS) guarantees and the obligations of the various parties. This is commonly referred to as a Service Level Agreement (SLA)" (Keller and Ludwig 2003, p. 58).

Building on these definitions, the constituent characteristics of a SLA can be defined as:

- A SLA represents some sort of negotiated agreement between the provider and consumer of a service.
- It specifies the guaranteed characteristics for a given service invocation in terms of measurable service aspects, such as availability or performance.
- Additionally a set of organizational assertions, such as the cost associated with a service invocation or the penalties of not delivering the guaranteed service quality, is defined.
- SLA documents in this vein are used for cross-organizational as well as internal service deliveries.

Hence, the SLA concept as employed throughout this thesis is defined as follows.

Definition 2.8 *A SLA is a structured document, describing a negotiated, bilateral*[4] *agreement between a SP and a SC on the terms and conditions of the invocation(s) of a (set of) (electronic) service(s). This agreement both obliges the SP to deliver to the SC the respective service(s) in the stated quality and the SC to reciprocate this with a defined compensation payment. Hence, a SLA contains a description of the delivered services' functionality, guarantees on the delivered service quality, as well as assertions on the associated costs and penalties in case of a violation. The service quality guarantee is further defined as a set of Service Level Objectives (SLOs). Finally, a SLA is always valid for a distinct period of time.*

Hence, a SLA document basically describes a temporary business relationship between a SP and a SC, during which a defined set of services is delivered. It thus helps answering the questions "Who is providing the service(s) to whom?", "What services are provided and under which conditions?" and "What consequences arise from either a service being delivered as guaranteed or not?".

When defining these respective QoS levels within a SLA document, SLO elements are used:

Definition 2.9 *A Service Level Objective (SLO) describes one particular QoS aspect within a SLA document, thus consisting of a commonly known and measurable service characteristic together with a respective target value. This target value then represents the QoS guarantee for the respective service characteristic.*

[4] Also multi-lateral SLAs are possible. However, they only play a minor role in settings, such as the IoS, and are therefore not further considered in this thesis.

These definitions already show a fundamental gap between a SLA's business-related role and its implications on the technical infrastructure. On the one hand it is used as a contractual agreement between SP and SC, within or even across enterprise boundaries, with all resulting requirements from a legal perspective. On the other hand it regularly acts as an input for the, often even automated, management of the technical IT infrastructure delivering the requested services. It can clearly be seen, that both usage scenarios pose quite different requirements on the SLA document and its role in the overall service management process.

Some scientists even accredit this fact in defining a SLA only being a part of a comprehensive contract between a SP and a SC (Karaenke and Kirn 2007; Leff et al. 2003), focusing on the "operational definition of a service" (Karaenke and Kirn 2007, p. 104). Similarly, they distinguish between SLAs targeting end users and those targeting other SPs (Berger 2005, pp. 29-32).

In the following I will address this issue by shortly assessing the abstract SLA concept as underlying this thesis in the light of both high-level business relationships and the very precise task of automated SLA-based resource management.

SLAs as Instruments for Business Relationship Management

When investigating SLAs from a business perspective, scenarios like application service provisioning, web hosting or IT outsourcing are of primary interest (Keller and Ludwig 2003). All those settings are characterized by bipartite relationships between SPs and SCs, whose characteristics are regularly defined in terms of a SLA document (Goo, Kishore, and Rao 2009). It subsequently acts as a legally binding contract between the involved parties, denoting the rights, obligations and compensation payments associated with the respective business transaction.

The ongoing trend of concentrating on the core business and thus outsourcing a lot of the supporting functions (such as most parts of the IT infrastructure) in an enterprise to specialist providers constantly increases the need for SLAs as a foundation for such business relations. Additionally, the technical possibilities for more and more fine-grained definition (and thus providing and purchasing) of individual services was one of the main business-related drivers of the IoS as a model for future enterprise IS.

SLA documents play a very fundamental role in such IT outsourcing scenarios (Masche, Mckee, and Mitchell 2006). When relying on external services for one's own economic performance, some degree of planning reliability becomes necessary. This can be realized with the binding guarantees stated in a SLA.

On the other hand, most companies find themselves in a position where they not only purchase external services, but also sell those to external customers (potentially even by combining them with internal ones). The formerly internal and very technology-oriented IS of a company is increasingly moving towards a customer-oriented infrastructure for delivering services at least as parts of an economic transaction (Boehmann and Krcmar 2004).

In such cases, a SLA document helps the respective enterprise (acting as a classical SP in this case) to plan and allocate their internal resources in a way that the promised

service guarantees are met. Several best practice frameworks for managing such enterprise IS in an efficient way have been published in the recent years, the IT Infrastructure Library (ITIL) (Office of Government Commerce 2007) and Control Objectives for Information and Related Technology (COBIT) (IT Governance Institute 2007) probably being the two most prominent ones. Especially ITIL defines *Service Level Management* as the fundamental set of management activities concerned with the negotiation, monitoring and enforcement of service delivery processes according to agreed-upon SLAs (Mayerl et al. 2005; Schaaf 2008).

The role of a SLA, from a business perspective, is thus quite clear: it defines all rights and obligations of an actual business transaction and thus represents the legal basis for any interaction between SC and SP. It has to cover a lot of terms and elements related to these legal requirements in addition to the actual service (quality) description. Even SLAs between individual departments within a large cooperation relate to this concept, however they will regularly state internal prices and be able to omit some of the legal aspects due to the overarching legal framework of the parent company.

Finally, SLAs allow a SP to fully exploit the economic measure of price differentiation when defining individual service (quality) levels and their associated prices. Exploiting different utility functions on the SC sides, a SP can thus maximize its profit, given it is possible to enforce different quality levels during the actual service delivery process.

SLAs as Instruments for (Automated) Resource Management

In contrast to this very business-centric view on SLAs, such documents also play an important role in the (potentially automated) management of technical IS infrastructures. First attempts to apply SLAs for automated resource management have been done in the telecommunications industry (see for example Chakravorty et al. 2003; Mantar et al. 2006) and subsequently in, mainly non-commercial, grids (see for example Parkin, Badia, and Martrat 2008; Seidel et al. 2007)[5].

The main focus here is to extract the SLOs stated in the SLA, derive respective requirements posed on the managed infrastructure, calculate how this infrastructure must be orchestrated in order to fulfill them and finally implement the result.

An important part of this task is marked by the scheduling of incoming requests and their assignment to the available services or raw computing and storage resources (Seidel et al. 2007). Furthermore, a SLA can even be employed for well-founded predictions on potential violations and subsequent enforcement measures (Padgett, Djemame, and Dew 2005).

In order to allow such automated management of IT services a digitally processable version of a SLA is needed (Leff et al. 2003).

Definition 2.10 *An electronic SLA is a digital equivalent of a paper-based SLA, structuring its contents into machine-readable elements.*

[5] Going back even further, very initial works on object-oriented software engineering already applied the concept of a contract as the main communication concept between individual software components, governing the delivered functionality just as with today's SLAs (Meyer 1992).

The main focus of such an electronic SLA thus lies in the information needed for the automated scheduling and resource allocation, such as the stated SLOs, associated costs and penalties as well as involved parties (as needed for example in systems with individual users priorities). Hence, an electronic SLA aims at closing the gap between the high-level business context, as defined in a SLA between the involved business partners, and the very technical aspects of the underlying IS.

This aspect becomes even more prominent as current distributed and high-performance computing infrastructures more and more evolve from purely academic and non-profit to commercial infrastructures (Leff et al. 2003). In such cases SLA-based service provisioning and the possibility to offer "different service levels to different clients by dynamically allocating resources for execution of individual [...] service requests" (Dan, Ludwig, and Pacifici 2003, p. 1) becomes increasingly crucial for economic success.

As opposed to the business perspective presented in the last subsection, from a technical point of view a SLA simply has to define the guaranteed service quality and functionality (in terms of machine interpretable service metrics). This allows automated service management systems to calibrate the available IT resources in a way that the overall utility of the SP is maximized, based on the currently active SLAs and their associated compensation payments and penalties.

An Integrated View on SLAs

This distinction between inter-organizational, legally binding business contracts and internal agreements for the (automated) management can also be found in the ITIL framework. It distinguishes external contracts, called SLAs in ITIL, from internal agreements, which are much more related to the underlying IS infrastructure, called *Operational Level Agreements (OLA)* (Schaaf 2008, p. 46).

When taking a closer look on these two concepts, as well as the statements made in the last two subsections, it becomes quite obvious on an abstract level that they do not differ much. In both cases, an agreement document is created, stating the involved parties, the functional and non-functional aspects of the service to be exchanged as well as a set of organizational and other (for example legal) assertions on the respective transaction. Current projects, especially in the domain of high-performance computing, are even working on mechanisms to automatically map SLAs to OLAs, directly employed for the automated infrastructure management (Hasselmeyer et al. 2006b).

This shows the structural equivalence of both agreement types. On an abstract view a SLA thus consists of a set of elements. Berger gives a very extensive overview on such elements, covering both the IT-relevant service aspects and the legal and organizational elements (Berger 2005, p. 68).

He defines a SLA to contain:

- *contractual* elements: defining the involved parties and legal conditions.
- *service-related* elements: specifying the functional and QoS assertions of the delivered service, along with the associated costs.

- *management-related* elements: mainly detailing the applied monitoring mechanisms and penalties.

Especially the second, but also the third, set of elements could potentially be employed directly as OLAs (given they can be represented in a machine-processable way) for the management of the IT services involved.

Given this equivalency of internal and external SLAs and also taking into account the abstraction level of my research goal I will not distinguish between those two concepts for the remainder of this thesis. In the context of an automated SLA negotiation infrastructure, the primary requirement posed on a SLA document as the object under negotiation, is that it must consist of a set of defined elements. Each of those must have a defined identifier (ID) and a value, which is under negotiation. Consequently these elements can either contain simple metrics and their values or even complex legal expressions on compensation payments in case of a SLA violation.

Backing up this assumption, the IoS scenario further blurs the distinction between internal and external services and thus agreements about their invocations. Every functionality offered within or outside a given organization can be purchased and integrated into a given workflow. Subsequently the document describing the respective transaction will ultimately not differ any more from internal to cross-organizational invocations.

Note that at the moment SLA documents crossing enterprise boundaries are negotiated manually and signed by human beings. An infrastructure as defined within this thesis addresses the emerging need for automated service management from a technical perspective. Such an approach however implies that software agents, used as negotiators, must be able to derive legally binding contracts in order to implement an IoS as envisioned.

This aspect is currently heavily discussed by lawyers and legislative institutions, as it demands a significant change of the body of laws or at least their interpretation by the involved judges (see for example Wettig and Zehendner 2004).

For my thesis I will assume that a solution to this problem will be found. This is questionable, however initial approaches to electronic negotiations crossing company boundaries are already applied in some isolated industries. Here, a framework contract governs the overall federation of companies taking part in this "isolated" service economy. This framework contract provides the legal basis for software agents to actually conclude legally binding contracts.

Given such developments and the progress achieved in computer science and jurisprudence in that area indicate that the assumption of legal conformity of automated negotiations can be assumed valid.

Description Languages for Electronic SLAs

A SLA document marks an important input for the configuration of an electronic service which is part of a business transaction. In order to allow the automated negotiation and management of such agreements a structured digital document type is needed, specifying how the SLA elements can syntactically be expressed, a SLA description language.

Such a digital document schema regularly also allows for the definition of SLA templates (Ludwig et al. 2005).

Definition 2.11 *A SLA template is a not completely filled out SLA document, which is to be finalized during a negotiation process. In doing so, a SLA template defines the SLA elements, that are under negotiation, optionally gives starting values for those and finally states the rules, following which the final SLA can be derived during the negotiation.*

A template thus marks the starting point for every SLA negotiation, providing the SC with offered QoS metrics along with initial values offered by the SP. Based on these values the SC can then create a counter offer or accept the offered SLA directly, depending on the negotiation protocol applied. Additionally, SLA templates can be used to implement efficient discovery processes, as they can be published and retrieved by potential transaction partners according to many discovery protocols for electronic data (the most prominent of which will be presented in section 2.2.2).

In the following the most well-known SLA definition languages are shortly sketched.

SLAng The Department of Computer Science at the University College London developed a language for representing electronic SLAs, building on the Extensible Markup Language (XML): the *SLAng* language [6]. SLAng aims at providing a clear definition of the quality attributes associated with a digital service.

In contrast to some of the other languages presented in the following, SLAng only specifies non-functional QoS attributes of a service, the functional aspects are not covered with this language. Hence SLAng SLAs contain elements stating the contractual partners, context statements, such as duration of the agreement and the actual service guarantees, called *Service Level Specifications (SLS)* (Skene, Lamanna, and Emmerich 2004) in SLAng.

The SLAng language was derived using an Application Service Provider (ASP) reference model subsequently defining which types of SLAs are available within SLAng. In addition to vertical SLAs governing service delivery from more technical layers (for example communication) to higher-level ones (for example application services) SLAng also provides means for horizontal SLAs governing subcontracting of services on the same layer (Skene, Lamanna, and Emmerich 2004).

Based on this architectural scenario SLAng exposes a two-layer design: the language syntax is defined in terms of a *Unified Modelling Language (UML) 2.0* (2005) model, whereas the semantic of the SLA documents are stated in terms of *Object Constraint Language* (2010) constraints. Based on these constructs SLA designers can specify a number of different service metrics for each of the available SLA types, such as performance, availability or security (Lamanna, Skene, and Emmerich 2003).

Finally, SLAng allows for the detailed definition of timing constraints for each service guarantee, so-called *schedules*, describing which guarantee is to be valid at what point in time. This very generic timing concept and the formal semantics just mentioned allow

[6] http://uclslang.sourceforge.net/index.php

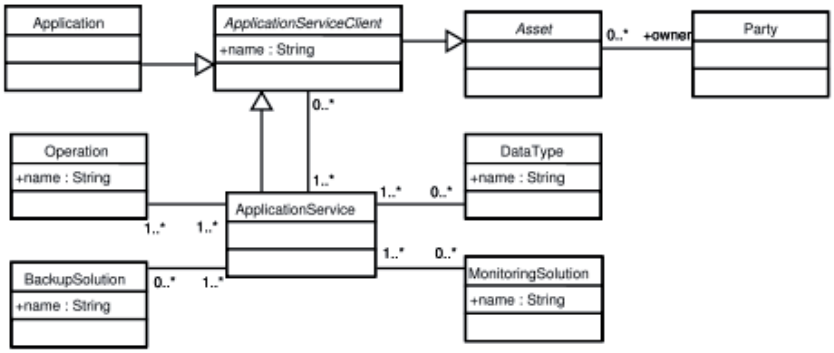


Figure 2.2.: SLAng "reference model for Application Services Provisioning" (Skene, Lamanna, and Emmerich 2004, p. 182)

for the automated validation and integration of different SLA documents. Additionally, not only the actual service properties, but also available monitoring and backup solutions as well as possible interaction behaviors of the involved parties can be modeled building on the expressiveness of the UML language (Skene, Lamanna, and Emmerich 2004). However the designers seem to have stopped working on that language in 2006 which makes it not very reasonable to rely on this concept for future SLA documents, especially since the authors of the corresponding website themselves advice SLA designers not to[7].

WSOL Another SLA language was designed by Tosic et al. and is called the *Web Services Offering Language (WSOL)* (Tosic, Patel, and Pagurek 2002). As already hinted in the name, this SLA language explicitly aims at supporting the Web Services technologies and therefore is compatible with related standards such as WSDL.

WSOL assumes web services to be offered in varying classes, marking a discrete configuration of the service along with its functional and non-functional attributes. Taking this into account a *service offering*, which can be expressed using the WSOL mechanism is "a formal representation of one class of service of one web service" (Tosic et al. 2002, p. 1). WSOL documents thus state a given service class and consequently the SLA associated with the services of that class.

In doing so, WSOL offers a set of language constructs to build a SLA document, such as *constraints*, *statements* and *constraint groups*. The constraints can be further subcategorized into *functional* (such as pre- or post-conditions of some service invocation) and *non-functional* (for example performance or availability of a service) constraints. A statement is basically some important information about the respective service that cannot be coded as a constraint, for example the price of a given service.

Finally, a constraint group allows for the, potentially recursive, grouping of individual constraints, allowing for example the assignment of different monitoring units to each of

[7] http://uclslang.sourceforge.net/index.php

the groups. For such groups WSOL also offers a template construct.

Building on this very low level language the authors also designed mechanisms for a structured Web Services Composition Management (WSCM) as well as a management infrastructure, applying WSOL documents for distributed service management (Tosic et al. 2004).

WSLA One of the most comprehensive SLA languages to date is the *Web Service Level Agreement (WSLA)* (Ludwig et al. 2003b) language developed by IBM[8]. This approach aimed at providing a specification for the definition and monitoring of SLAs in a Web Service environment. Employing such a language an automated management of electronic services based on the abstract notion of a contract was envisioned (Keller et al. 2002b).

The main design principles of WSLA are an unambiguous specification of Web Service SLAs, enabling an automated monitoring of the respective services, ease of SLA creation by introducing SLA templates and offering XML Schema specifications for the WSLA language as well as the incorporation of a distributed monitoring framework, in which relevant monitoring and evaluation tasks can be outsourced to independent third party monitoring services (Keller and Ludwig 2003).

To this end, WSLA not only defines the SCs and SPs (so-called *primary parties*) of the SLA but also secondary parties supporting the enactment of the contract, so called *supporting parties*. Such parties can be represented by measurement or condition evaluation services, implementing the monitoring of the SLA or management service responsible for taking respective management actions once a violation has occurred (Ludwig et al. 2003a).

A WSLA document thus not only describes the service related *metrics*, but also how they are supposed to be measured (*measurement directives*) in case of raw metrics and how they can be computed in case of higher-level metrics (in terms of aggregation functions over lower-level parameters) (Ludwig et al. 2003a).

Finally, a SLA adhering to the WSLA specification contains a set of *obligations* stating the SLOs and *action guarantees* of the respective SLA. A SLO represents a guarantee concerning one or more service metrics as opposed to an action guarantee which defines a promise to perform an action under a given condition.

Up to date the description of SLA elements concerning the validity of a given guarantee (in WSLA implemented as *validity periods*), the metrics it builds on, how to measure or calculate them as well as qualifying conditions on their measurement, has not been designed in a more detailed way as done with the WSLA language. Nevertheless WSLA is mostly focused on the sole purpose of describing a SLA document and does not cover how to create or even negotiate one.

Hence, a new SLA language has been developed based on the WSLA ideas, and partly by the same authors that represents the currently most widely used SLA language in service based systems: *Web Services Agreement (WS-Agreement)*. This standard simplifies the WSLA data model and accompanies it with a set of defined protocol primitives

[8] http://www.ibm.com/us/en/

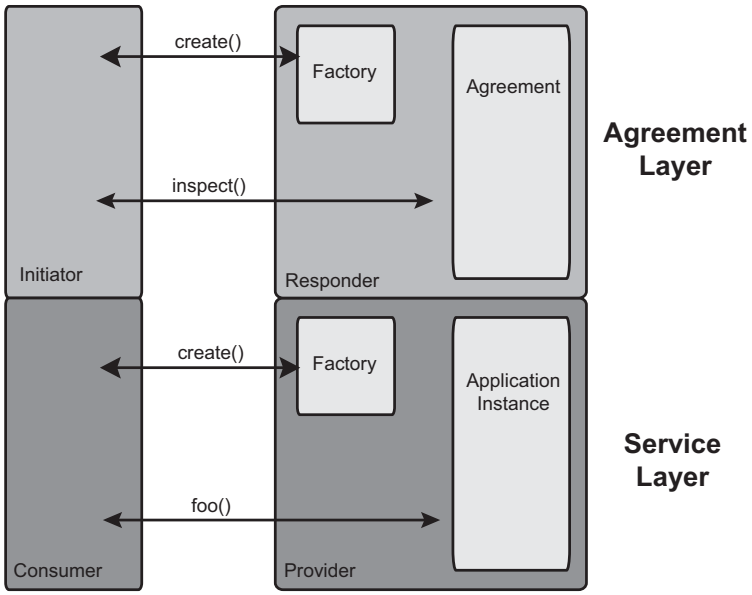


Figure 2.3.: WS-Agreement Architectural Model (Andrieux et al. 2007, p. 12)

and interfaces for agreement creation and monitoring.

WS-Agreement The WS-Agreement specification is a standardization effort conducted in the OGF in order to facilitate creation and monitoring agreements between a SP and a SC. The specification draft (Andrieux et al. 2007) defines an XML representation of agreements and agreement templates, a simple agreement establishment protocol as well as corresponding interfaces for creating an agreement and monitoring it at run time.

WS-Agreement defines two roles in creating agreements: *agreement initiator* and *agreement responder*. These two roles are completely independent from SP and SC, as both the consumer and the provider shall be able to initiate the agreement creation process.

WS-Agreement depicts a layered service model consisting of two layers: the *service layer* and on top of that the *agreement layer*. The *service layer* represents the domain- and application specific part of the proposed architecture. It contains the actual services the agreements are created for. "The *agreement layer* provides a Web service-based interface that can be used to create, represent and monitor agreements" (Andrieux et al. 2007, p. 12). In order to facilitate agreement creation, agreement templates can be offered just as in the WSOL approach.

In order to create a SLA, the agreement initiator proposes an agreement, optionally derived from a template. The agreement responder then checks the offered agreement and decides to accept or reject the offer according to its resource situation.

Agreements and agreement templates are both defined using the XML language. The high-level elements are illustrated in figure 2.4.

Figure 2.4.: WS-Agreement SLA Model (Andrieux et al. 2007, p. 14)

Every agreement and agreement template is identified by an *agreement id* attribute. This id has to be unique between the agreement initiator and responder. The main body of any agreement consists of an optional *name* element used for human understandability, a context section and the *agreement terms*. Agreement templates additionally contain a *creation constraint* section (Andrieux et al. 2007).

The context section contains information about initiator and responder of the agreement along with other metadata concerning the agreement as a whole. The agreement terms on the other hand define the main part of an agreement. They denote the obligations of the involved parties resulting from the agreement. The WS-Agreement defines two types of terms: *service terms* and *guarantee terms (GTs)*.

Service terms "provide information needed to instantiate or otherwise identify a service to which this agreement pertains and to which guarantee terms can apply" (Andrieux et al. 2007, p. 17). The *service terms* are further subcategorized as *service description (SDTs)*, *service reference (SRTs)* and *service property terms (SPTs)*. Service description terms define the functionality of the service for which the agreement is created. Service reference terms point to an existing service endpoint, to which the agreement relates, whereas service property terms define "measurable and exposed properties associated with a service" (Andrieux et al. 2007, p. 20). These measurable aspects of a service are described as a set of variables. Each variable relates to an attribute of the service and is associated with a metric to enable evaluation of this variable.

In order to define assurances on service quality for the described services, additional guarantee terms can be specified representing the service levels both parties are agreeing on. Each guarantee term specifies the obligated party of the guarantee which is needed for enabling consumer-side guarantees. It can also define an optional *qualifying condition* which must be met for the guarantee to be valid (Andrieux et al. 2007). Qualifying conditions are assertions over service attributes and / or external factors such as date or

time. The actual guarantee is described in the *service level objective* element. Also each guarantee is accompanied with some *business values*. Business values represent different value aspects of an agreement. This construct can be used to represent the importance of an agreement, the associated penalties and rewards for compliance to or violation of an agreement or preferences on different service configurations.

All elements described up until now are present for agreements as well as for agreement templates. However, agreement templates can be supplemented with an optional section called agreement *creation constraints*. This element specifies "constraints on possible values of terms for creating an agreement" (Andrieux et al. 2007, p. 29).

After having presented the foundations of service and SLA management from a static perspective, concentrating on the content and usage of SLA documents, the related processual aspects are presented in the following two subsections. Given the focus of my dissertation these comprise primarily two phases of the service life cycle (see figure 2.1): the discovery and the negotiation phase.

2.2.2. Discovery Phase

The main goal of the discovery phase is to bring together two or more potential transaction partners as a preparation of the following negotiation phase, in which the characteristics of the product subject to the transaction are determined.

Definition 2.12 *The discovery phase, as applied in the IoS, represents the process of a SC looking for a respective service to invoke, locating such a service and contacting the respective management component for a subsequent negotiation.*

Each SP must be able to promote its offered services in a way that as many potential SCs as possible are aware of this offering. On the other hand a SC must be able to access as many service offerings as possible in order to find an appropriate transaction partner.

Analog to real world transactions a SC must also be able to filter the offered services, and respective SLA offerings, according to some search criteria regularly received from a human user or workflow orchestration engine. This possibility to filter potential offerings becomes even more important in service-based infrastructures employing electronic SLAs. In such settings each offered service does not only provide potential transaction partners with a set of descriptions of the offered functionality but also with non-functional parameters, guaranteed for each service invocation. SCs can not only search for fitting services, in terms of the functionality they provide, but also for the ones doing it best in terms of offered QoS.

In the recent years a set of different mechanisms supporting the discovery and publication of electronic services have been developed (see for example Detken 2004; Hasselmeyer 2005). In the following I shortly sketch the most promising approaches with respect to the anticipated IoS scenario.

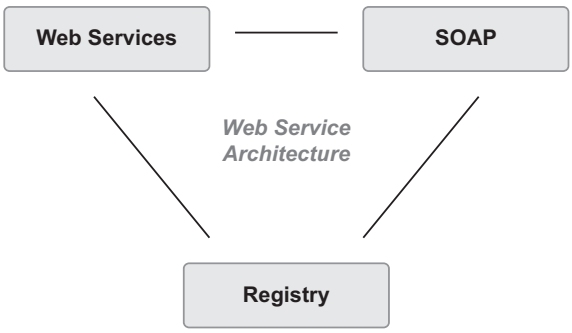


Figure 2.5.: Relation between Web Services Standards

Central Registries

Central registries represent a very common discovery mechanism. According to Hasselmeyer (2005, p. 3) a service registry is a "service that provides references to other services. It accepts requests for registration from services [...] and relays registration information on demand to clients."

The main task of a service registry is thus to store relevant information of available service offerings (including the offered SLA templates) and provide this information to respective SCs. For this purpose a defined data scheme for internal representation of service data along with a set of query and registration methods are offered. However, mostly no mechanisms are provided for finding such a central registry; this issue is transferred to the actual service designers and deployers.

Some of the most common registry standards currently available are UDDI or the ebXML[9] Registry, both proposed by the Organization for the Advancement of Structured Information Standards (OASIS) committee. Registry services like that are seen as one of the three core parts of SOC, along with WSDL used to define electronic services and the SOAP protocol as a way to access them.

Following the abstract consideration for a service registry, UDDI for example defines a data model for the description of services and their providers along with a defined interface for registration and discovery of services. The UDDI standard mainly builds on the separation of service related data into three different types, represented as different *pages* respectively:

- Information stored as *white pages* contains general data of a service provider, such as the company name, contact data or other descriptive information.
- *Yellow pages* serve the same purpose as their pendant in the real world, the assign the service (providers) to a set of categories which eases the discovery process.

[9] http://www.ebxml.org/

- Finally the *green pages* contain technical information concerning the offered web services, such as syntax of the offered methods and the endpoint of the services. WSDL documents are thus regularly referenced in these green pages.

Employing the SOAP protocol, SPs and SCs can register and discover respective data sets at a given registry service. Especially the relation between SPs and actual service offerings on a database level allows for querying trusted business partners and then choosing one of their offered services (Hasselmeyer 2005).

Multi- or Broadcast-based Mechanisms

Another set of discovery mechanisms focus much more on the actual discovery process as on the internal data model, while at the same time targeting massively decentralized architectures. These systems can roughly be characterized as *multi- or broadcast-based* systems.

Such mechanisms rely on multi- or broadcast communication (in the following subsumed as *manycast*) in order to find or promote a given service offering over a communication network. SPs send manycast messages describing the service(s) they offer, hoping that a respective SC listens and responds to such a message. Contrarily, a SC can demand a particular service by sending a manycast message containing its needs to potential SPs, hoping one of them is offering a fitting service.

Manycast-based systems produce a massive amount of traffic, due to the non-directed communication paradigm. Some of the protocols described in the following will employ mediating third parties in order to address this issue.

One of the most important protocols building on manycast communications is the *Service Location Protocol (SLP)*. It was published as an IETF Request for Comments (RFC)[10]. In contrast to UDDI, only the discovery protocol and not the applied data model is specified along with the notion of *service agents* as service management components, which follows the rationale of this thesis.

The roles present in the SLP are *service agents*, acting on behalf of one or more services to be published, *user agents*, querying the offered services and optional *directory agents* acting as registries.

Whether or not directory agents are used, depends on the size of the network; in small networks they might be omitted and only manycast communication may be used. If present however, manycast messages are only used for discovering these registries and afterwards unicast communication is applied between SPs and SCs for efficiency reasons.

In order to reach a high robustness, service registrations are lease-based. A service agent must renew its lease, and thus state that the respective service offering is still valid, within fixed time intervals.

A very similar approach is marked by the *Web Services Dynamic Discovery (WS-Discovery)* protocol (Beatty et al. 2005), which was developed by a large industrial

[10] http://www.ietf.org/rfc/rfc2608.txt

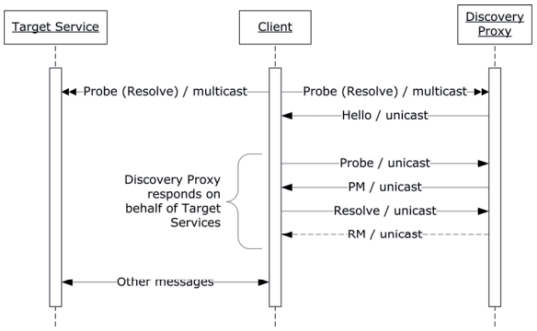


Figure 2.6.: WS-Discovery Protocol with Discovery Proxies (Beatty et al. 2005, p. 13)

consortium consisting of BEA Systems (now owned by Oracle)[11], Microsoft[12], Canon[13], Intel[14] and WebMethods (now owned by Software AG)[15].

In WS-Discovery a service joining the network sends a *hello* manycast message in order to notify potential consumers of its presence. These service clients either listen for such hello messages (*passive discovery*) or send a *probe* manycast message stating the desired service type or scope (used for semantic grouping of services). SPs offering an appropriate service respond via a unicast message. For discovery of a service by its name, a *resolution* manycast message is sent accordingly, which is also answered by the SP agent offering the service respectively.

To minimize manycast traffic, mediator services, so-called *discovery proxies* (see figure 2.6) can be present. These answer to any manycast message (probe or resolve) with a hello unicast message announcing them. From then on the SCs switch from regular to mediated mode and only contact these proxies in order to query potential services (unicast). The proxy service from now on answers on behalf of the individual SPs.

Peer-to-Peer-based Mechanisms

Peer-to-Peer (P2P) systems are characterized by a highly dynamic set of participating entities (called *peers*), constantly interacting in a cooperative manner (Hasselmeyer 2005). All peers act as "both [...] a client and as a server ("servant") and pay [their] participation by providing access to some of [their] resources" (Aberer and Hauswirth 2002, p. 1). These resources are mostly media or other files that are distributed within a P2P network. Prominent examples for P2P-systems are the Gnutella[16] or BitTorrent[17] infrastructures.

[11] http://www.oracle.com/bea/index.html
[12] http://www.microsoft.com/en/us/default.aspx
[13] http://www.canon.com/
[14] http://www.intel.com/
[15] http://www.softwareag.com/corporate/default.asp
[16] http://wiki.limewire.org/index.php?title=GDF
[17] http://www.bittorrent.com/

P2P systems follow three very fundamental design principles (Aberer and Hauswirth 2002):

- *resource sharing*: by combining distributed resources, more complex tasks can be undertaken (similar to GC).
- *decentralization*: the involved resources are physically distributed among the peers, thus avoiding central bottlenecks.
- *self-organization* of the peers: no central coordination node is present.

In such massively decentralized and dynamic systems the discovery of a given resource becomes one of the main problems to deal with. When looking at the applied discovery protocols, P2P systems can be distinguished into two major classes: *unstructured* and *structured* systems (Steinmetz and Wehrle 2004).

In unstructured P2P systems a requesting node has no information on where the desired digital resource can be found within the network. Regularly, so called *flooding* protocols are employed. In a flooding protocol the requesting node sends a query message to all neighboring peers and these pass the message on throughout the network. Whenever a node offering the requested resource receives the query it simply answers the requesting agent directly (Steinmetz and Wehrle 2004). It is quite obvious that such an approach scales only very badly for very large networks (Kelaskar et al. 2002).

Trying to address this problem, structured P2P networks implement a distributed indexing mechanism in a way that each peer is responsible for a distinct subset of the available resources, each of which is uniquely identified with a key value. Whenever a SC node requests a resource, it can use this indexing mechanism to locate the peer offering the requested resource directly and does not have to broadcast the query message into the whole network (Steinmetz and Wehrle 2004). These mechanisms are also called *Distributed Hash Tables (DHT)* as they port the idea of key-value pairs for retrieving and storing data to distributed systems and allow simple put- or get-operations to the requesting application, just as with a regular hash table (Balakrishnan et al. 2003).

The probably most well-known DHT mechanism is Chord (Stoica et al. 2001). In a Chord system the participating nodes are associated with unique IDs and structured into a logical ring, in which each participant has exactly one predecessor and one successor node. The ID, created by a hash function, determines the position of the peer in the ring.

By employing these hash-generated IDs, the peer responsible for storing a distinct data (and thus the peer that is to be asked during later discovery processes) can be determined. Whenever a peer requests a resource it just creates the hash value of the respective resource ID (for example service ID) and subsequently retrieves the responsible node (the respective range of data files this node is responsible for depends on its own ID and thus position in the ring).

Employing Chord-like discovery mechanisms potentially allows for leveraging both the robustness of distributed P2P systems on the one hand and a scalable search mechanism on the other.

After having given an overview on discovery protocols and system architectures, I will now elaborate on the subsequent phase of the SLA management life cycle: the negotiation phase. In this phase the potential transaction partners (that have found each other during the discovery phase) actually engage in a negotiation process, which if concluded successfully, results in an agreed-upon SLA document, acting as an input for the following phases of the life cycle.

2.2.3. Negotiation Phase

Regularly, a negotiation on the found service(s) follows the discovery phase. This raises the fundamental question on why such a negotiation is at all necessary. Why don't we just implement configurations where all resources are under the control of the parties needing them? Why is there a need for coordination between a set of parties concerning some distinct resources?

A lot of research groups work on questions like that, investigating in which situations negotiations can and should be applied as well as what protocols to use in various settings (see for example Bichler, Kersten, and Strecker 2003; Kersten, Law, and Strecker 2004; Kersten and Noronha 1999; Neumann et al. 2003; Ströbel and Weinhardt 2003; Wurman, Wellman, and Walsh 1998, 2001). These numerous efforts basically resulted in two distinct findings:

1. Negotiations are necessary in certain situations.
2. Potentially every negotiation situation requires a different protocol to result in efficient outcomes.

In all situations where each side cannot achieve its goal locally and depends on some other party, some negotiation mechanism is necessary for reaching an agreement on the subject matter of conflict. This mutual dependency of the negotiating parties is referred to as the key necessary condition for negotiations to occur (Ströbel 2000b). Sufficient conditions necessary for negotiation in a narrow sense, as defined above, are according to Ströbel (2000a,b) for example:

- unique goods
- non-repetitive, non-standard transactions
- unknown or very dynamic demand or supply
- unknown consumer value perceptions
- perishable goods

This assumption holds for a variety of situations in either the real (e.g. market places for any types of goods, contract negotiations between countries or business contracts between participants of a given supply chain etc.) or the electronic world (e.g. SLA

negotiations in the IoS). All of those scenarios are made up of individual nodes requesting resources or goods from each other, therefore fulfilling the necessary condition.

Also at least one sufficient condition can be found for the IoS scenario, where the present services are only available for a given timeslot and therefore exhibit all characteristics of perishable goods. Also the complex structure implies very dynamic demand and supply, fulfilling the third item listed.

After having shown the necessity of negotiations in the anticipated IoS setting, I will now derive a definition of the mere term negotiation, as to be used for the remainder of this thesis.

Since the contexts in which negotiations take place are numerous there are many different definitions of negotiation processes describing the concept from different perspectives. Pruitt for example describes negotiations as "a form of decision making in which two or more parties talk with one another in an effort to resolve their opposing interests" (Pruitt 1981, p. xi). "The parties first verbalize contradictory demands and then move toward agreement by a process of concession making or search for new alternatives" (Pruitt 1981, p. 1).

In my thesis I will use the following definition, as it focuses much more on electronic environments:

Definition 2.13 *A negotiation represents an "iterative communication and decision making process[...] between two or more agents (parties or representatives) who: 1) cannot achieve their objectives through unilateral actions; 2) exchange information comprising offers, counter-offers and arguments; 3) deal with interdependent tasks; and 4) search for a consensus which is a compromise decision" (Bichler, Kersten, and Strecker 2003, p. 318).*

This definition explicitly describes the information exchanged in the communication process as offers and counter-offers. Additionally the parties involved in a negotiation are further specified to be the parties, affected by the consensus to be reached, or alternatively agents acting on their behalf, which fits the IoS scenario very well.

The theoretical foundations of the service negotiation phase presented in the following are subdivided into the already introduced perspectives on negotiation processes: *protocols*, *objects* and *strategies / participants*[18].

Negotiation Protocols

In general, a protocol describes the rules of an interaction between two or more parties. This involves the definition of states the individual participants can be in, the possible actions of the participants dependent of the current state and, optionally, particular stimuli like received messages for example. Another facet of protocols is to define the messages and message formats used in the interactions.

[18] According to the research question to be answered in this thesis, I will primarily focus on the protocol perspective.

Definition 2.14 *A negotiation protocol describes the sequence and exchanged content of a negotiation process between the involved parties.*

To give a short initial overview on existing negotiation protocols, a taxonomy is derived in the following. It is based on other research works found in the literature, originating in software agent, negotiation and electronic market research (see for example Bartolini, Preist, and Jennings 2005; Bichler and Kalagnanam 2006; Hudert et al. 2009; Lomuscio, Wooldridge, and Jennings 2003; Ströbel and Weinhardt 2003; Wurman, Wellman, and Walsh 1998, 2001).

Essentially, negotiation protocols can be very roughly categorized according to their overall configuration, e.g. the number and distribution of negotiating agents. Such a categorization will result in the following classes of protocols:

- In *1:N negotiations* one seller agent and an arbitrary set of buyer agents take part. The buyers post bids to the one seller agent which in turn chooses the best posted offer and engages in an agreement with the sender of this bid. Such 1:N situations are called *auctions*.

- *N:1 negotiations* represent the class of *reverse auctions*. Reverse auctions consist of one buyer and a set of sellers posting bids. The buyer then chooses the best bid analogously to regular auctions, except that the best bid in a reverse auction is the one with the lowest stated price whereas in regular auctions the highest posted price wins.

- In *N:M negotiation* settings both, seller and buyer sets, can consist of an arbitrary number of agents. Each agent is allowed to post bids to a central market instance which matches offer and demand.

- *1:1 negotiations* comprise two agents exchanging bids in order to reach an agreement. 1:1 negotiations are also called *bargaining* scenarios.

When trying to further subcategorize these classes a more detailed set of characteristics can be employed:

- *Roles*: The roles attribute specifies which roles are present in a negotiation. In general two roles are inherently participating in a negotiation: SP and SC. Additionally, an optional mediator role can be present, allowing for the implementation of central market instances (often called market *brokers*), auctioneers or trusted third parties.

- *Agents*: This parameter defines the minimum and maximum number of agents allowed to join a negotiation in a given role (1:1, 1:N, N:1, N:M).

- *Sides*: The sides parameter specifies which of the involved roles are allowed to post bids.

- *Status*: The status flag indicates whether the negotiation status, e.g. the currently winning bid, can be accessed by the negotiating agents or not. This allows for definition of *sealed-bid* negotiations.

Based on the shown taxonomy each negotiation protocol type can be defined as a tuple of the following form: $< SP(a,b,c), SC(a,b,c), Med(a,b), S >$

For each of the roles present in a negotiation protocol (SP, SC and mediator) the minimum (a) and maximum (b) number of agents allowed is presented along with a flag indicating whether this / these agent(s) are allowed to post bids (0 indicating they are not, 1 indicating they are allowed to). Posting bids is only possible for service providers and / or service consumers and therefore the third parameter (c) is only present for these two roles. Finally another flag is given defining whether the current status of the negotiation (S) is accessible or not, e.g. whether the agents can access the current bid or not.

English Auction
$< SP(1,1,0), SC(1,N,1), Med(0,1), 1 >$
In an *English Auction (EA)* the price of the good under negotiation steadily increases until no bidder is willing to pay the next price step. The bidder having bid the last, and therefore highest, valid bid will get the agreement and win the auction.

An EA always has exactly one seller (SP) and an arbitrary number of buyers (SC). Depending on the implementation there can be at most one mediator present, if not the SP also acts as auctioneer and no mediator is needed. In the EA the currently winning bid is always known to the negotiators, so the status flag is set to 1.

Dutch Auction
$< SP(1,1,0), SC(1,N,1), Med(0,1), 1 >$
During a *Dutch Auction (DA)* the price for the negotiated good is set quite high at the beginning and then lowered during the auction process. Whenever an agent decides to buy the product for the currently given price it posts the bid and engages in the agreement.

Analogous to the EA there is one SP, possibly an indefinite number of SCs and an optional mediator.

Vickrey / First-price-sealed-bid (FPSB) Auction
$< SP(1,1,0), SC(1,N,1), Med(0,1), 0 >$
In a *Vickrey* or *FPSB Auction* the negotiators only post one bid each. They do so in a sealed way, e.g. the other negotiators are not able to access the content of the posted bids. After a certain event occurs the auction clears and the bid offering the (second) highest price wins[19]. Concerning the involved roles and numbers of agents in each role the Vickrey and FPSB Auctions do not differ from EA / DA protocols. However, in a Vickrey / FPSB Auction the status is not visible to the agents.

[19] In a FPSB Auction the bid offering the highest price and in a Vickrey Auction the one offering the second highest bid wins.

Reverse English Auction
$< SP(1, N, 1), SC(1, 1, 0), Med(0, 1), 1 >$
A *Reverse English Auction (REA)* is basically the same protocol as a regular EA except that there is only one buyer and many sellers. This way the price for the negotiated good is steadily decreasing from a starting price. In the end the seller willing to accept the agreement for the lowest price wins the auction.

Reverse Dutch Auction
$< SP(1, N, 1), SC(1, 1, 0), Med(0, 1), 1 >$
In a *Reverse Dutch Auction (RDA)* there is only one buyer and possibly many sellers as in a REA. Now the product price is simply increased by a certain step in each round. The first seller accepting the current price step posts a bid and wins the negotiation by doing so.

Reverse Vickrey / FPSB Auction
$< SP(1, N, 1), SC(1, 1, 0), Med(0, 1), 0 >$
During a *Reverse Vickrey / FPSB Auction* protocol each of the N sellers posts one sealed offer. After a certain event occurs, the best offer, e.g. the one stating the (second) lowest price will result in an agreement. This differs from a regular Vickrey / FPSB Auction in having only one buyer and possibly many sellers, respectively.

Call Market
$< SP(1, N, 1), SC(1, M, 1), Med(1, 1), 0 >$
A *Call Market (CM)* is characterized as a scenario, consisting of a set of sellers and buyers posting bids to sell or buy to a central market broker (mediator). This broker then matches offer and demand messages according to some defined solver algorithm. This matching regularly takes place after certain intervals of bidding. Each offer sent to the market instance is sent in a sealed manner, so that the other agents on the market cannot adapt to posted bids themselves.

Continuous Double Auction (CDA)
$< SP(1, N, 1), SC(1, M, 1), Med(1, 1), 1 >$
A CDA exhibits the same characteristics as a Call Market except that the posted bids are submitted visible to all market participants unlike with a Call Market.

Bargaining
$< SP(1, 1, 1), SC(1, 1, 1), Med(0, 0), 1 >$
In one-on-one bargaining negotiations two agents negotiate by exchanging offers and counter offers. The agents take turns in posting the bids to one another until one of the agents is offered an agreement it can accept. Then the respective agent just sends an accept message back and the agreement is in place. If no agreement can be reached, for example if the accept intervals for the price of both agents do not overlap, the negotiation

Table 2.1.: Negotiation Protocol Types

Protocol	Attribute Values
English Auction.	$< SP(1,1,0), SC(1,N,1), Med(0,1), 1 >$
Dutch Auction	$< SP(1,1,0), SC(1,N,1), Med(0,1), 1 >$
Vickrey/FPSB Auction	$< SP(1,1,0), SC(1,N,1), Med(0,1), 0 >$
Reverse English Auction.	$< SP(1,N,1), SC(1,1,0), Med(0,1), 1 >$
Reverse Dutch Auction	$< SP(1,N,1), SC(1,1,0), Med(0,1), 1 >$
Reverse Vickrey/FPSB	$< SP(1,N,1), SC(1,1,0), Med(0,1), 0 >$
Call Market	$< SP(1,N,1), SC(1,M,1), Med(1,1), 0 >$
CDA	$< SP(1,N,1), SC(1,M,1), Med(1,1), 1 >$
One-On-One Barg.	$< SP(1,1,1), SC(1,1,1), Med(0,0), 1 >$

is canceled after a certain condition occurs (an example would be a certain number of offers sent).

As it is the most commonly used approach for SLA definition and negotiation I want to give a short remark on the protocol proposed in the WS-Agreement (Andrieux et al. 2007) and WS-Agreement Negotiation (Waeldrich et al. 2010) specifications respectively. The former offers only a very simple protocol in which an initiator can query the available SLA templates from a responder, chose the one fitting best and create an agreement offer from it. This is subsequently proposed to the responder node, which in turn can accept or reject it. No counter offer possibilities are present in this protocol. WS-Agreement Negotiation extends this model by introducing a new Negotiation Layer, above the Service and Agreement Layers. This layer comprises *Negotiation Factory* and actual negotiation components, allowing the involved parties to create a negotiation (implemented as a service instance from the factory) and post offers to each other. Given both parties offer the mentioned negotiation components an actual bargaining protocol, consisting of offers and counter-offers is possible.

Table 2.1 shows the formal description of the protocol types just identified.

For all of these types a variety of more or less complex sub-types have been developed, each varying in terms of the sold product, allowed messages or message sequences, matching algorithms etc. (see for example Bichler 2001; Ermolayev and Keberle 2006; Li, Giampapa, and Sycara 2003; Smith 1980; Ströbel and Weinhardt 2003; Walsh, Wellman,

and Ygge 2000; Wurman, Walsh, and Wellman 1998a)[20].

Negotiation (Protocol) Description Languages In the past years quite a lot of work has been done in defining languages and taxonomies for the formal description of negotiation protocols. Since my negotiation framework also builds on a structured protocol description in order to achieve protocol-generity (see chapter 3), I now give a short overview on such languages.

Ontology-based languages mostly describe not only the negotiation protocol itself, but also model the applied strategies (Ermolayev and Keberle 2006) and other potentially private aspects of the negotiators. In doing so, a set of commonly known ontologies along with semantic reasoning mechanisms is applied.

Languages based on parameters aim at describing a negotiation protocol by a set of process attributes and respective values. Due to this very simple approach it is not surprising that many of the languages found follow that paradigm.

Lomuscio et al. proposed a "Classification Scheme for Negotiation in Electronic Commerce" (Lomuscio, Wooldridge, and Jennings 2003). This approach, however, focuses on pre-negotiation phases and assumes high human interaction rates, contradicting with the IoS scenario.

Wurman et al. presented a set of auction parameters while developing an Internet-based "platform for price-based negotiation - the Michigan Internet AuctionBot" (Wurman, Wellman, and Walsh 1998, p. 1) . This system was designed to serve as an auction server for humans as well as for software agents. Unfortunately, the focus is on one single negotiated issue: the price. Thus, only auction protocols are supported. The same authors extended this taxonomy to also cover multidimensional auctions in a follow-up, much more comprehensive paper (Wurman, Wellman, and Walsh 2001).

The most comprehensive parameter-based description language so far was published by Ströbel and Weinhardt (2003). The classification scheme presented there covers a very comprehensive set of negotiation protocols, both from the human-centric and automated perspective, without stressing technology-related issues or focusing just on a subset of possible negotiation protocols.

In general, parameter-based approaches are often deemed insufficient for describing temporal patterns of negotiations, such as the sequence of possible events or actions (Lochner and Wellman 2004).

Rule-based languages on the other hand, try to tackle this problem by specifying action rules that prescribe what a negotiating agent has to do under certain conditions. By also incorporating user-defined variables, such as counters etc., rule-based systems can offer a high flexibility to the auction designer (Lochner and Wellman 2004). A good example for such an approach is (Wurman, Wellman, and Walsh 2002).

Finally, *state automata* can be used to define the behavior of a given service or a whole system. This concept allows for the definition of internal states and possible state

[20] A subset of these protocols has also been published by the Foundation for Intelligent Physical Agents (FIPA) , as part of their protocol library for agent-based communications, which can be found under http://www.fipa.org/.

changes along with optional conditions for such changes as well as triggered events when entering or leaving a respective state (Martin 2002).

In this thesis, a description language building on parameters, optional rule-elements as well as state machines is employed. For more details see subsection 3.2.2.

Negotiation Objects

Definition 2.15 *The good under negotiation (in the IoS represented by the offered electronic services and thus SLAs) is commonly referred to as the negotiation object.*

Depending on the number of negotiable characteristics, negotiation objects can be distinguished on a first level into *single-* and *multi-attributive* ones (a distinction also sometimes used for the negotiation processes concerning such products) (Lai et al. 2004).

Single-attributive negotiation objects expose only one negotiable attribute, the price. Even negotiation objects that are more complex, but only offer one negotiable attribute are considered single-attributive (with respect to the negotiation they are involved in). Within the IoS setting such mechanisms could be applied in situations where commodity services in a standard configuration are traded, only offering the price attribute to be negotiable.

Accordingly, multi-attributive negotiation objects expose multiple different negotiable attributes. Apart from the price, such attributes could comprise the date of delivery or maximum response time etc. For the IoS scenario multi-attributive negotiation processes deem appropriate, as a SLA regularly comprises several SLOs of which mostly more than one are negotiable.

In a second dimension, negotiation objects can be distinguished according to whether they represent a single good (instance of a service in the IoS setting) that is under negotiation or a whole bundle of such goods. Traditional negotiation taxonomies assume a single negotiation object (which is single- or multi-attributive respectively). Recent developments in negotiation research identified two distinct classes of negotiations concerning bundled negotiation objects: *combinatorial* and *multi-unit* negotiations.

In combinatorial negotiations bidders "place bids on combinations of items, called "packages" rather than just individual items" (Cramton, Shoham, and Steinberg 2006, p. 1). Each negotiation object therefore represents an offer describing the desired bundle of (potentially quite different) items and the price one offers to pay for it. This is especially useful when complementaries are present among the items to be sold[21].

In multi-unit negotiations on the other hand the negotiation object comprises several units of the same product to be sold within as a bundle. They differ from combinatorial negotiations in that the individual units do not differ significantly and are treated interchangeably throughout the negotiation. The resulting agreement thus concerns a set of identical goods that are sold.

As each service in the IoS is referenced individually it therefore defines an individual item, treated in an own negotiation process. If a combination of individual services or service units are to be sold as a bundle this will regularly result in them being combined

[21] Items are complementary if the utility of a set of items succeeds the sum of the individual utilities.

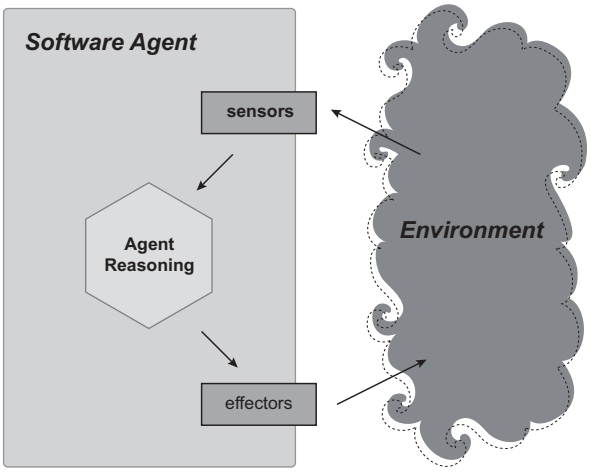


Figure 2.7.: Abstract Architecture of a Software Agent (Russell and Norvig 1995, p. 45)

to a higher-order, complex service. Combinatorial and multi-unit negotiations are thus not appropriate in the anticipated setting and will not be in the primary focus of my work.

Negotiation Strategies / Participants

When describing the actors within a negotiation process, three types of negotiations can be distinguished with regard to their level of automation: *unsupported*, *supported* and *automated* (electronic) negotiations (Bichler, Kersten, and Strecker 2003).

In an unsupported negotiation process humans negotiate about a given good without any help of an IT system. During a supported negotiation the involved people delegate some of the negotiation-related task to software components (e.g. creation of a counter-offer or weighing of different offer alternatives). Finally, a (fully) automated negotiation depicts the process "involv[ing] software agents that make decisions and control the entire process, including the specification of offers and concessions, and the final decision about agreement or disagreement" (Bichler, Kersten, and Strecker 2003, p. 322).

The IoS scenario is characterized by a huge amount of economic transactions as well as very high interaction rates. Most of the purchased service invocations happen within the context of a business workflow, most of which are time-critical and therefore show a tendency to high automation. Workflow engines will need to request, buy and integrate external services much more rapidly than would ever be possible with human negotiators. Even accompanied with negotiation support systems, increasing the efficiency of negotiation processes dramatically, such an approach would still not be appropriate within a setting where possibly thousands of negotiations happen in parallel at a given time and up to a couple of hundreds would have to be conducted by one particular user. Thus, I

deem the application of software agents for negotiations in the IoS mandatory and focus on these for the remainder of this thesis.

One of the most well-known definitions for a software agent was given in (Jennings 2000, p. 280):

Definition 2.16 *"An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives" (Jennings 2000).*

The defining architectural characteristics of an agent are thus that it is situated in an environment, some aspects of which it can observe with some sensor mechanism and within which it can undertake some actions, using some effectors at its disposal (Russell and Norvig 1995, p. 31). Figure 2.7 illustrates this abstract architecture of a software agent.

In contrast to simple reactive devices, such as passive software demons, agents are accredited to have the following defining characteristics (see for example (Wooldridge 1997):

- *Reactivity*
- *Proactiveness*
- *Autonomy*
- *Social-Ability*

Reactivity The claim for reactivity states that a software agent must be capable of observing its environment (consisting of human users, other simple software components or even sophisticated peer agents) and "respond in a timely fashion to changes that occur in it in order to satisfy [its] goals" (Wooldridge 2005, p. 23). In order to do so, an agent must not only be able to observe and subsequently act within its environment, but also must be equipped with an internal processing mechanism choosing the action to be undertaken after a given stimulus occurred.

Proactiveness Software agents are generally supposed to "exhibit goal-directed behavior" (Wooldridge 2005, p. 23). They are assumed to proactively pursue the goals received from a human user or other software device. In contrast to purely reactive systems a software agent therefore has to anticipate results of its own actions and assess its options in the light of its goals. It will thus choose actions that will bring it closer to this goal over others. Again such a behavior demands some cognitive capabilities (commonly referred to as *strategy*) and experience with actions chosen in the past and results received then.

Autonomy Autonomy refers to the way an agent reacts to a stimulus received through one of its sensors. When being capable of autonomous actions an agent is assumed to not simply react deterministically to external stimuli, but rather to choose its actions with respect to the experiences it has made over time (Russell and Norvig 1995). This implies that agents have a means for remembering past decisions, and respectively taken actions, and the so induced environmental changes and are capable to incorporate these "learned coherencies" into their current behavior. Autonomy thus adheres to adaptivity to some point, as agents are supposed to "be able to operate successfully in a wide variety of environments, given sufficient time to adapt" (Russell and Norvig 1995, p. 35). Commonly, such an adaptive behavior is achieved using automated learning algorithms building on for example genetic algorithms (Goldberg 1989) or neuronal networks (Haykin 1994).

Social-Ability The last aspect pays tribute to the mere vision underlying the software agent paradigm: decentralization. Software agents are individual nodes, interconnected with each other and the environment they reside in. Each is trying to reach its individual goals, which cannot be achieved unilaterally. Consequently the agents interact in order to jointly reach goals they would not be able to achieve alone.

Without stressing actual implementation techniques for software agents, these four characteristics already show the potential software agents offer for decentralized settings such as the IoS. Their ability to decentrally coordinate, cooperate and negotiate (Wooldridge 2005, p. 3) fits excellent with the demanding problem of cross-organizational service management. Initial works have already been done to port software agent technologies to this problem domain especially when concerned with the negotiation of service agreements (see for example Buyya, Abramson, and Venugopal 2005; Eymann, Streitberger, and Hudert 2007; Gradwell and Padget 2005).

This concludes the conceptual foundations of my work from both a SLA or service centric view and a processual view regarding the discovery and negotiation mechanisms, regularly employed in distributed IS. Building on these fundamental mechanisms, a set of research project already aims at solving the problem of distributed SLA management and particularly the electronic discovery and negotiation of such documents. The solutions derived within these projects therefore represent alternative solutions to a very similar (or even the same) research problem, to which my approach has to be compared in the end.

2.3. Related Work

In this section, related research work concerning SLA negotiations in distributed IS is presented. Each of the identified infrastructure proposals is described in terms of its overall aims, abstract architecture and, specifically, its SLA discovery and negotiation mechanisms. Finally, every approach is assessed on the basis of the requirements identified in section 2.1; a summary of this assessment can be found in table 2.2.

For clarity purposes, I focus on the ones most closely related to my work (e.g. in terms of similarity of the anticipated scenario to my IoS model) and most prominently

perceived in the research community.

Given the variety of different service management infrastructures present today, this approach is bound to be incomplete. On the other hand it ensures a consistent perspective on related research as a context for this thesis.

Especially several negotiation frameworks originating in the area of software agents have been omitted. This is due to the fact that they either do not at all stress the characteristic aspects of SLA-based service management (e.g. the discovery processes, term restrictions or template-based provisioning), such as (Bartolini, Preist, and Jennings 2005), (Paurobally, Tamma, and Wooldridge 2007), (Jonker, Robu, and Treur 2007) or (Mobach et al. 2005), or only focus on the definition, deployment and parameterization of centralized negotiation systems, such as (Ströbel 2001), (Kim and Segev 2005) or the works of Benyoucef et al. (Benyoucef and Rinderle 2006; Benyoucef and Verrons 2008).

2.3.1. Projects Building on WS-Agreement

AssessGrid

A very prominent project dealing with the usage of SLAs for distributed infrastructures was AssessGrid[22] ("Advanced Risk Assessment & Management for Trustable Grids", funded under the European Commission's (EC) sixth framework programme, contract number 031772). The project started in April 2006 and ended in March 2009.

Its main goal was to address GC's "shortcomings related to security, trustworthiness and dependability" by "develop[ing] and integrat[ing] methods for risk assessment and management in all Grid layers" (Molderez et al. 2006, p. 12). The lacking support for such aspects was identified as a key "obstacle[...] of a wide adoption of Grid [...] technologies in business and society" (quoting the AssessGrid homepage), and therefore addressed primarily.

Building on four different roles (*customer*, *end-user*, *brokers* and *providers* (Padgett et al. 2006)), three "showcase scenarios" were investigated with regard to the discovery and negotiation of SLAs:

- Single-task scenario: the user directly agrees upon a distinct SLA with a provider (the broker is only used optionally to retrieve a set of potential providers along with their risk assessment)
- Workflow scenario with broker as mediator: the user wants to submit a set of jobs for which the broker retrieves potential providers. After this, the user still has to agree upon a SLA with each individual provider.
- Workflow scenario with run time-responsible broker: the user submits a whole workflow to the broker and agrees to one single SLA with it. The broker, on the other hand, choses one provider for each job in the workflow. It is thus providing the user "a higher level of service by selecting the service provider to carry out

[22] http://www.assessgrid.eu

each task and managing the relationship with it" (Parkin, Badia, and Martrat 2008, p. 14).

In order to support a holistic risk assessment within all three scenarios, a series of mechanisms and software components were developed, assisting the user in selecting a needed service based on its trustworthiness. This results in a "vertically integrated solution including all planning, monitoring, and risk management methods for the Grid end-user client, Grid broker, and Grid provider" (Molderez et al. 2006, p. 19).

The consortium regards SLAs as fundamental mechanisms for "all aspects of a business relationship" (Molderez et al. 2006, p. 12), and thus central to their work.

Across all identified scenarios, the standard WS-Agreement negotiation protocol (Andrieux et al. 2007) is employed[23]. Consequently, no possibility of employing different negotiation protocols is given. Also no explicit publication or discovery mechanisms are defined (Parkin, Badia, and Martrat 2008, p. 15). This is even the case in brokered negotiations (here also no assertion is made on how the broker knows the SPs), and thus represents one of the major drawbacks of the system design. Similarly, some of the scenarios demand the broker to take part in the negotiation, although no actually brokered negotiation protocol, such as an auction, is employed. The broker basically acts as an information intermediary or is simply negotiating with consumers first and the providers afterwards. Finally, AssessGrid explicitly aims at human end-users to negotiate with the service brokers or providers, so no automation of this process with software agents is considered.

Note: The Highly Predictable Cluster for Internet-Grids (HPC4U) project[24] also basically represents a reference implementation of the WS-Agreement standard in GC. Hence it exposes the same characteristics as AssessGrid in terms of the requirements used in this thesis and is thus not presented individually here.

Akogrimo

Running from July 2004 to June 2007, the Akogrimo[25] project ("Access to Knowledge through the Grid in a mobile World", EC's sixth framework programme, project reference 004293), developed a Grid architecture for mobile grid services allowing for the implementation of "mobile dynamic virtual organizations"(Wesner et al. 2005, p. 13).

Akogrimo proposed a distributed service infrastructure, building on SLAs as a tool for quality assurance. This was realized as a series of prototype implementation and was validated in two use cases (eLearning and eHealth (d'Andria et al. 2006)).

As with the AssessGrid project Akogrimo more or less directly uses WS-Agreement for the negotiation and representation of SLAs[26] (Terracina et al. 2007, p. 64), so again

[23] Parkin et al. claim in (Parkin, Badia, and Martrat 2008, p. 16) that one additional confirm message (to be sent from the consumer to the provider after having inspected the created SLA) has been introduced. However, I could not find any comment on that in the official project deliverables.

[24] http://www.cit.tu-berlin.de/menue/forschung/hpc4u/

[25] http://www.mobilegrids.org/

[26] Apparently an additionally accept/reject message was introduced after the completion of the standard WS-Agreement protocol (d'Andria et al. 2006) and WSLA terms were used additionally within the

no interchange of negotiation protocols is possible.

A set of registries is present however, such as the *Semantic Service Discovery Service*, for publication and discovery of services and SLAs respectively (Terracina et al. 2007). I could not exhaustively clarify whether or not software agent technology was used for discovery and negotiation of SLAs. This is due to the fact that on the one hand the service discovery and usage is illustrated with a browser-based application and on the other hand user and service agents are mentioned throughout the project deliverables (Olmedo et al. 2007). To my comprehension the actual discovery and negotiation of a service is done via graphical user interfaces, by human users, whereas more low-level tasks like maintaining presence information and communicating with central registries is done by more or less sophisticated agents. Finally, the Akogrimo system also employs (market) brokers, at least assisting the discovery and negotiation processes to some extent.

BEinGRID

A very generic and end user oriented approach on distributed service infrastructures was marked by the BEinGRID[27] project ("Business Experiments in GRID"). It was funded by the EC as the largest integrated project in the sixth framework programme and ran for 42 months until February 2010.

BEinGRID's main focus was to "demonstrate the business benefits from Grid technology" (Parkin, Badia, and Martrat 2008, p. 18). For this, it was heavily involved with end users (the consortium consisted of 95 partners) of Grid technology, focusing on their respective use cases. As a result, the project covered 18 different business experiments along with a set of horizontal activities dealing with areas of interest, common to all experiments, such as the SLA management.

BEinGRID explicitly did not try to design a completely new service infrastructure, but rather deduct best practice mechanisms and concepts from the experiments done. Its deliverables comprised common *technical requirements*, common *capabilities* (groups of needed functionalities), *design patterns* (describing possible implementations of the common capabilities) and only a relatively small set of software components implementing such ideas (Rosenberg and Juan 2009). These components can then be integrated using a "plug&play approach" (Dimitrakos 2009, p. 1) to realize at least parts of a service management infrastructure for a distinct system.

The BEinGRID "SLA Cluster" group was working on a SLA management infrastructure as summarized in (Rosenberg and Juan 2009), which was seen as a crucial component within several of the business experiments. This shows the relevance of reliable services and SLA-based service provisioning from a business view. The assertion is even more convincing, given that the involved end users represented a total of twelve different business sectors, such as finance or architecture (Parkin, Badia, and Martrat 2008).

SLA documents.

[27] Original website is offline; most information on this project can nevertheless be found on http://www.it-tude.com/

Throughout the press releases and otherwise published information documents about BEinGRID, no actual assertion on the usage of software agents as negotiators could be made. From architectural models the existence of a *SLA Negotiator* component is at least mentioned, however its negotiation capabilities, thus whether or not this is just a graphical user interface or an actual negotiator agent acting autonomously to a certain degree, are not described sufficiently. Additionally, the consortium admits to not have defined any publication and discovery mechanisms, although they are considered relevant in future developments (Rosenberg and Juan 2009).

The developed components build on the WS-Agreement specification with a slightly more sophisticated negotiation protocol. The SC requests a quote and after receiving it from the SP can either request a different one or propose the received SLA document to the SP just with WS-Agreement; the SP then accepts or rejects it accordingly (basically this represents an iterated WS-Agreement protocol). Although the protocol is a little more flexible than the traditional WS-Agreement protocol, the chosen approach still prevents any protocol-generity and even non-negotiable SLA terms as claimed by requirement R3 are not possible, as this is not explicitly considered in WS-Agreement up to date.

2.3.2. Approaches Offering Significant Progress Beyond WS-Agreement

BREIN

As one of the first projects, BREIN[28] ("Business objective driven reliable and intelligent Grids for real Business") tried to employ ideas from the software agent and semantic web community for distributed service management. It was funded under the contract number 034556 within the sixth framework programme and ran from September 2006 until August 2009.

BREIN set out to introduce into the grid a "Software as a Service model [...] supporting collaborations among enterprises in a dynamic and flexible way" (Frutos and Kotsiopoulos 2009, p. 39). Building on two real-world business cases ("airport ground handling and virtual engineering design simulations" (Taylor et al. 2009, p. 171)) BREIN explicitly focused on "creating, managing and assessing business-relationships between a provider of services and their customers" (Parkin, Badia, and Martrat 2008, p, 21). For this the consortium employed a fundamental architectural principle in which each business actor is part of a value or supply chain in which it can play both the SC and SP roles, even at the same time (Taylor et al. 2009). A dynamic set of "business virtual organizations" (Taylor et al. 2009, p. 175) arises, in which each actor can only observe and influence a limited area along the overall supply chain.

From this very business-centric view the need for SLA-based service management arises, and is also reflected in the overall architecture, consisting of workflow, SLA and resource management, business relationship and underlying messaging frameworks (BREIN 2010). Herein, a vast number of toolkits and components were developed, implementing the BREIN vision built upon "bipartite agreements", "virtualization" and

[28] http://www.gridsforbusiness.eu/

flexible "orchestration" of business services (Frutos and Kotsiopoulos 2009, p. 40). Especially the semantically enhanced SLA templates and respective negotiation features, stemming from software agent and semantic web concepts, can be seen as a very unique and innovative feature of BREIN (Kotsiopoulos et al. 2008).

Concerning the publication and discovery of the offered services, BREIN follows an approach very similar to the one that is applied in this thesis: a set of service and SLA template descriptions are published at a *service discovery registry*, where the SCs can subsequently query the respective documents and select the most appropriate for their needs[29] (Mora et al. 2009). BREIN does not use only one central registry, but even a P2P-based system of registry nodes, thus following the ideas mentioned in 2.2.2. A very complex SLA structure is employed throughout the project, combining WS-Agreement and WSLA concepts and enhancing them with semantic annotations (Parkin, Badia, and Martrat 2008).

This project represents an important mark in the automation of service management processes, as it explicitly allows the use of software agents for the actual SLA negotiation process (Laria et al. 2009). The overall system is divided into a "head body architecture" (Laria et al. 2009, p. 88), where the head, basically representing the service management layer, is represented by a *Multi-Agent System* and the body is the distributed service system, to be managed by the software agents. The communication between the agents is realized via standard grid and WS technologies (SOAP, WSDL etc.), thus encapsulating the management agents from the actual infrastructure (Laria et al. 2009). These agents negotiate with each other using a newly designed negotiation protocol: the *Combinatorial Contract Net Protocol* as is described in (Laria et al. 2009, pp. 95-100), an extension to the standard FIPA *Contract Net Protocol*, allowing reverse combinatorial multi-attribute auctions. Although this protocol is one of the most sophisticated ones found in the literature, it is still the only one possible, so no protocol-generity, as demanded by requirement R4, was realized.

Note: The Adaptive Services Grid (ASG) project[30] (running from September 2004 through January 2007, funded under the sixth framework programme with the contract number 004617) and in particular the Adaptive Service Agreement and Process Management (ASAPM) sub-project (Chhetri et al. 2007), dealt with a very similar problem. The primary goal of this project was to "develop [a] reference architecture and a prototypical implementation for a semantic services platform for adaptive, semantic service discovery, composition, creation and enactment" (Meyer et al. 2007, p. IV). Assessed on the basis of the requirements identified in 2.1 it can be seen as congruent to BREIN. Both employ a Contract-Net-based negotiation protocol, semantic annotations and agent-based negotiation components. Therefore the ASAPM system is not presented separately here.

[29] This process is actually a little more complicated, as all relevant documents are semantically enhanced and these semantic annotations must consequently be evaluated during the service selection. For this, a set of services, assisting both SCs and SPs, were developed.

[30] http://asg-platform.org/

NextGRID

The NextGRID[31] project ("NextGRID: Architecture for Next Generation Grids") was again funded under the EC's sixth framework programme with the project reference 511563 and ran from August 2004 until August 2007.

Its main goal was to "define the architecture of [...] next generation grids", which are "economically viable", allow to integrate "new and existing business models" and whose security and privacy protecting mechanisms "give confidence to business, consumers and the public" (Snelling and Anjomshoaa 2007, pp. 1-2). In order to realize this vision, the project consortium consisted of various stakeholders including Grid users, hardware and software providers, service infrastructure providers and research organizations (Snelling, Fisher, and Basermann 2005).

The main architectural principles applied for NextGRID were (Snelling and Anjomshoaa 2007):

- "Service Level Agreement Dynamics", where all interactions are done in the context of an agreed SLA.
- Ease of "Service Construction and Consumption" for implementing a dynamic grid infrastructure.
- Design of a "minimal service infrastructure" as a technical basis for "all services operating in [the] NextGRID environment" .

Especially the first principle already shows the importance of SLAs for the NextGRID system. The designed SLA management infrastructure allows the user to discover Grid services and respective SLA template documents from service registries and template repositories. This is done via a specifically designed NextGRID user interface. After having discovered a service fitting the current need, the user (and its associated *Client Negotiator* component) engages in a negotiation with the providers *Negotiator Service*. (Hasselmeyer et al. 2007)

In contrast to most of the other research projects employing SLAs, NextGRID does not build on standard WS-Agreement or WSLA document types, but defined its own SLA schema. It structures a SLA document into *parties*, *dynamic* and (non-negotiable) *static terms* (Mitchell and Mckee 2005). This corresponds directly to requirement R3. During the actual negotiation process the *Discrete Offers* protocol is employed, a very simple offer-accept process, during which the consumer requests an offer from the provider and subsequently accepts or rejects it (Hasselmeyer et al. 2007). It is not possible to change the applied protocol or even migrate from one protocol to the other during run time. Finally, although some of the contributors investigate the usage of negotiation brokers for both SP and SC in (Hasselmeyer et al. 2006a), the developed system does not employ software agent technology as demanded by requirement R6, but rather focuses on human users.

[31] http://www.nextgrid.org/

Note: The TrustCOM project (running from February 2004 through January 2007) resulted in a "framework for trust, security, and contract management for secure, collaborative business processing and resource sharing in dynamically-evolving virtual organizations" (Wilson 2007, p. 4). Concerning the requirements used in this thesis this project is basically equivalent to NextGRID, so it is again not considered in all detail here.

2.3.3. Projects Focusing on Economic Aspects

CATNETS

As opposed to the aforementioned projects, CATNETS ("Catallaxy paradigm for decentralized operation of dynamic application networks")[32] did not focus on the management of SLA documents per se, but rather on the application of economic "market-based resource allocation mechanisms in application layer networks such as grids" (Veit et al. 2007, p. 3). It was again funded under the EC's sixth framework programme under the contract number 003769 and lasted from September 2004 until August 2007.

The main goal of CATNETS can be summarized as the assessment of market-based resource allocation mechanisms in massively distributed computing infrastructures. For this investigation a two-layered market setting was assumed, consisting of a higher-level *Service Market* and lower-level *Resource Market*. The idea behind this vision was that *Basic Services* or respective Agents are constructed on the Resource Market by integrating a distinct set of raw electronic *Resources*, such as storage or computation. These Basic Services are subsequently combined on the Service Market to more complex services or workflows (Schnizler et al. 2005b). Within this scenario setting, two competing approaches for economic resource management were compared: a central, auction-based and a decentralized Catallaxy-based (Ardaiz et al. 2006) mechanism.

During the project both a simulation environment for assessing these two approaches as well as a proof-of-concept prototype (Ardaiz et al. 2007) (however only incorporating the decentralized mechanism) was developed (Schnizler et al. 2005b). Additionally, a lot of theoretical research in the area of resource allocation mechanisms, market design and evaluation of such concepts was done.

Although the agents within the CATNETS system negotiate in a simplified proprietary negotiation language the project consortium could show how this language can be mapped to SLA languages such as WS-Agreement (Schnizler et al. 2005b). Since the primary focus was on the economic mechanisms this decision is very comprehensible. On the other hand the CATNETS consortium very thoroughly investigated how an economic middleware as the CATNETS prototype could be integrated with existing middleware systems, subsequently acting as a sophisticated resource allocation tool. This effort primarily built on WS-Agreement, thus reflecting the importance of SLA-based service provisioning in CATNETS (Joita et al. 2007).

Regarding the identified requirements it can be stated that this project actually integrates software agents for discovery and negotiation of the traded services. This is the

[32] http://www.catnets.uni-bayreuth.de/

case on both markets. Regarding the applied protocol one must distinguish between the centralized and the decentralized mechanism. The former applied a CDA on the Service Market (single-attributive goods are traded here) and a multi-attributive combinatorial exchange on the Resource Market (Schnizler et al. 2005a). The latter always employs the *Alternate Offers Protocol (AO)*, a one-on-one negotiation approach (Veit et al. 2007). Similarly, the discovery process also differs in the two market types: In the centralized approach no actual discovery is mentioned. I assume the agents are parameterized with the location of the broker, as this setting was only realized in a simulation environment. The decentralized setting on the other hand employs a broadcast discovery protocol (Veit et al. 2007).

Note: In terms of the identified requirements the infrastructure proposal described in (Ouelhadj et al. 2005) is quite similar to the CATNETS system; the only difference is, that the authors do not mention an actually implemented prototype of their system.

SORMA

SORMA ("Self-Organizing ICT Resource Management")[33], being a follow-up project of CATNETS, further developed the idea of market-based resource allocation in distributed infrastructures. It was funded under the sixth framework program, contract number 034286, and lasted from August 2006 until July 2009.

According to one of the first project deliverables, SORMA aimed at "the development of methods and tools for establishing an efficient market-based allocation for resources in a more efficient way in order to enable resource accessibility for all users and to increase user's satisfaction, profit and productivity" (Matros et al. 2008, p. 8). For this, the project consortium built on software agents to enable a fully automated and economically sound service trading process (Matros et al. 2008).

The resulting architecture consists of six different layers (SORMA 2007):

- *Grid Application Layer*: applications to be run on the resources, purchased over the SORMA market.
- *Intelligent Tool Layer*: smaller software components supporting the SCs and SPs in the trading process (for example the *Bid Generator*).
- *Open Grid Market Layer*: mainly economic services, responsible for offer matching (Trading Management) and SLA handling (*Contract Management*, *SLA Enforcement* and *Billing*) etc.
- *Core Market Services*: extension services, needed to allow market-based resource allocation in traditional Grid middlewares (for example transaction logging, market directory etc.).
- *Economically Enhanced Virtualization Middleware*: resource-centric interface to standardized Grid implementations, allowing for resource co-allocation or SLA management on the basis of an individual resource.

[33] http://www.sorma-project.org/

- *Grid Resource Layer*: set of services to be traded over the SORMA market.

This logical architecture was subsequently implemented in a proof-of-concept prototype employing a CDA-based trading system (SORMA 2009).

Given its economic setting, SORMA explicitly builds on SLAs as a fundamental concept, both defining the relationship between SC and SP as a result of the matching process, but also as an input for the SLA enforcement and billing system (SORMA 2008). This component subsequently "perform[s] a continuous run time SLA violation prediction, detection, and reaction that can help prevent SLA violation from occurring" (SORMA 2008, p. 58).

Following the ideas already proposed during CATNETS, SORMA aims at an economically sound resource-allocation based on markets. Although the need for different negotiation protocols is explicitly stated in (SORMA 2007, p. 9), only a single protocol (a CDA) was actually implemented in the final prototype system (SORMA 2009, p. 57). The mentioned possibility of integrating various negotiation protocols via the Market Exchange Service was apparently not or not sufficiently implemented within the project (SORMA 2009, p. 50).

On the other hand, SORMA builds on a distributed set of market registries, assisting the SC and SP agents during the discovery and publication processes respectively (SORMA 2008). Originating in the economic usage scenario underlying SORMA, an extended SLA description language was created allowing for the definition of technical and also economic aspects of a service invocation (Borissov et al. 2009). Finally, software agents are heavily used during both the discovery and the negotiation of SLAs in SORMA (Borissov, Neumann, and Weinhardt 2009).

2.3.4. Initial Approaches Towards Protocol-Generity in SLA Negotiations

NegoFAST

Resinas et al. (Resinas, Fernandez, and Corchuelo 2010) very recently defined a comprehensive framework for the design and implementation of generic SLA negotiation components, the *NegoFAST* framework. For this the authors conducted a comprehensive survey on electronic negotiation systems and subsequently deducted a set of requirements, similar to those presented in this thesis. Based on these requirements, a reference architecture for generic service negotiators was designed, followed by a proof-of-concept implementation.

The authors claim that their main goal is "to understand the requirements of automated negotiation systems of service agreements in open environments and to provide the foundations for developing such systems" (Resinas, Fernandez, and Corchuelo 2010, p. 2).

The resulting reference architecture comprises four different modules: *Protocol Management* ("selection and execution of negotiation protocols" (Resinas, Fernandez, and Corchuelo 2010, p. 13)), *Decision Making* ("determin[ing] the behavior of the [...] system] (Resinas, Fernandez, and Corchuelo 2010, p. 14)), *World Modeling* (used for "gather[ing], analys[ing] and manag[ing] useful information to make decisions during a negotiation"

(Resinas, Fernandez, and Corchuelo 2010, p. 13)) and *Coordination* (orchestrating the other modules during the negotiation process).

These modules subsequently have to be instantiated when implementing a given negotiation system for a particular setting. By rigorously building on common interfaces and a generic data model[34] the NegoFAST architecture allows for implementing various decision making strategies and negotiation protocols, from which system designers can chose during this instantiation process. The authors even claim their system allows changing the applied negotiation protocol or decision making component at run time (Resinas, Fernandez, and Corchuelo 2010).

It is clearly observable that the NegoFAST framework particularly focuses on the negotiation phase of the service life cycle. No assertions on service discovery and the application of software agents therein are made. On the other hand a variety of different SLA representation schemes is supported, basically all SLA languages that can be mapped to the defined generic data model. Similarly, all negotiation protocols that can be built using the defined negotiation primitives are eligible in the framework. The authors mention in (Resinas, Fernandez, and Corchuelo 2010) that the negotiation protocol actually applied is again negotiated beforehand, however no assertion on how this is done (or most importantly how the respective protocol is described / communicated to the other party) could be found.

Aneka and GridBus

Another work, very similar to my approach was published by Brandic et al. (2008a). It was done in the context of the Aneka[35] and GridBus[36] projects of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne.

The authors state the problem that most SLA negotiation infrastructures demand a pre-defined configuration of the participants in terms of applied protocol or negotiated SLA terms and propose a solution based on a comprehensive *meta-negotiation architecture*.

This meta-negotiation system builds on a set of meta-negotiation documents, "where participating parties may express: the pre-requisites to be satisfied for a negotiation [...], the negotiation protocols and document languages for the specification of SLAs that they support and conditions for the establishment of an agreement" (Kertesz, Kecskemeti, and Brandic 2009, p. 29).

Brandic et al. not only defined the overall process for meta-negotiation but also presented a XML-based language description to be used when constructing respective documents, as well as a role-based architecture for their system. It comprises a set of registries, where the documents are published as well as some middleware components on both SC and SP sides used to parse these documents and load the respective strategy

[34] Within this data model a set of negotiation primitives (for example *accept*, *rejectNegotiation*, *rejectProposal*, *commit*, *propose* or *inform*) is used as a common basis for all supported negotiation protocols (Resinas, Fernandez, and Corchuelo 2006).

[35] http://www.manjrasoft.com/products.html

[36] http://www.gridbus.org/

modules necessary for the negotiation. A proof-of-concept prototype based on the Aneka GC system and the GridBus resource broker was implemented as an extension to the infrastructure presented in (Venugopal, Chu, and Buyya 2008).

The Aneka meta-negotiation system builds on a registry-based discovery mechanism and allows a distinct set of different negotiation protocols to be used subsequently. However, these protocols are simply referenced by their name, so a common understanding of the respective protocol is necessary for successful execution. In my thesis, I try to use a more generic approach, in which only some defined protocol primitives have to be known (similar to the NegoFAST model) and more complex protocols can be constructed from these. Brandic et al. also do not mention the usage of software agent technology or the achieved degree of automation within their work. Also, no market intermediaries are introduced, as needed for example in a brokered market structure.

SLA@SOI

The most recent project dealing with electronic SLA management in a business context is SLA@SOI [37]("Empowering the service industry with SLA-aware infrastructures"). It is funded under the EC's seventh framework programme under the contract number FP7-216556, started on 31st of May 2009 and will run through 38 months. The consortium consists of eleven partners from both industry and academia, the former covering many significant fields involved in the IoS vision: telecommunications, software development, business services etc.

According to one of their initial project deliverables, the main goal of SLA@SOI is to design and implement "a business-ready service-oriented infrastructure empowering the service economy in a flexible and dependable way" (Theilmann et al. 2009, p. 7). In that, they develop a framework architecture, supporting the management of both SLAs and the underlying service infrastructure on either the *business*, *software* and *infrastructure* level (Theilmann et al. 2010). It comprises software components supporting the overall service life cycle (see (Theilmann et al. 2010, p. 19) for details), such as *SLA Managers* responsible for managing the available SLA templates and negotiation processes building thereupon, *Service Evaluators* handling prediction tasks or *Service Managers* for run time management of the service instances.

According to (Kotsokalis et al. 2010) and (Lambea et al. 2010), a rule-based protocol engine is used within each SLA Manager node, allowing for protocol-generic negotiations. For this, it is configured with a respective description file, to which all involved negotiators agree and subsequently enforces the protocol rules by intercepting all message exchanges (one SLA Manager is located at both the SP and the SC realm).

The fundamental concepts, applied in SLA@SOI, are very similar to my approach: a generic negotiation component is employed, and is configured with a protocol description document as per negotiation it engages in, a registry-based discovery phase is assumed and the negotiation protocols themselves are described, at least partially, on the basis of state machines. This fact underlines not only the motivation for my work, but also the general design idea applied therein (see 3.1 for more details).

[37] http://sla-at-soi.eu/

However, most of the concepts presented in the currently available project deliverables are in a very preliminary stage and some claims are even still contradictory at this point in time. The consortium for example states, that the current prototype supports all "single negotiations (one-to-one, one-to-many and many-to-one)" (Lambea et al. 2009, p. 51) but state in the very same document that "at the moment, the protocol [they] decided to use in SLA@SOI is a one-to-one protocol (multi-attribute bilateral protocol)" (Lambea et al. 2009, p. 53). At the date of publication of this thesis, the actual degree to which amount the SLA@SOI system fulfills the stated requirements could thus not be determined. Therefore it is not integrated into table 2.2 and only mentioned here shortly. When working on distributed SLA management in the future, the results achieved therein can nonetheless be foreseen to be highly relevant and are recommended for further investigation.

Table 2.2.: Related Work

	Related Projects	AssessGrid / HPC4U	Akogrimo	BEinGRID	BREIN / ASG - ASAPM	NextGRID / TrustCom	CATNETS	SORMA	NegoFAST	Aneka / GridBus
Requirements										
Registry-based Discovery Phase [R1]		?, no assertion	Yes	No	Yes	Yes	No	Yes	?, no assertion	Yes
Service Management based on SLAs of varying Complexity [R2]		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Restrictable SLA Offers incl. non-negotiable Terms [R3]		No	No	No	No	Yes	No	No	?, no assertion	Partly, usage of standard SLA languages with a possibility to define the negotiable terms beforehand
Protocol-generic SLA negotiations [R4]		No	No	No	No	No	No	No	Yes	Yes
Minimal Restrictions on the used Set of Protocols [R5]		No	No	No	No	No	No	No	No	No
Automated, agent-based service management [R6]		No	?, no assertion	?, no assertion	Yes	No	Yes	Yes	?, no assertion	?, no assertion
Market Intermediaries [R7]		Partly (service broker available, although no actually brokered negotiation protocol is employed)	Partly (QoS / Virtual Organization brokers available, although no actually brokered negotiation protocol is employed)	No	Partly (intermediaries are not directly involved in the negotiation)	Partly (no actually brokered negotiation protocol is employed)	Partly (broker available in the centralized case)	Yes	No	No

The requirements derived in section 2.1 allow for a structured assessment and comparison of the identified related research projects. As can be seen in table 2.2 none of these projects is currently capable of fulfilling all of the stated requirements. The BabelNEG system, the design of which is presented in the next chapter, aims at closing that gap. A detailed investigation of its capability to fulfill all posed requirements (and thus indirectly its comparison with the projects presented above) is given in section 4.2.

After having presented either the conceptual and technical foundations and alternative approaches to the stated research problem along with a conceptual framework for their assessment (in terms of the stated requirements) the next two chapters focus on the actual design and evaluation of the BabelNEG system; my proposed solution to the stated research problem.

3. Design and Development

In this chapter, the central deliverable of this dissertation project, the infrastructure design, is presented. Due to its complexity it is further structured according to three well-recognized perspectives on IT-artifacts, which, when used in combination, result in a comprehensive view on the investigated system: *data*, *interaction* and *functional* perspective (Ferstl and Sinz 2008, p. 137)[1]. This trichotomy also corresponds to the three already introduced perspectives on (electronic) negotiations: negotiation object (data perspective, focusing on the documents involved in the negotiation), negotiation protocol (interaction perspective, describing the negotiation processes) and decision making strategy (functional perspective, dealing with the processing of messages and events).

Example instantiations of the developed data structures and agent roles as well as potential start-up and usage parameters are presented along with the demonstration and evaluation steps in chapter 4.

3.1. Abstract Design Idea

The basic design idea, underlying this work, is to offer a given good (SLA for an electronic service in the IoS setting) independently from the way an agreement concerning this product can be attained (negotiation protocol).

Such an approach has many analogies in real-world settings. Many products sold in everyday life are sold with different negotiation protocols in different situations. For example a TV set, displayed at an electronic retailer, implicitly states that the only way to negotiate about it is to accept the stated price. This protocol thus corresponds to a classic catalogue pricing model, also called *Fixed Price Auction* (FPA)[2]. Then again, the same TV set, offered over an online auction platform such as eBay[3], implies that the consumers have to outbid each other until a certain deadline occurs. This corresponds to an EA protocol. eBay could even act as an example for both of these protocols, since apart from the standard auctions also *buy it now*-offers are available, which are equivalent to FPAs. Although the product sold in both cases is exactly the same (a new TV set of a given type), the negotiation protocols applied are quite different.

[1] Regularly a fourth, dynamic view concerning the actual processes is used. In this thesis this view is integrated with the interaction perspective, allowing for the conjoint description of communication channels, i. e. messages, and the actual processes building on them.

[2] A FPA is basically a very simple auction protocol in which the price is fixed and each bidder can only accept the stated value or leave the auction. The first bidder sending the accept message wins the negotiation, given only one unit of the sold product is available (otherwise the first n bidders receive one item each, given n units are available).

[3] http://www.ebay.com/

The protocol to be used within a given negotiation situation is regularly chosen by the provider, based on factors like the market context, distribution of providers and consumers in the particular market or characteristics of the negotiated good itself (such as expiration dates etc.).

This decoupling of protocol and object allows for a protocol-generity in real-world negotiation settings. Transferred into the IoS scenario, it implies that a given SLA (template) under negotiation could be offered with a whole set of different negotiation protocols at different points in time or on different markets in parallel. This would ultimately lead to service systems in which the SPs could choose a negotiation protocol for a distinct SLA negotiation, best fitting the current setting (for example in terms of market configuration), or even swap the applied protocol due to changes in this setting respectively[4].

The SCs would be able to search for appropriate services in a more detailed way, as they will be given the opportunity to use the offered protocols as search criteria for a given service request. Just as well, they could only focus on the service type without caring about the associated protocol. They would just have to make sure they can adapt to it afterwards. This can again be observed at the eBay platform, where users can search for a specific good either with a specific restriction on the applied protocol (auction or buy-it-now offer) or without one (default case).

In a fully automated scenario, such as the IoS, the negotiation protocol applied for a given service should not only be decoupled from the actual SLA but must also be made explicit in terms of its communication rules. The software agents must be able to adapt to it, solely based on the so defined protocol description.

An analogue mechanism is also used in the real world. In some examples, such as just accepting the price at a traditional electronic retailer or the possibility to bargain on the payment conditions for example at a used-car dealer, the protocol used is described implicitly[5]. Others, such as the auction protocol used on EBay, is very clearly defined in terms of its bidding rules[6].

The designed infrastructure follows this approach and consequently builds on a conceptual architecture of machine-readable (service) description documents providing the management agents with exactly the information needed for the protocol adaption process.

In the remainder of this section these service description documents are presented first (data perspective), followed by the protocol components for both the discovery and negotiation phase, representing the interaction perspective. Finally, the functional architecture of the overall system is sketched in terms of the internal routines of the

[4] Allowing SPs to offer a given SLA with different protocols in parallel and on the same market is very demanding with regard to race conditions in the admission phase, but at the same time it doesn't create a valid surplus with regard to the stated research question. Offering a service with different protocols in parallel is thus assumed out of scope for the proof-of-concept prototype presented in this thesis.

[5] In unsupported negotiation settings solely involving humans, implicit protocol rules are sufficient as the involved negotiators can talk to each other to resolve any misunderstandings.

[6] See http://pages.ebay.com/help/buy/bidding-overview.html for a detailed description of the eBay auction mechanism.

developed management agents (functional perspective)[7].

3.2. Service Description Documents

The set of service description documents to be developed for the prototype infrastructure needs to allow a) the discovery of an appropriate service or respective SLA (template) for this service and b) the exhaustive description of the negotiation protocol used to reach an agreement as needed for an automated adaption. Also, a high reusability is desirable to minimize redundancies.

To this end three different data structures have been designed:

- *Service Type (ST)*: definition of the functional and non-functional aspects, a given class of services can offer.
- *Extended SLA Template (EST)*: definition of the QoS guarantees (building on the non-functional aspects given in the respective ST) as well as the negotiation protocol offered by a given class of services.
- *Service Identificator (SI)*: identification of an individual service instance along with links to the respective ST and EST documents associated with it.

Figure 3.1 gives a short overview on these documents and their relations.

The fourth document type, needed for a SLA-centric service management, is the agreed-upon SLA itself, which is shortly sketched at the end of this subsection along with the way it can be created building on the three other document types and the SLOs defined throughout the negotiation process.

All four data structures have been defined on the basis of XML (Bray et al. 2006) and XML Schema (Thompson et al. 2004), respectively. This technology deemed appropriate as its distinction between attributes and elements represents an intuitive analogy to the parameter-based description of service attributes and (negotiation) protocols. It also serves as a lingua franca in the envisioned service-oriented scenarios, representing the de facto standard for communication therein[8].

3.2.1. Service Type

The basic goal of the ST document is to describe a class of service instances in terms of their functional capabilities and the non-functional parameters, for example QoS metrics like throughput or business relevant aspects like price. It therefore follows the ideas for functional descriptions of electronic services as were present ever since the first introduction of SO (see for example Bellwood et al. 2004).

[7] Due to the developed system being a proof-of-concept prototype, error handling apart from semantic errors related to the discovery and negotiation itself has largely been excluded from this work.

[8] For clarity reasons only excerpts of the actual XML Schema documents are presented when needed for the description of the respective data structures. The complete documents can be found in appendix A.1.

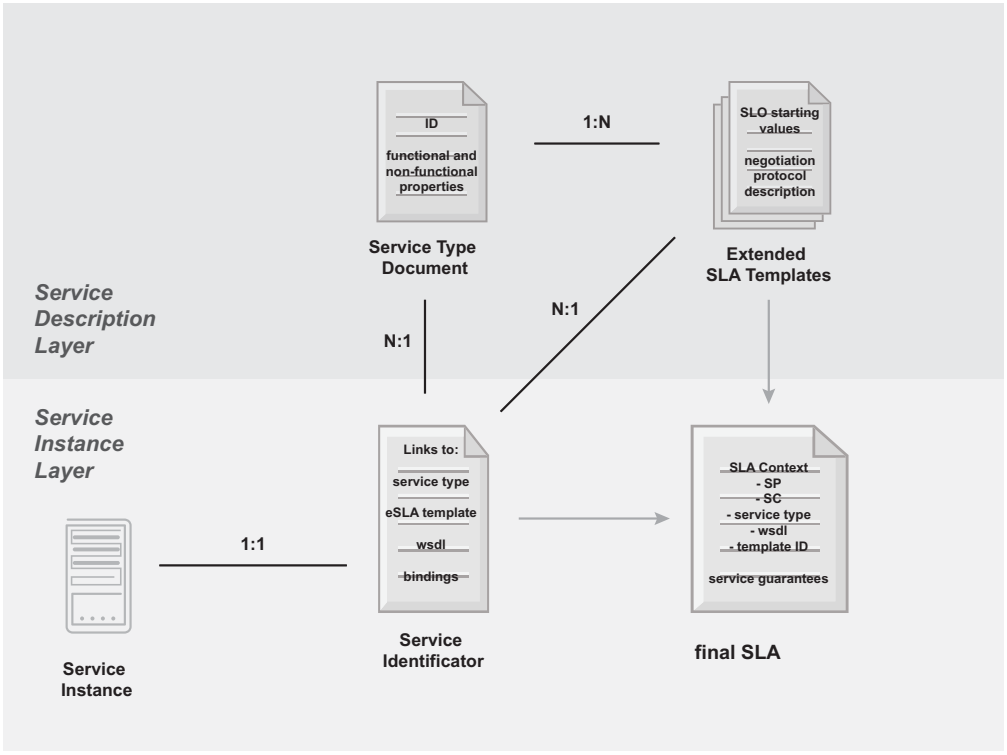


Figure 3.1.: Document-based Architecture

The functional aspects are primarily used to search for a service, taking into account the search criteria, received from the user / service requestor. For the subsequent negotiation process, these parameters play almost no role at all as regularly the functions offered by a service instance are hard-coded and cannot be altered at run time.

In contrast, the non-functional parameters or rather their actual values are mostly not hard-coded per se and are the very object of any SLA negotiation. Only the mere existence of such a parameter (in terms of its parameter id and value domain) has to be decided upon at design time (to this end, respective internal mechanisms allowing for run time adjustments have to be implemented), the parameter's actual value however varies during run time depending on, for example, the current workload.

An example for this would be the non-functional parameter throughput. At design time such a parameter has to be considered, in terms of its id, value range and implemented internal routines, capable of measuring the current throughput and predicting its value depending on current workloads. The incorporation of respective monitoring and enforcement components offers the management agent the possibility to negotiate about this metric and subsequently enforce the respective guarantee, as stated in the SLA, even at run time.

The ST document contains the following elements (see figure 3.2):

- *serviceTypeID*: ID for this particular ST (needed for references in other service

```
<xsd:complexType name="PropertiesType">
  <xsd:sequence>
    <xsd:element name="domain" type="xsd:String" />
    <xsd:element name="declaration" type="xsd:String"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="propertyID" type="xsd:anyURI" />
</xsd:complexType>

<xsd:element name="ServiceType">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="serviceTypeID" type="xsd:anyURI" />
      <xsd:element name="serviceDescription"
        type="xsd:anyType" />
      <xsd:element name="property" type="PropertiesType"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure 3.2.: Excerpt of the ST Schema Definition

description documents). This element should have the form of a Uniform Resource Identifier (URI) in order to ensure uniqueness in a global setting.

- *serviceDescription*: definition of the functional aspects of this ST. This could be realized in the form of a link to a comprehensive UDDI description or even an in-line description with a proprietary description language[9].
- *List[property]*: list of non-functional properties offered by services of this type. Each property element consists of a unique name (URI format), a domain such as Integer or Double and an optional textual description[10]. In order to achieve interoperability and, at the same time, semantic integrity such properties should be defined in an industry-wide ontology.

3.2.2. Extended SLA Template

Most of the present SLA (management) mechanisms for distributed systems incorporate the possibility to define SLA templates (see for example Andrieux et al. 2007). A SLA template is basically a partially filled out SLA document, that is offered from the SP to potential consumers.

A mechanism like this allows the SP to pronounce its willingness to negotiate a SLA for a given service and gives her the opportunity to define an initial proposal on some of the SLOs she is willing to offer. This can tremendously increase the speed of convergence

[9] These service descriptions could also include semantic annotations for individual functionalities as proposed by for example in (Overhage and Thomas 2005).

[10] It should be noted, that two different types of domains are present: *ordered* and *not-ordered* domains. The former are value domains, such as Integer or Double, in which the individual values can be compared using relational operators, such as $\leq$ or $\geq$. The latter represent domains, in which the individual values cannot be compared in such a manner, for example String.

```
<xsd:complexType name="NegotiationProtocolType"
  mixed="true">
  <xsd:sequence>
    <xsd:element name="context" type="ContextType" />
    <xsd:element name="negotiationObject"
      type="NegotiationObjectType" />
    <xsd:element name="offerRestrictions"
      type="OfferRestrictionsType"/>
    <xsd:element name="offerAllocationPolicy"
      type="OfferAllocationPolicyType"/>
    <xsd:element name="informationPolicy"
      type="InformationPolicyType"/>
    <xsd:element name="process" type="ProcessType" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SLOType">
  <xsd:sequence>
    <xsd:element name="value" type="xsd:anySimpleType" />
  </xsd:sequence>
  <xsd:attribute name="propertyID" type="xsd:anyURI" />
</xsd:complexType>
```

(a)

```
<xsd:complexType name="SLATemplateType">
  <xsd:sequence>
    <xsd:element name="SLO" type="SLOType"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="ExtendedSLATemplate">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="slaTemplateID"
        type="xsd:anyURI" />
      <xsd:element name="slaTemplate"
        type="SLATemplateType" />
      <xsd:element name="negotiationProtocol"
        type="NegotiationProtocolType" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

(b)

Figure 3.3.: Excerpt of the EST Schema Definition

in a negotiation process as the SC already gets an idea of what the SP potentially finds acceptable and can thus a) identify SPs more quickly with whom she will potentially never reach an agreement and b) if a compatible template was found, create counter offers which are more likely to be accepted.

The internal structure of a SLA template is basically equivalent to an actual SLA (for a more detailed description of such a document see 3.2.4). The only difference is, that a set of rules is optionally defined, helping the SC to identify (combinations of) SLOs that are a) valid and / or b) acceptable by the SP. The WS-Agreement standard is a good example for this: In this specification the template documents are structurally equivalent to the actual SLA documents (consisting of a context, SDTs, SRTs, SPTs and GTs) and additionally a set of *Creation Constraints* is given, defining how new SLOs should be created or already stated ones could be altered during negotiation (Andrieux et al. 2007, pp. 30-33).

For the BabelNEG system the SLA template concept has been extended to also include a description of the applied negotiation protocol. Since this protocol description is very closely linked to the SLOs already stated in the template it was considered appropriate to integrate it into the same data structure. The other document closely linked to the protocol description is the ST, as it defines the metrics that can be used within the negotiated SLOs. This document has been modeled independently in order to increase reusability as, even in infrastructures without SLA (negotiation) mechanisms, such STs are employed for simple discovery and binding purposes. The SLA template and the negotiation protocol on the other hand will not be used independently, as a template cannot be employed without negotiating it subsequently and a negotiation protocol always has to be defined upon a negotiation object, i.e. a SLA template.

An EST consequently consists of the following elements (see figure 3.3):

- *slaTemplateID*: ID for this particular EST.

```
<xsd:simpleType name="AdmissionRestrictionFormType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="open"/>
    <xsd:enumeration value="restricted"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="AdmissionType">
  <xsd:sequence>
    <xsd:element name="admissionRestrictionRule"
      type="xsd:anyType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="admissionRestrictionForm"
    type="AdmissionRestrictionFormType"/>
</xsd:complexType>
```

(a)

```
<xsd:complexType name="RoleContextType" mixed="true">
  <xsd:sequence>
    <xsd:element name="maximumNumberOfAgents"
      type="xsd:integer" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="minimumNumberOfAgents"
      type="xsd:integer" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="admissionRestriction"
      type="AdmissionType"/>
  </xsd:sequence>
</xsd:complexType>
```

(b)

Figure 3.4.: Type Declaration: Role Element

- *slaTemplate*: potentially empty list of initial SLOs, i.e. guarantee assertions consisting of the name of the metric and the guaranteed value [11].

- *negotiationProtocol*: a structured description of the applied negotiation protocol[12].

Since this protocol description marks one of the core deliverables of this thesis, it is described in more detail in the following.

Where possible, it builds on simple parameters. If further restrictions concerning the negotiation process are needed, any desired (external) rule language, such as Jess[13], may be used within some of the language elements. Moreover, the developed language incorporates state automata concepts when describing the actual protocol steps occurring in a negotiation.

For clarity reasons the set of negotiation parameters is further subcategorized into six different perspectives: *negotiation context* and *object*, *offer restrictions*, *allocation* and *information policy* and *negotiation process*. This structure roughly follows other negotiation description approaches, such as (Ströbel and Weinhardt 2003) or (Lomuscio, Wooldridge, and Jennings 2003).

Negotiation Context The negotiation context defines the involved agents and their permissions and obligations. For this purpose, two *role* elements are specified for the subsequent negotiation, one for both SP and SC.

Each of these elements exposes up to three child elements: *minimumNumberOfAgents*, *maximumNumberOfAgents* (both optional) and *admissionRestriction*.

[11] The respective parameters and value domains have to be defined in the ST this template relates to.

[12] This description language does not cover multi-unit negotiations, which deal with the exchange of multiple units of the same product within one negotiation. Since each service in the IoS can and has to be unambiguously identified, each service, even if offering the same functionality as other instances, represents an individual good under negotiation, which has to be treated in a negotiation of its own.

[13] http://herzberg.ca.sandia.gov/jess/

```
<xsd:simpleType name="ValuesType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="single"/>
    <xsd:enumeration value="multiple"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="NegotiableSLOType">
  <xsd:sequence>
    <xsd:element name="values" type="ValuesType" />
  </xsd:sequence>
  <xsd:attribute name="propertyID" type="xsd:anyURI"/>
</xsd:complexType>

<xsd:complexType name="NegotiationObjectType">
  <xsd:sequence>
    <xsd:element name="negotiableSLO"
      type="NegotiableSLOType" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

Figure 3.5.: Type Declaration: Negotiation Object Element

The optional *maximumNumberOfAgents* and *minimumNumberOfAgents* elements define the maximum and minimum amount of permitted participating agents for that particular role. The description of a one-on-one bargaining protocol for example, would thus state a minimum and maximum number of agents for both roles to be 1.

On the other hand an auction protocol, would restrict one side (SPs or SCs) to a maximum number of one and the other side would possibly not expose any upper bound of permitted agents.

Some negotiation settings require the joining agents to satisfy some criteria to be admitted. Such admission restrictions can be specified within the *admissionRestriction* element. It consists of an optional element of xsd:anyType (*admissionRestrictionRule*), representing a placeholder for arbitrary admission restriction rules expressed in some external rule language, and one attribute defining whether admission restrictions do exist at all (*admissionRestrictionForm*). If *open* admission is defined here, no admission restriction rule is specified (the respective element would be omitted). In the *restricted* case an admission rule would have to be present.

Negotiation Object After specifying the involved roles and their permissions, the actual negotiation object has to be defined in terms of the SLOs under negotiation (see figure 3.5).

Following the common understanding of electronic SLA documents and their internal structure, this element consists of a (potentially empty) list of *negotiableSLO* elements[14]. Each of these refers to a respective service property as defined in the ST; this link is created via the property's ID. Additionally, a child element is present, defining whether multiple values are allowed to be offered for this SLO or not. Intuitively not all possible

[14]An empty list would indicate, that no SLO is negotiable, thus defining a FPA protocol.

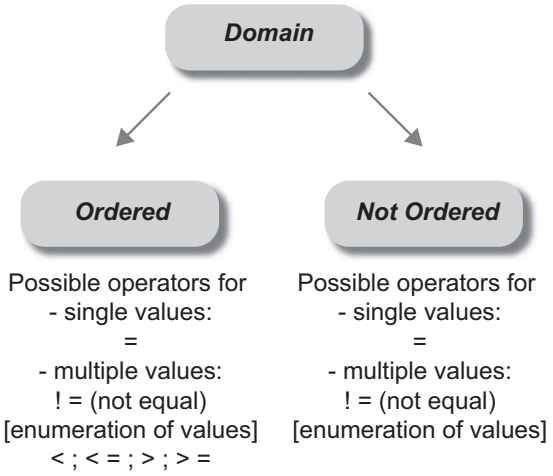


Figure 3.6.: Ordered vs. Not Ordered Domains

relational operators used to express multiple values are applicable in every type of domain (as defined for the respective SLO within the ST). Not-ordered domains only allow $\neq$ or simple enumerations to express multiple possible values. In ordered domains on the other hand, relations like e.g. $<$, $\leq$, $>$ and $\geq$ are possible.

Offer Restrictions Each negotiation description may optionally define an arbitrary number of *attributeRestriction* and / or *generalRestriction* elements. Each of these represents a specific restriction on how one particular SLO is to be treated within the negotiation. If, for example, a negotiation designer wants to specify that for some particular attribute a new offer always has to succeed the last offer, like in an EA, this would be achieved with an attributeRestriction regarding this SLO.

Each *attributeRestriction* element thus contains a *propertyID* attribute referencing the SLO it applies to. Additionally, one of three possible restriction classes (modeled as child elements) can be chosen:

- The *progress* element defines the direction of a negotiation regarding the referenced attribute. With this element one can specify whether new offers have to state higher or lower values for this SLO, compared to the previous offers. Intuitively the *progress*-element is only applicable for ordered domains. In addition to the direction, specified in the *progressForm* child element, some minimum increment or decrement may be defined with the *delta* child element.

- The *threshold* element defines what is called a reserve value in negotiation theory. A reserve value is one negotiator's upper or lower bound of acceptable values for a particular negotiation attribute. Normally, this construct is only used for the price, but in this more general approach it can also be applied to all other service attributes of ordered domains. A negotiation designer could for example specify

```
<xsd:simpleType name="ProgressFormType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="ascending"/>
    <xsd:enumeration value="descending"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ProgressType" mixed="true">
  <xsd:sequence>
    <xsd:element name="progressForm"
      type="ProgressFormType"/>
    <xsd:element name="delta" type="xsd:anySimpleType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ThresholdType">
  <xsd:choice>
    <xsd:element name="lowerBound"
      type="xsd:anySimpleType"/>
    <xsd:element name="upperBound"
      type="xsd:anySimpleType"/>
  </xsd:choice>
</xsd:complexType>
```

(a)

```
<xsd:complexType name="AttributeRestrictionType"
  mixed="true">
  <xsd:choice>
    <xsd:element name="progress"
      type="ProgressType"/>
    <xsd:element name="threshold"
      type="ThresholdType"/>
    <xsd:element name="restrictionRule"
      type="xsd:anyType"/>
  </xsd:choice>
<xsd:attribute name="propertyID" type="xsd:anyURI" />
</xsd:complexType>
```

(b)

Figure 3.7.: Type Declaration: Attribute Restriction Element

that in either case at least 2 GB of memory have to be available for some service; any value lower than that would not be acceptable.

- Finally, a *restrictionRule* element of xsd:anyType can be used to express any attribute-related constraint in addition to *thresholds* and *progress* restrictions. In this element external rule languages may be employed.

Similarly to such free-hand restriction rules relating to exactly one SLO each, more general restrictions relating to more than one service attribute can be expressed with the *generalRestriction* element. This could be used for example, if a negotiation designer wants to specify that in a new offer at least for one of two different attributes has to be offered a higher value than in the current offer.

Offer Allocation Policy The *offerAllocation* element defines the way the clearing of the negotiation is conducted, that is how the winning offers are identified and transformed into a valid SLA.

Offer matching can be of *forwarded* or *defined* form. In either case a valid agreement is created by one side receiving an offer and accepting it [15]. The only difference lies in whether the other participants know how the offer allocation is conducted or not. In the forwarded case the matching algorithm is not explicitly described. The accepting agent chooses one offer to win without letting the other participants know according to which rules. In the defined case these rules are given within the protocol description and are therefore openly available. This way every agent can predict the winning offer before

[15] Depending on the protocol only one side could be allowed to actively accept an offer. This is defined in the *process* element of the protocol description.

the actual agreement is posted to the involved parties (this is for example the case in an EA, where the winning bid is always the one stating the highest price).

The *offerAllocation* element thus consists of a *matchingForm* child element specifying whether defined or forwarded offer matching is applied and an optional child element, called *matchingRule*, for the definition of the matching algorithm.

Information Processing Policy The information available during a negotiation can be divided into two data sets: the *negotiationStatus* and the *pastOffers*. The negotiationStatus is represented by all current offers of all participants allowed to post offers[16], whereas the pastOffers are all offers posted in this negotiation until now (sometimes also called *negotiation history*).

To allow for differentiated definitions of accessible negotiation data the *informationProcessing* element contains a *negotiationTransparency* and a *statusTransparency* element for the pastOffers and the negotiationStatus respectively. These two elements are both restricted to one of three possible values: *public*, *protected* and *none*. If *public* is stated, the negotiation status as well as the past offers may be queried by all agents; no restriction is applied to such information requests. *Protected* transparency denotes that only SPs and SCs involved in the actual negotiation can do so, whereas *none* is used if no data can be queried at all. This way, negotiation designers can specify, for example, sealed-bid auctions. Additionally, two optional elements defining the respective contents of past offers and status data (*negotiationContent* and *statusContent*) can be present. Here, an external restriction language, defining exactly which elements of past or current offers are visible, would have to be employed (in XML environments this could for example be XQuery (Boag et al. 2007)).

Negotiation Process After having defined the static aspects of a negotiation, the last element is dedicated to the actual *negotiation process*. (Negotiation) protocols are mostly defined in terms of state machines describing how participants (of a particular role) act during the actual communication. A state machine is defined as a set of *events*, a set of *actions*, a set of *states* (from which one is the initial and one or more are final states) and a *transition function* defining the action and / or state change a particular participant undertakes in a given state after a given event occurred (Martin 2002, p. 79).

Investigation of a significant amount of negotiation protocols has shown that while still actually negotiating the involved parties normally don't change their states regarded from the abstraction level of the negotiation protocol[17]. After having started a negotiation they stay in the NEGOTIATING state just until an agreement is reached (sending or receiving of an acceptance message), the negotiation is aborted or the participant didn't win the negotiation (receiving of a rejection message). Hence, given this language

[16] It is assumed that each participant can only have one valid offer at a time and newly posted offers replace older ones.

[17] Once negotiating they continuously send and accept negotiation messages. Internal sub-states of the high-level NEGOTIATING state and the change between such do not have any effects on this behavior within the negotiation protocol.

is intended only to describe negotiation protocols, no state changes are anticipated before the final acceptance phase. The *process* element as described in the following will thus build on events and actions only. This is arguable, however the element definition is designed in a way that it can easily be extended to incorporate the existence of several different states within the negotiation phase as well, if it becomes necessary.

All agents taking part in a negotiation adopt the SC, the SP or the *Negotiation Coordinator (NC)* role. A NC is a third party governing the actual bidding process, such as for example as an auctioneer. Employing these three roles, basically all the negotiation protocols that were investigated as part of this work can be described.

Next, the events and actions in electronic negotiations are identified. There do exist internal and external events or actions. As the EST document is intended to describe a protocol in a way that another agent can adapt to it, only external events or action are considered. Only those potentially concern the other negotiator, and are thus of interest during the negotiation process. For all negotiation protocols investigated, the sets of possible external events and actions could be identified to be equivalent; they are defined as the set of negotiation messages:

- *Call_for_Bids*: This message implements a pull mode within the actual bidding process. An auctioneer uses this message to ask the negotiators for their current offers.
- *Offer*: An offer is used by a negotiator to express its currently acceptable agreement.
- *Notification_Reject*: This message indicates that the receiver did not win the respective negotiation (or the negotiation was aborted)[18].
- *Notification_Accept*: Vice versa, an accept message is sent to the winner.
- *Still_Interested*: This message is used whenever a given SLA proposal is offered to a set of agents and these want to indicate that they are still interested. This is used in many auction protocols, where the individual bidders are not allowed to accept an offer, but only to indicate their continuing interest, while the auctioneer continuously in- / decreases a SLO value until only one interested bidder is left, the winner of the negotiation.
- *Admission*: An admission message (only permitted as an event) is used to define the first protocol step of a pro-active negotiation. Here, the SC does not react on incoming messages, it rather sends the very first negotiation message, modeled as a reaction to the admission acknowledgement message.

Employing these messages, the behavior of a SC during the protocol can be defined. Hence, the *process* consists of an unrestricted number of *protocolStep* child elements.

[18] In the current BabelNEG version this message is assumed to always be a possible action and can thus also be used by SCs for explicitly leaving a negotiation.

```
<xsd:simpleType name="RoleNameType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="serviceProvider"/>
    <xsd:enumeration value="serviceConsumer"/>
    <xsd:enumeration value="negotiationCoordinator"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="MessageType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="callForBids"/>
    <xsd:enumeration value="offer"/>
    <xsd:enumeration value="still_interested"/>
    <xsd:enumeration value="notification_accept"/>
    <xsd:enumeration value="notification_reject"/>
    <xsd:enumeration value="admission"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="EventActionType" >
  <xsd:sequence>
    <xsd:element name="messageType"
      type="MessageType" />
  </xsd:sequence>
  <xsd:attribute name="from" type="RoleNameType" />
  <xsd:attribute name="to" type="RoleNameType" />
</xsd:complexType>
```

(a)

```
<xsd:complexType name="ProtocolStepsType">
  <xsd:sequence>
    <xsd:element name="protocolStep"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="event" type="EventActionType“ />
          <xsd:element name="possibleAction"
            type ="EventActionType" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProcessType">
  <xsd:sequence>
    <xsd:element name="serviceConsumer"
      type="ProtocolStepsType"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

(b)

Figure 3.8.: Type Declaration: Process Element

Each of these defines an *event* and a list of *possibleAction* elements, that are possible for the SC in case the respective event occurs [19].

Both, events and actions, state a particular message (as specified in the *MessagesType* element) to be sent or received. For the respective *messageType* child element, one of the five different message types listed above can be chosen. As sender (in case an event is defined) or recipients (in case an action is defined) one of the three possible roles SC, SP or NC must be stated.

Figure 3.8 shows the respective XML Schema definitions of the process and protocolStep elements[20].

3.2.3. Service Identificator

Each individual service instance is defined within a respective SI. Such a document describes where exactly this service can be found (important for the actual binding process), what its type is and which EST is offered for it. It thus consists of the following elements:

[19] A description of the protocol from the NCs and the SPs view is assumed not appropriate since only SCs are potentially entering new, formerly unknown protocols at run time. A SP, just as a NC is always implemented for a defined set of protocols and will thus not have the need to adapt at run time. It would, if necessary, just be re-deployed with a new protocol. Even if an adaption of the SP is desirable this would probably be implemented by just loading another negotiation strategy component (strategy pattern (Gamma et al. 1995, p. 315)). This again would not include any generic protocol adaption as intended for the SC within this thesis.

[20] Due to their structural similarity one common type definition was created for both event and action elements.

- *serviceID*: ID for this particular SI. This element should again be of the type URI in order to ensure uniqueness.
- *serviceTypeID*: link to the description of this service's type.
- *slaTemplateID*: link to the EST used for this service.
- *wsdlFile*: reference to the WSDL file, describing the actual service interface in terms of operations with input, output parameters and error types as well as its endpoint reference (EPR), defining where the actual service instance is located.
- *negotiationCoordinator* and *serviceProvider*: these two elements represent role bindings for this service.

3.2.4. Final SLA

An electronic SLA consists of some information on the SP and the SC, the actual service and its description as well as of a set of guarantees (SLOs) on how the service is to be delivered[21]. For this, the scope of such SLOs is considered to be known before a negotiation and / or can be extracted and interpreted by the negotiators based on each SLO's declaration in the ST document.

In the context of this thesis the WS-Agreement standard (Andrieux et al. 2007) has been used as a reference structure for SLA documents. Thus, the following elements of an electronic SLA have been identified[22]:

- *slaID*: ID of this SLA as used for monitoring purposes. This element can also be found in the WS-Agreement specification.
- *context*: context information for this SLA, such as the involved parties (SP, SC), the IDs of the associated ST and EST documents and the WSDL file. WS-Agreement also defines a context element with very similar content. However, the definition of the actual service (ST element in my approach) is only given implicitly in the form of SDT and SPT elements. Similarly the reference of the actual service EPR is given in terms of the SRTs in WS-Agreement (Andrieux et al. 2007, p. 20).
- *List[SLO]*: this element finally contains all the negotiated SLOs making up this SLA document. In WS-Agreement these elements are presented as GTs.

Given the abovementioned service documents, a concrete SLA document is created by combining elements of the initial EST, the ST and the SI document. The final SLOs are based on their definition (ST), the negotiated values during the negotiation and / or their

[21] It is assumed, that a SLA always concerns only one particular service instance, or more precisely one particular configuration of one service instance. Contractual guarantees concerning more than one service are modeled as a SLA over a composite service.

[22] For each of this elements a short hint on where this can be found in the WS-Agreement specification is given.

initial values as stated in the EST. The WSDL file, the involved SP and the employed EST and ST is derived from the respective elements of the SI. The SC, representing the second contractual party is also derived during the negotiation or even during the admission process.

3.3. Protocol Design

In order to support both, the discovery and the negotiation of SLAs, a set of protocol primitives has been developed, building on the abovementioned data structures. They are shortly described in the following.

3.3.1. Discovery Phase

The discovery phase basically represents the set of activities ultimately leading to a situation where potential transaction partners (thus SCs and SPs) know one another and can start a negotiation process. This means the discovery phase is supposed to support a given SC to find one or more SI documents fitting the search criteria it received from a respective user. To this end, the SPs should be able to publish the services they offer (thus the respective SI documents) in a way that they can be found by potentially interested SCs.

As a quite common problem in SOC, a mechanism is thus needed that supports the publication and discovery of electronic documents in a distributed setting. A set of the most prominent approaches in that area has been discussed in section 2.2.2. As already elaborated there, the most promising paradigms for the IoS are probably central (but potentially replicated) registries in combination with multi- or broadcast-based messaging protocols or P2P-based infrastructures. The former offer the advantage of high-performance of the individual registry nodes and the possibility to implement complex retrieval algorithms there, whereas the latter have proven to be very resilient to node failures.

For my proof-of-concept implementation, a very simple single registry node, which can be found via a broacast-based discovery protocol, has been chosen. Due to the nature of such a prototype, technical quality metrics such as robustness (favoring P2P solutions) are not of primary interest. Therefore, a simple architecture based on a central registry is more than sufficient for investigating the research question at hand. If in future versions more elaborate mechanisms become necessary, it would nonetheless be possible to integrate them respectively, as shortly sketched in section 5.4.

Consequently, a set of protocol primitives along with the respective messages for such a discovery phase have been developed: The first step for both SP and SC, when publishing or discovering a given service, is to find a respective registry node. For that a *registryDiscovery* message has been defined, containing the ID of the sender agent. This message is sent via a broadcast in order to find one or more registries for the next steps. This basically corresponds with the broadcast-based mechanisms described in section 2.2.2. Subsequently, all registry nodes that received this message answer with an *acknowledgement (ACK)* message stating the EPR, where they are located.

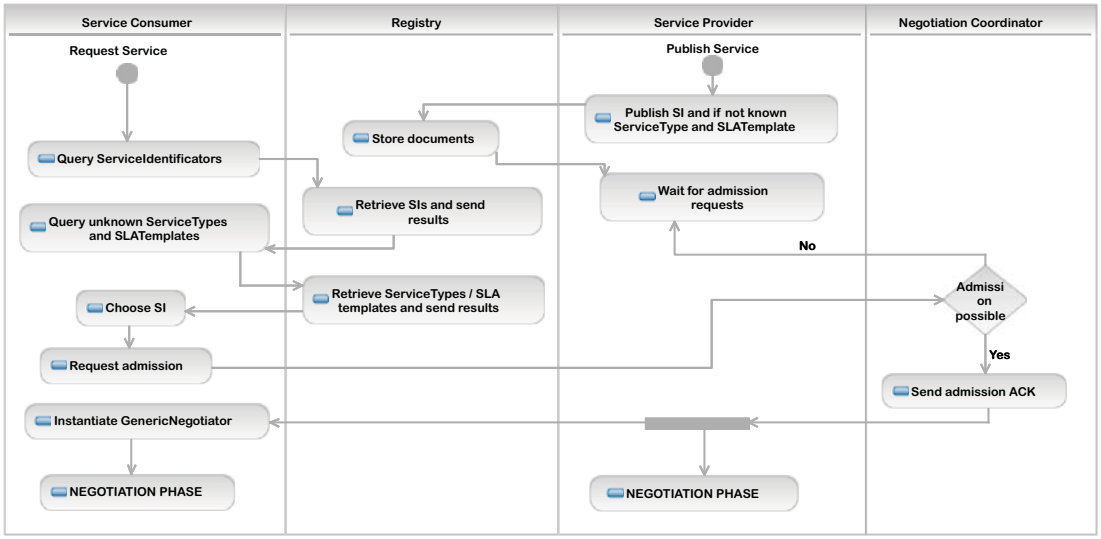


Figure 3.9.: Discovery Phase Overview

From now on SP and SC processes differ. A SP, wanting to publish a given SI, first queries whether the respective ST and EST documents associated with the SI are already available at the registry (*queryServiceTypes* and *querySLATemplates* messages). In case either of them is still unknown, the SP registers the respective document with the registry (*registerServiceType* or *registerSLATemplate* messages)[23]. Once both documents are correctly registered (as confirmed again with *ACK* messages) the actual SI can be published (*registerServiceIdentificator* message).

The SC queries the SI documents available at the registry (or registries in case more than one was found) using a *queryServiceIdentificator* message[24]. After potentially having found several fitting SIs the respective EST documents used therein are queried (*querySLATemplates* message), if not already known.

Given a list of adequate SIs were found (for which also both ST and EST documents could be retrieved), the agent chooses one of them and tries to start / join a negotiation. In order to do so, the SC must pass through an explicit admission process, for which a central entry point is assumed, the NC. The resulting process consists of a SC requesting admission to a service negotiation at the NC node, which in turn, answers with an ACK (SC was admitted) or NACK (SC was rejected) message[25]. It is therefore required that the SC inserts its credentials in the *joinNegotiation* message, so the NC agent can make a

[23] This is necessary, since respective SCs need both documents when deciding on a SI to negotiate for and when adapting to the stated protocol.

[24] In the current prototype system only the ID of a distinct ST can be used as a search criteria. The query message was however implemented in a way that other, more powerful queries, for example concerning individual elements of a ST, can be integrated. In that case, the SC would probably first query the ST documents for ones fitting its search criteria and subsequently query the SIs fitting any of the found STs.

[25] NACK is commonly used for messages stating that some request was “not acknowledged”.

valid decision on whether or not admission can be granted. Figure 3.9 gives an overview on this discovery approach[26].

The availability of a service instance is advertised implicitly in that for a busy service (the service is running or the SP is currently negotiating for it and joining this negotiation is not possible) the NC just rejects new admission requests[27].

In case of a double auction protocol (CM or CDA), the given SI is posted from the SP to the NC via an *offerToSell* message and can subsequently be matched with incoming offers to buy. After this matching, the service is busy and will not be considered for future matchings until the *offerToSell* for this SI is re-posted by the SP agent.

In case of a successful admission the SC and SP now engage in the actual negotiation process.

3.3.2. Negotiation Phase

Since the main goal of this thesis is to define an infrastructure for protocol-generic negotiations, no single one negotiation protocol can be identified for this phase. Rather, a set of negotiation message types along with their respective contents has been defined. These messages can then be used in a given negotiation process, orchestrated according to the protocol description in the EST document.

To make sure that as many different negotiation protocols as possible can be mapped to the defined message set, a thorough literature review has been conducted, basically creating the minimally necessary superset of messages used in the found protocols definitions (as listed for example in the FIPA protocol library[28]).

The result reflects the fundamental characteristic of any negotiation protocol, the fact that it mainly consists of a process of exchanging offers followed by a final acceptance or rejection of one of the involved partners. This already implies an *offer*, an *accept* and a *reject* message type. Additionally, some of the protocols found incorporate pull-paradigms within the bidding process in that they distribute *callForBids* messages to the bidders, which in turn answer with an *offer* accordingly. Finally, in some cases a central auctioneer just asks the bidders whether or not they are still interested in the currently offered agreement, while it constantly changes its offer, just until only one interested bidder is left, the winner of the negotiation. The identified message set thus results in the message types already introduced on page 82, except for the *admission* message, which is only used there to model pro-active protocols.

Using these message types, a vast variety of different protocols can be designed and subsequently described in an EST document.

[26] This figure describes the discovery process from an abstract role-centric view; no assertion on role adoptions by individual agents are made. A detailed description of this aspect is given on page 89.

[27] In order to give a rejected SC agent a meaningful reason for its rejection (it could be rejected on the basis of its credentials or because the service is unavailable at the moment) a comprehensive fault model should be used, building on a set of standardized fault types.

[28] www.fipa.org

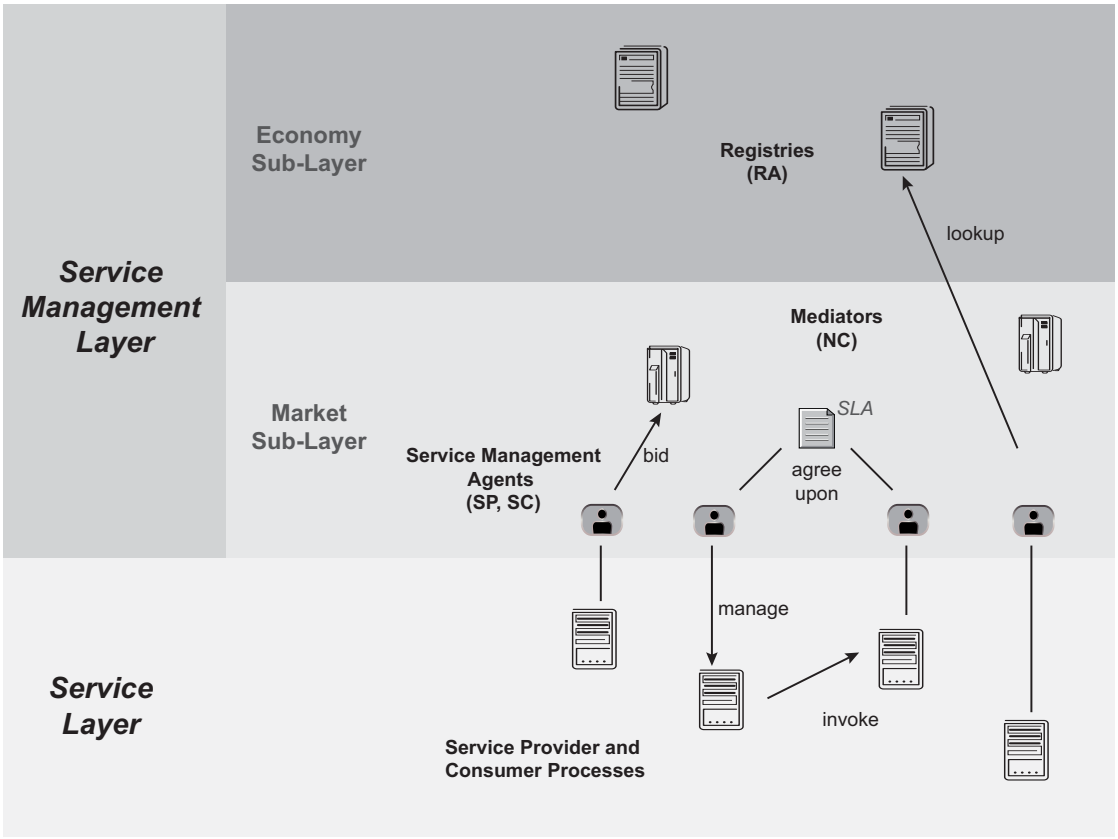


Figure 3.10.: Role-based Architecture

3.4. Architecture Design

3.4.1. Role-based Architecture

The developed prototype infrastructure builds on a defined set of roles, the service management agents can adopt. These roles are geared towards the discovery and negotiation protocols just presented.

The two basic roles present in this system are the SC and the SP, representing buyer and seller agents. Additionally a set of registry services / agents (RA) are needed in supporting the publication and discovery of service description documents. Even when other discovery mechanisms are used, the nodes providing a (sub-)set of such documents can be seen as abstract registries and would thus adopt the respective role. Finally, the NC role represents mediating agents as already sketched.

Both SP and SC agents mainly offer an interface to one another, allowing them to send and receive messages related to the discovery and negotiation phases. On a technical level, such messages are implemented as standard SOAP messages in a service-based system and are only further distinguished internally. Hence, a simple method for message reception is implemented in both roles, allowing for role-dependent internal processing

of the actual messages. Additionally, each of these roles offers one routine to external users of the infrastructure: A SP agent offers a method for publishing and selling and a SC one for discovering and purchasing a service on a user's behalf, respectively.

Due to the fact that it does not have any actual contact with external users, but is only employed within the service management infrastructure, a RA only offers the possibility to receive electronic messages. More precisely, only discovery related messages are eligible as the RA does not take part in any other phases of the life cycle.

Finally, NC agents are responsible for admission of SCs or SPs to and potentially mediation of a given negotiation. Hence, they offer methods for exchanging respective messages.

Such a centralized concept seems to contradict the vision of a massively decentralized IoS at first; however, since there is regularly one NC for each negotiation taking place, the respective nodes, as a set, already form a much decentralized admission infrastructure. Additionally, it allows for central management of trust-building mechanisms, such as security or reputation concepts (Rana et al. 2008; Silaghi, Arenas, and Silva 2007), in a coherent way across a given negotiation setting. Therefore, a central NC for every negotiation was considered an appropriate compromise between decentralism and consistent security and access management, enabling trusted service marketplaces.

The fact that a NC can also act as an independent broker allows for a set of different possible relationship configurations between SP and NC nodes:

A One agent acts as both SP and NC.

B Separate agents for SP and NC, the bidding taking place between the SC(s) and the NC.

C Separate agents adopt the SP and NC roles, the bidding taking place between the SC(s) and the SP.

In the first case one agent takes over admission (NC) as well as the actual negotiation (SP). This is the default case, as it reduces the overall amount of necessary messages to a minimum.

An independent NC, also negotiating with the SC(s), represents a central market broker (case B). In such a situation SP agents only post their offers to sell a given service to the NC node, which in turn matches them with requests received by the SC agents. After a successful matching, both SP and SC, are sent a message containing the results of this matching process (e.g. achieved agreement, transaction partner etc.).

Similarly, an independent NC agent could also be employed only for the admission of SCs (case C). The actual bidding would then take place at the SP. Newly admitted SCs would be simply notified by the NC and integrated into the negotiation by the SP accordingly (if allowed by the negotiation protocol).

In the following the internal routines of each role, as implemented in the proof-of-concept prototype, is further detailed. For better visualization UML state diagrams are given for the more complex SP and SC roles[29].

[29]Comprehensive UML activity diagrams for the different agent roles, describing all internal steps and

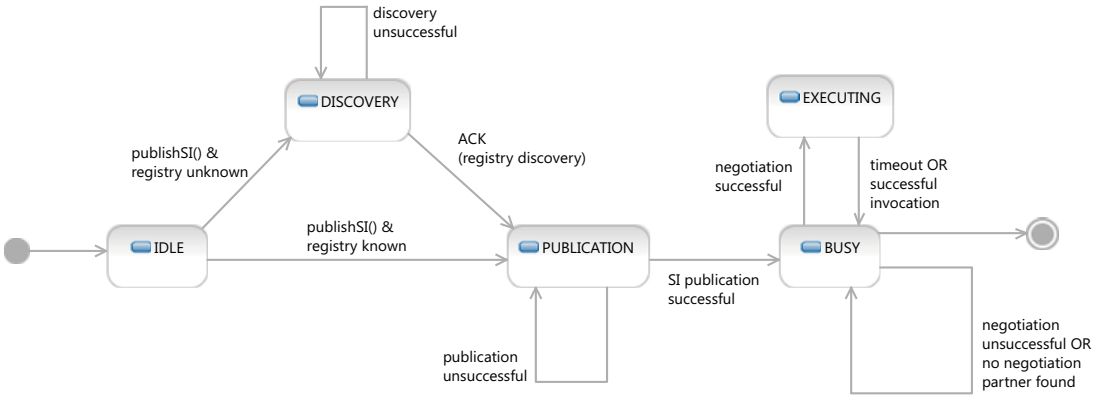


Figure 3.11.: State Diagram: Service Provider

3.4.2. Internal Behavior: Service Provider

A SP agent's basic purpose is to publish a given service to potential customers and sell it to them subsequently[30]. Figure 3.11 depicts the resulting internal behavior in terms of a state diagram.

After successful startup a SP agent is in the IDLE state, awaiting requests (by human users or automated users, such as an application hosting engine) to sell a service. Once such an inquiry has arrived, it checks whether or not a RA agent is already known and can be used for the publication of the service documents. If not, a registry discovery step, as already described, takes place.

If the RA is already known or successfully discovered, the SP starts the publication process (and changes its state accordingly). It first queries the RA for the ST and EST documents associated with the service to be published. If those are not known until now it must also register them at the RA respectively.

In both the DISCOVERY and the PUBLICATION state, timeout mechanisms are employed to ensure robustness of the system. In the developed prototype, in both cases the SP simply re-tries to discover a RA or to query and optionally submit the ST and EST documents until it finally succeeds. In future extensions of the system more elaborated strategies, for example a final abortion of the process after a certain amount of attempts, are thinkable.

Once the SP could ensure that both the ST and the EST documents are available at the market registry, the actual SI document is published. After the successful execution of this step the publication process is finished and the SP moves to the BUSY state, in which the service is continuously offered and negotiated about. The SP only exits this

control flows, can be found in appendix A.2.

[30] Within this thesis a SP agent is assumed to only manage one service offering at a time. For each new service instance a new SP agent is needed.

state when the service is taken off-line, is re-deployed or executed[31].

Whenever a negotiation could be conducted, and was successful, this agent moves to the EXECUTING state, during which the associated service is invoked by the SC that won the negotiation before[32].

After the execution phase (even if it wasn't successful and ended with the reaching of a timeout) the SP returns to the BUSY state and is again free for further negotiations[33].

Note that although up to now the first configuration, i.e. one node acting as SP and NC, was assumed, the diagram above also applies to the other two cases. In these situations the SP simply does not only publish the service documents to the registry, but also assigns the respective admission task to a NC agent. If configuration B is applied a second difference occurs, as the SP does not actually negotiate with the SC(s) but also delegates this task to the NC; this again happens in the BUSY phase. Hence, the set of possible states and the associated transitions remains the same for all three configurations.

3.4.3. Internal Behavior: Service Consumer

While the SP is assumed to only manage one service at a time, and has to be re-deployed if that service changes, the only restriction applying to a SC is that it only interacts with one SP at a time. Parallel negotiations and executions are thus out of scope for the prototype system. Due to the race conditions appearing in such settings this was considered an appropriate approach, especially since concurrent behavior of that kind is not relevant to the addressed research question.

Nevertheless, a SC can process a whole set of different service requests (as received from respective users) during its life time. Such requests can even be accepted while the SC is busy and will be processed once it has finished its current task.

This aspect leads to a more complex state diagram for SC agents, as shown in figure 3.12.

[31] As, apart from the actual negotiation, all SP agents operate in the same manner, regardless of the applied protocol, the state diagram explicitly abstracts from the negotiation phase. For this step an individual negotiation strategy component (defining the applied protocol) is loaded and all negotiation relevant messages are dispatched to it. This follows the well-known strategy pattern in object oriented software engineering (Gamma et al. 1995, p. 315).

[32] No advance reservation mechanisms are considered in this thesis, so the service is considered unavailable for new negotiation requests during this phase.

[33] Depending on the protocol applied, this state could potentially have sub-states, such as IDLE or AWAITING_BIDS. However, since this diagram aims at describing the protocol-independent behavior of a SP agent I abstracted from this protocol-dependent aspects here.

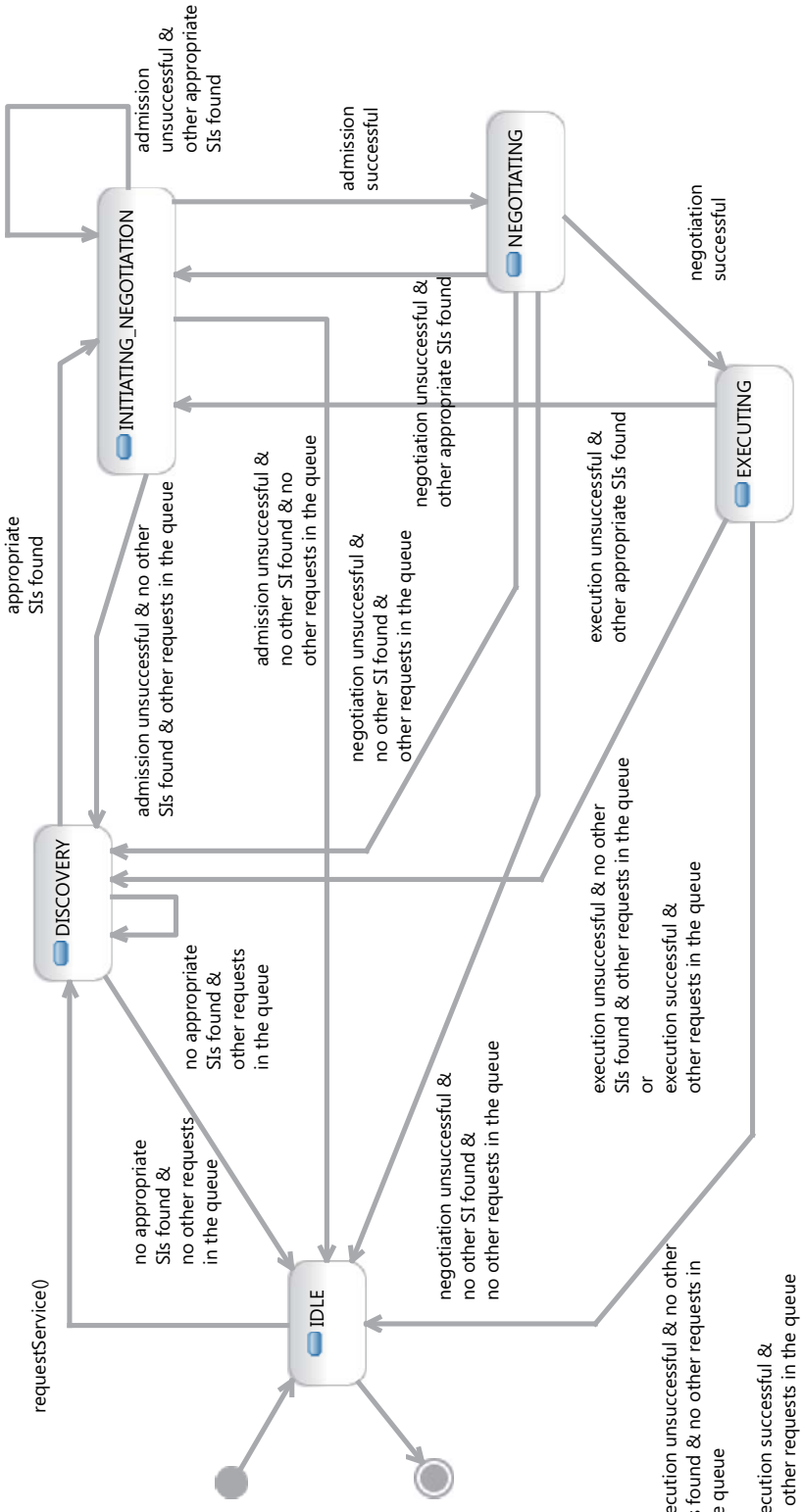


Figure 3.12.: State Diagram: Service Consumer

The default process for a SC is to receive a service request and move to the DISCOVERY state, during which a set of fitting SI documents are retrieved. After choosing an appropriate SI the SC agents requests admission to the respective negotiation (INITIATING_NEGOTIATION state), joins the negotiation process (NEGOTIATION state) and finally invokes the service in the EXECUTING state if it could win the negotiation (see figure 3.12[34]).

In a more detailed perspective, the SC agent, similarly to the SP, is in the IDLE state after start-up. After having received the first service request it moves to the DISCOVERY state and tries to locate a RA (this step has to be undertaken only once, as the RA is assumed known afterwards). Next, the SC queries the SI documents available at the RA according to the received search criteria. After having retrieved some SI documents fitting the query, the SC must retrieve the associated EST and ST documents. This is again implemented in terms of a query message sent to the RA.

When the discovery timeout occurs the agent checks whether it could retrieve one or more SIs fitting the search criteria and for which it could retrieve the respective EST and ST documents in time[35]. If so, it changes to the INITIATING_NEGOTIATION state and sends an admission request to the NC, stated in the chosen SI[36]. If no fitting SI could be found or the respective EST documents could not be retrieved in time, this request is considered failed. The agents re-starts the discovery process if it has received another request in the meantime (such requests are simply buffered until they can be handled) or goes back to IDLE if not.

This principle can be seen all along the process of a SC agent: In case of a failed attempt to move to the next state (here it could not move from the DISCOVERY to the INITIATION_NEGOTIATION state) it checks its options for another such attempt. In this case it checks whether or not any other SI was found, which it could request admission to. Only if no such options are available it moves back to the predecessor state and checks whether it can start over from there or if even there no other options are available and so on.

This can be illustrated with the next state change from INITIATION_NEGOTIATION to NEGOTIATION. If the SC was rejected for a given negotiation it will first check whether there are other SIs it to start a negotiation with. If so, the SC will simply proceed with the overall process as usual. If no such SIs could be found, it will check if other service requests have been received in the meantime; this would mean it can re-start the process in the DISCOVERY state. Only if even this option is not possible, it will move back to the IDLE state.

Due to the fact that no parallel negotiations are considered, the NEGOTIATION step differs slightly from this principle, as in case of an unsuccessful negotiation the SC cannot just proceed with another negotiation process, but has to retract to the

[34] Due to readability reasons, the registry discovery process, which is analog to the one applied for the SP, was omitted in this diagram.

[35] In doing so it also ignores those SIs, which are a) invalid (in that the EST does not fit to the ST) and b) for which an agreement is not at all possible in terms of the stated SLO restrictions.

[36] Whenever more than one fitting SI was found, the one to start a negotiation with is chosen randomly in the prototype system.

INITIATION_NEGOTIATION state and start over from there.

During the EXECUTING state this leads the SC to be able to change to the INITIATION_NEGOTIATION, the DISCOVERY or the IDLE state, depending on whether the invocation was successful or not, whether other SIs were found the agent can start a new negotiation with or, if not, whether other service requests have been received in the meantime.

Similarly to the SP, this diagram does not illustrate the actual negotiation phase. The internal routines enabling a SC to adapt to a given protocol are described separately in subsection 3.4.6.

3.4.4. Internal Behavior: Registry

The RA node is a placeholder element for any discovery mechanism used in future versions. It was designed to simply receive register messages and store the corresponding data into an internal data structure. This is for example used when a SP publishes a new SI document. Supporting the discovery of service documents, it also offers the possibility to query the stored SI, EST and ST documents. No complex internal states are maintained and changed throughout its life time. It simply processes requests to register or query service documents or answers *registryDiscovery* messages.

This very basic registry service was considered appropriate due to the fact that the developed system only acts as a proof-of-concept. Future version will probably replace it with a more sophisticated registry implementation or even use more decentralized discovery mechanisms.

3.4.5. Internal Behavior: Negotiation Coordinator

The main task of a NC agent is to handle the admission of SCs to a given negotiation. For this, it offers the possibility to submit respective messages along with a set of credentials, as needed for the admission decision.

Additionally, if the negotiation phase is also assigned to the NC (configuration C), it must also be able to process incoming negotiation messages. Just with configurations A and B (see page 89) these are simply forwarded to a negotiation strategy component, which in turn provides the protocol-specific functionality.

Similarly to the RA nodes, a NC thus does not expose complex internal states and state changes. Admission requests are evaluated and answered based on the received data and the implicit service availability information, and negotiation related messages are simply forwarded to the strategy component.

3.4.6. The protocol-generic SC Strategy Component

The strategy subcomponent of an SC agent represents one of the central deliverables of this thesis, as basically all routines needed for adaption to a new negotiation protocol are implemented herein.

During the instantiation process for a respective *Generic Negotiator (GN)* component, it receives from the SC:

- the service description documents,
- the ID of the current negotiation,
- a link back to the SC,
- the constraints the user posed on the SLA to be negotiated (e.g. her reservation values) and
- some additional negotiation parameters, such as timeout or concession values.

As a first step, the GN partitions the stated SLOs into those that are fixed and those that can be negotiated. The former are simply stored. For each of the latter a new *SLONeg* object is created, containing both the initial SLO (property ID and value) and all additional information that is associated with it (e.g. attributeRestrictions or domain descriptions). These objects are then again saved and subsequently treated as the first offer from the negotiation partner.

Next, the *SLOConstraint* objects initially received from the user are processed, each of which basically describes the user's demands on one particular SLO under negotiation (fixed or negotiable). Following the rationale from above, two types of such constraint classes are present: *EnumerationSLOConstraints* or *OrderedSLOConstraints*, depending on the value domain of the underlying property[37].

Both provide methods for requesting the stated restrictions (e.g. upper or lower reservation values or concession steps in case of an ordered domain) and simple methods calculating whether or not a given (received) value fulfills this constraint or, contrarily, is so far away from any acceptable value that this offer can be regarded as rejectable. Additionally, they also provide the GN with a method for creating counter offers based on a received value.

An *EnumerationConstraint* object basically states a set of acceptable values, based on which it can decide whether or not a received SLO value is acceptable. Intuitively received values applying to an *EnumerationConstraint* are never rejectable, as no assertion can be made whether or not the negotiation partner would accept one of the demanded values (these values do not expose any relationship from which such an assertion could be deduced). In the current prototype, when creating a counter offer, simply one acceptable value is randomly chosen and returned[38].

Contrarily, an *OrderedSLOConstraint* defines a range of acceptable values, regularly only bounded on one side (either a minimum or maximum reservation value is present[39]). Consequently an offer is deemed acceptable based on these boundary values. Deciding

[37] For each constraint received from the user an internal constraint object, representing it, is created during initialization of the GN.

[38] In future versions, potentially a list of all acceptable values will be sent in order to assure determinism of the negotiation.

[39] It is assumed that a negotiator has a clear preference on a ordered negotiation parameter. She will therefore accept all values as long as they are higher / lower than her reservation value. A SC will for example accept a price as long as it is lower that a given reserve value but will not have any lower boundary for the acceptable values.

whether or not an offer is rejectable is more complicated, as it differs from one user to the other.

For the creation of a counter offer, two cases must be distinguished: an offered value for the respected SLO was already received or not. In the latter case, this agent posts the first offer for this SLO. Here it simply offers the best value acceptable for the SP (stated as upper or lower reservation values in the EST) or if such values are not specified it offers half its own upper reservation value / double its lower reservation value[40]. In the former case it just concedes from its lastly posted offer by an amount, specified by the user for this particular SLO during instantiation (concession step value).

Once all this information was extracted the GN checks whether it has to start the negotiation (pro-active protocol). If so, two possible actions can be defined: this agent can be allowed to post an offer or to simply accept all the values stated in the EST (catalogue pricing model).

Here, a basic principle of the GN becomes obvious: it always seeks to maximize its profit and thus will always choose some actions over others (if both are allowed). In this case (pro-active protocol) it will first check, whether an offer is possible, which could further improve the currently offered agreement. If so, an offer will be sent; if not it will simply accept the current values, as no negotiation on them is allowed.

The same principle applies throughout the whole negotiation process. Whenever a negotiation message is dispatched to the GN it checks its options: In case of a reject or accept message the negotiation is over. It simply processes the result in that it passes the respective information / reached agreement to the SC agent. In case of a *callForBid* message it creates an offer and sends it to the SP agent[41]. All these possibilities are straightforward, as they don't give the GN any option to choose among a set of possible actions.

When receiving an offer, this could potentially change. If the offer is not completely rejectable (this is checked by iterating over all involved SLOConstraint objects) and a counter offer is possible, the GN will always do so. A counter offer is always the best option, as it potentially increases value of the agreement to be reached for the user. If this is not possible, the GN will check whether a *stillInterested* message is allowed, providing it with the possibility of an ongoing negotiation, even if it cannot actively influence the changing of the negotiated values. If even this is not possible it will finally check whether the received offer can be accepted or in the end rejected completely and do so. This routine gives the GN the possibility to react on incoming messages in a way that maximizes its further options during the negotiation and in the end potentially even its utility in terms of the reached agreement.

Summarizing this chapter, a detailed overview on the developed service description documents, the employed discovery and negotiation protocols and the designed software agents employed therein has been given. In the next chapter I describe how these concepts can be instantiated and how the prototype system can be deployed and started.

[40] These factors are chosen randomly and would be provided by the user in future versions.

[41] Offer messages are simply created by iterating over each not yet fulfilled constraint object and triggering each to provide a counter offer value, as described before. These values are then combined to one bundled *offer* message.

Also, the results of a demonstration of the prototype's effectiveness (with regard to the stated requirements) as well as its efficiency (evaluation step) are given.

4. Assessment of the Developed System

After having presented the infrastructure design in the last chapter, the results of its assessment are presented in the following.

4.1. Prototypical System Implementation

The concepts introduced in chapter 3 have been implemented in a Java-based proof-of-concept prototype infrastructure, building on the agent-based IoS simulation toolkit SimIS (König, Hudert, and Eymann 2010). This way a multitude of different configurations, even very extreme market situations, can be assessed only focusing on the negotiation and discovery phases, before the components are ported to a productive IoS platform dealing with the whole service life cycle.

This toolkit aims at providing researchers with a comprehensive framework for investigating distributed algorithms or protocols within the context of the IoS. Building on the generic Recursive Porous Agent Simulation Toolkit (REPAST) (North, Collier, and Vos 2006), it proposes a two-tiered architecture dividing the overall system into an *Application Layer (AL)* and an *Infrastructure Layer (IL)* (see figure 4.1).

The IL models topological settings of the IoS, whereas the AL represents the actual set of service management agents and respective services. The fundamental idea is that all AL agents are linked to a single IL agent each, which represents their server platform. This infrastructural node deals with message handling and routing issues.

In the AL the actual services of the IoS vision are defined, communicating via the offered messaging interfaces and routing functionalities offered by the IL[1]. Each service (as represented by an AL agent) is implemented as a plain Java class and can therefore exploit the full potential this programming language offers.

The agent types and messages developed for this prototype have consequently been implemented as specific AL agents and Java message objects within SimIS. Each of the management agents (SCs and SPs) is additionally accompanied by a strategy object, the SCs with the protocol-generic GN component and the SPs with a strategy distinctly fitting to the offered protocol (as parameterized during startup).

[1]The message structures and service / agent interfaces, building on them, have been designed equivalently to real-world SOAP messages and WS interfaces. This eases porting the prototype to a productive infrastructure building on WS technologies (the de facto standard for such infrastructures as of today) in the future.

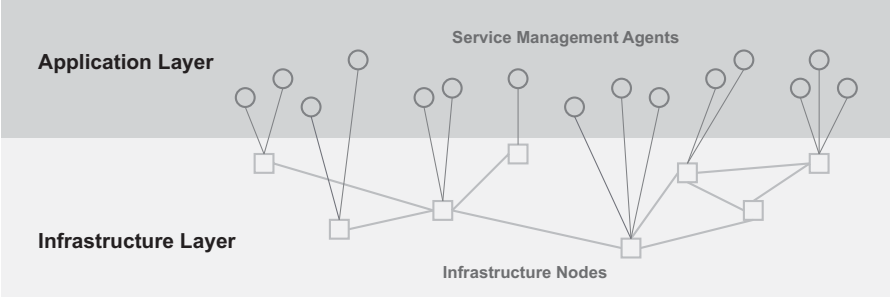


Figure 4.1.: SimIS Toolkit

4.2. Conceptual Demonstration

As described in 1.3.2, this section assesses the effectiveness of the developed infrastructure. In this demonstration step the capability of the designed mechanisms and data structures to fulfill the posed requirements is investigated.

4.2.1. Assessment on the Basis of the Stated Requirements

Recapitulating the requirements stated in 2.1, the following assertions can be made:

R1: Need for market registries The discovery phase in my approach explicitly builds on a registry role (RA) for publication and discovery of the service description documents (EST, ST and SI).

R2: Support for SLAs of different complexity as a conceptual basis for the service management

The ST, EST and SLA documents provide a very generic service (SLA) description structure. By offering rule-based text elements along with the pre-defined and typed elements for quantitatively measurable service aspects, a comprehensive service description can be created. The defined document structures also enable the usage of external, standard languages for describing service characteristics (for example WSDL when describing the service interface).

R3: Support for restrictable SLA offerings, including non-negotiable terms.

The separation of offered service parameters (defined in the ST document) and the assertions on their negotiability as well as their starting values (stated in the EST document) directly corresponds to that claim. By defining re-usable ST and EST documents, a very generic approach for restricting individual SLA parameters (to pre-set values) is offered.

R4: Support for protocol-generic SLA negotiations

This represents the most fundamental claim made in this thesis. The main goal of my work is to define mechanisms and data structures allowing for the dynamic creation of knowledge about and subsequent adaption to a priori unknown protocols for negotiating electronic SLAs.

Each SC agent, present in the developed infrastructure is accompanied by a protocol-

generic negotiation strategy component, the GN. This module is capable of parsing negotiation protocol descriptions coded within EST documents and subsequently taking part in the respective negotiation process. No prior knowledge about the protocol to be executed is necessary for this.

A more detailed proof of feasibility is given in subsection 4.3, underlining the statement just made.

R5: A priori unrestricted set of possible negotiation protocols

The GN component is capable of adapting to every protocol that can be described within an EST document. The only restriction posed on the set of possible protocols is thus the structural restriction of the EST document and the used ontology of SLO parameters (this is assumed to be defined for the given industry the infrastructure is used in).

Hence, given the EST document offers a high enough generity for describing different protocols, this requirement can be assumed fulfilled. In subsection 4.2.2 this expressiveness is assessed in more detail.

R6: Software agents acting as negotiators

The system architecture, as implemented in the proof-of-concept prototype, heavily builds on software agents as instantiations of the defined roles. Such components are the basic actors in the BabelNEG system.

R7: Need for intermediaries

The developed infrastructure incorporates an explicit intermediary role, used for the definition of market brokers, the NC.

In summary, the requirements can be considered fulfilled, once the expressiveness of the EST in describing different negotiation protocols and the GN's ability to adapt to such descriptions can be shown. Respective efforts are made in the following.

4.2.2. Conceptual Assessment of the Service Description Documents

The overall goal of the service description documents is to enable the definition of all negotiation protocols and SLA characteristics commonly used in electronic negotiation settings. However, there is no exhaustive list of such aspects, upon which the expressiveness of the language could be evaluated. Therefore, for both the service / SLA and protocol description parameters, representative examples must be identified and further used for demonstration purposes.

The description of an electronic service or SLA document commonly comprises a set of functional and non-functional characteristics of the respective service instance. A SLA document additionally specifies some context elements, such as the involved SC and SP. This basic structure can be found in all relevant standard languages currently in use (see for example Andrieux et al. 2007; Lamanna, Skene, and Emmerich 2003; Ludwig et al. 2003a).

Therein, the functional properties of a service are described using some interface description language, such as WSDL, sometimes also augmented with semantic annotations. On the other hand, the non-functional aspects are defined as tuples of QoS metric definitions and respective values.

```
<ServiceType>
  <serviceTypeID>
    http://www.abc.com/demandForecasting
  </serviceTypeID>
  <serviceDescription>
    http://www.abc.com/demandForecasting/functionality
  </serviceDescription>
  <property propertyID="sla/price">
    <domain>Double</domain>
    <declaration>http://www.sla.org/ontology/
      price</declaration>
  </property>
  <property propertyID="scm/forecastingAlgorithm">
    <domain>String</domain>
    <declaration>http://www.scm.org/ontology/
      forecastingAlgorithm</declaration>
  </property>
  <property propertyID="sla/allocatedMemory">
    <domain>Double</domain>
    <declaration>http://www.sla.org/ontology/
      allocatedMemory</declaration>
  </property>
</ServiceType>
```

(a) ST Document

```
<ServivceIdentificator>
  <serviceID>
    http://www.xyz.com/demandForecasting
  </serviceID>
  <serviceTypeID>
    http://www.scm.com/demandForecasting
  </serviceTypeID>
  <slaTemplateID>
    http://www.slaNeg.com/standardDutchAuctionTemplate
  </slaTemplateID>
  <wsdlFile>
    http://www.xyz.com/demandForecasting/
      forecastingService.wsdl
  </wsdlFile>
  <negotiationCoordinator>
    http://www.xyz.com/demandForecasting/
      dutchAuctionNegotiator
  </negotiationCoordinator>
  <serviceProvider>
    http://www.xyz.com/demandForecasting/
      dutchAuctionNegotiator
  </serviceProvider>
</ServivceIdentificator>
```

(b) SI Document

Figure 4.2.: Sample Service Description Documents

In the BableNEG system, the characteristics of a service are defined in the ST, the SI and, after successful negotiation, in the SLA document as presented in section 3.2:

The functional aspects of a service are expressed within the ST and the WSDL file[2], which is referenced in the SI document. The involved parties (before the SLA is signed only SP and NC are known) are also stated therein. Additionally, the non-functional aspects a service of a given type exposes, and can thus offer guarantees on, are defined in the ST document. The actual values of these metrics are not necessarily defined before the actual negotiation. In contrast, a service mostly only offers the possibility for defining a guarantee on a particular SLO before the actual SLA negotiation. The actual value of this guarantee is regularly agreed upon during the negotiation process.

If some of the metrics are already associated with values before a negotiation, this indicates either a starting value to be argued over subsequently or a fixed characteristic that cannot be altered for the described service instance (SLO is non-negotiable). Such initial SLO values are stated in the EST document when defining the negotiation object.

A short demonstration of these aspects is given in figures 4.2(a), 4.2(b) and 4.3, showing example ST, SI and SLA documents, adhering to the schema definitions presented in section 3.2.

They describe a demand forecasting service, offering three non-functional attributes: the price, the allocated memory (determining the duration of each forecasting run) and the used forecasting algorithm itself. The first two are of an ordered domain (Double), representing US Dollar (USD[3]) and gigabyte (GB) values respectively, as opposed to

[2] In traditional SOS, only such a WSDL description (potentially with some semantic extensions, as for example used in (Overhage and Thomas 2005)) is used.

[3] The used measure for a SLO is assumed to be pre-defined in the context of the target industry or, alternatively, specified in the ST document.

```
<SLA>
  <slaID>
    http://sla.org/14294
  </slaID>
  <context>
    <serviceProvider>
      http://www.xyz.com/demandForecasting/
      dutchAuctionNegotiator
    </serviceProvider>
    <serviceConsumer>
      http://www.abc.com/scm/negotiator
    </serviceConsumer>
    <serviceID>
      http://www.xyz.com/demandForecasting
    </serviceID>
    <serviceType>
      http://www.scm.com/demandForecasting
    </serviceType>
    <slaTemplate>
      http://www.slaNeg.com/standardDutchAuctionTemplate
    </slaTemplate>
    <wsdlFile>
      http://www.xyz.com/demandForecasting/
      forecastingService.wsdl
    </wsdlFile>
  </context>
  <SLO propertyID="sla/price">
    <value>1.50</value>
  </SLO>
  <SLO propertyID="scm/forecastingAlgorithm">
    <value>bayesianForecasting</value>
  </SLO>
  <SLO propertyID="sla/allocatedMemory">
    <value>4.0</value>
  </SLO>
</SLA>
```

Figure 4.3.: Example SLA Document

the third, exposing an unordered domain (String, representing the algorithms name). Additionally, a WSDL file for this service can be found at "http://www.xyz.com/demand Forecasting/forecastingService.wsdl".

After the negotiation, a SLA document was achieved between the SP "http://www.xyz .com/demandForecasting/dutchAuctionNegotiator" and the SC "http://www.abc.com/ scm/negotiator" defining a price of 1.50 USD. The "bayesianForecasting" algorithm is to be used and an amount of 4 GB of memory is allocated to this service.

In the following, the documents' capabilities to describe actual negotiation protocols are assessed. For this, a representative set of mutually as different as possible protocols[4] has been selected. Such an approach cannot claim completeness, however since there is no comprehensive list of possible protocols, there is no approach able to claim that. By choosing very different protocols at least a sufficiently generalizable assertion on the language's expressiveness can be made.

The protocols chosen for this assessment are an EA, a DA (both 1:N protocols), a FPA, a CM (M:N protocol)[5] and two types of AO protocols (1:1 protocols), one being

[4]The chosen protocols are mainly based on the FIPA interaction protocol library, being one of the most commonly used protocol taxonomies for agent-based interactions.

[5]This protocol follows the M+1 clearing algorithm, presented in (Wurman, Walsh, and Wellman 1998b).

single-attributive and one multi-attributive. In the following, each of these protocols is presented in terms of a UML sequence diagram and subsequently described in terms of a respective EST document. As the negotiation object, a service of the type described in figure 4.2(a) is assumed.

From a protocol perspective, the EA and the DA only differ slightly. In both protocols the SP (in this case also acting as NC) multi-casts an offer message to the SCs involved in the negotiation, which in turn answer by expressing whether they are still interested in the currently offered SLA or not. After an internal timeout the SP in- (EA) or decreases (DA) the current price value. After this step another round of *offer* multi-cast and *stillInterested* messages takes place. This process stops whenever at most one SC is still interested in the new offer. This SC then wins the negotiation[6]. Figure 4.4 depicts this process. The corresponding EST document for an EA protocol is described in figure 4.5[7].

In both protocols the configuration of one SP and many SCs is defined within the context-element. Also, the price is defined as the one negotiable SLO for both protocols. The actual process can be described by only one protocolStep element[8]. It states an incoming offer from the SP as the triggering event and allows the SCs to answer with a *stillInterested* message. The price starts at a value of "1.0 USD" and is constantly increased by "1.0" in each new round[9]. Finally, both protocols define protected information policies, allowing all involved agents to query available negotiation information.

The Fixed Price Auction basically represents a catalogue pricing model, not defining any multi-round offer exchange. The SC can only accept the offered SLA or reject it (see figures 4.6).

Accordingly, the EST document for this protocol already states values for the offered SLA metrics (a price of "10.0 USD", "4.0 GB" of allocated memory and the forecasting algorithm "bayesianForecasting"). No negotiable SLOs are defined, reflecting the catalogue pricing scheme. This protocol is also pro-active in that the SC has to act directly after being admitted to the negotiation (by accepting or rejecting the offered SLA). This is reflected in the process-element as shown in figure 4.7[10].

The CM is a M:N protocol, where both sides (SC and SP) post an *offer* or *offerToSell* to the central broker. The broker then matches an *offer* to a fitting *offerToSell* (according to an internal matching function) and forwards the matching result to the respective bidders. If no matching could be calculated the respective bidder is sent a *notification_reject* message.

Similarly to the FPA, the respective EST document defines a pro-active bidding process, in which the only possible action for a SC is to post an *offer* to the broker.

[6] In case no SC is still interested, the first SC having answered in the last round wins the negotiation.

[7] The respective DA description would only differ in terms of the progress element (descending instead of ascending) and it would define a upper instead of a lower bound.

[8] This is the case since only the SC side of the process must be described and the acceptance of *notification_accept* and *notification_reject* messages is assumed to be implicitly possible for any protocol.

[9] In the DA the price would start for example at "20.0" and would constantly be decreased by "1.0".

[10] For clarity reasons only the parts of the EST actually differing from one protocol to another are shown for the remainder of this section (the differences to the EA are highlighted). For a complete EST, refer to figure 4.5 for an overview of an EA protocol.

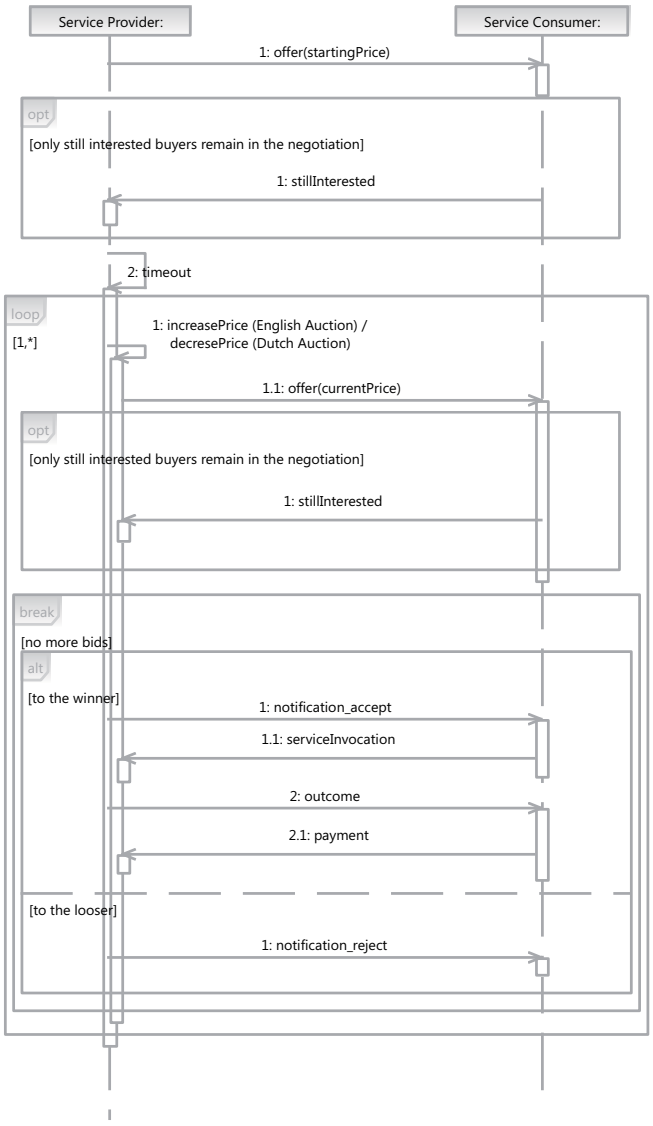


Figure 4.4.: Sequence Diagram representing an EA / DA Protocol

Finally, the (multi-attributive) AO protocol is used for bilateral offer exchanges. It is characterized by both SPs and SCs posting *offers* to each other. After reception of such an *offer*, it is evaluated and, depending on the content, a counter-offer, a *notification_accept* or a *notification_reject* message is sent to the opponent. Figure 4.10 shows this process[11].

[11]Due to clarity reasons, only the accept case was modeled; reject messages are possible in the same way as the shown accept messages.

```
<ExtendedSLATemplate>
  <slaTemplateID>englishAuctionTemplate</slaTemplateID>
  <slaTemplate>
    <SLO propertyID="scm/forecastingAlgorithm">
      <value>bayesianForecasting</value>
    </SLO>
    <SLO propertyID="sla/allocatedMemory">
      <value>4.0</value>
    </SLO>
  </slaTemplate>
  <negotiationProtocol>
    <context>
      <serviceProvider>
        <maximumNumberOfAgents>
          1
        </maximumNumberOfAgents>
        <minimumNumberOfAgents>
          1
        </minimumNumberOfAgents>
        <admissionRestriction admissionRestrictionForm="open"/>
      </serviceProvider>
      <serviceConsumer>
        <maximumNumberOfAgents>
          -1
        </maximumNumberOfAgents>
        <minimumNumberOfAgents>
          1
        </minimumNumberOfAgents>
        <admissionRestriction admissionRestrictionForm="open"/>
      </serviceConsumer>
    </context>
    <negotiationObject>
      <negotiableSLO propertyID="sla/price">
        <values>single</values>
      </negotiableSLO>
    </negotiationObject>
```

(a)

```
    <offerRestrictions>
      <attributeRestriction propertyID="sla/price">
        <threshold>
          <lowerBound>1.0</lowerBound>
        </threshold>
      </attributeRestriction>
      <attributeRestriction propertyID="sla/price">
        <progress>
          <progressForm>ascending</progressForm>
          <delta>1.0</delta>
        </progress>
      </attributeRestriction>
    </offerRestrictions>
    <offerAllocationPolicy matchingForm="forwarded"/>
    <informationPolicy>
      <negotiationTransparency>
        protected
      </negotiationTransparency>
      <negotiationContent>none</negotiationContent>
      <statusTransparency>protected</statusTransparency>
      <statusContent>agent_price</statusContent>
    </informationPolicy>
    <process>
      <serviceConsumer>
        <protocolStep>
          <event from="serviceProvider" to="serviceConsumer">
            <messageType>offer</messageType>
          </event>
          <possibleAction from="serviceConsumer"
            to="serviceProvider">
            <messageType>stillInterested</messageType>
          </possibleAction>
        </protocolStep>
      </serviceConsumer>
    </process>
  </negotiationProtocol>
</ExtendedSLATemplate>
```

(b)

Figure 4.5.: Sample EST Document for an EA protocol

The only difference between an multi-attributive and a regular AO protocol is that the former allows for negotiating about more than just a single SLO.

In the demand forecasting service example this means that in the former price and allocated memory and in the latter only price are negotiable (see figures 4.11 for an EST description of the AO protocol[12]).

These example instantiations provide a good estimation of the expressiveness achieved with the developed language structure. Consequently, the last remaining aspect that is to be shown before all stated requirements can be assumed fulfilled is the actual protocol adaptability based on these documents. This is addressed in the next section.

4.3. Simulative Demonstration of the Prototype's Effectiveness

For the simulative demonstration of the proof-of-concept prototype (and thus the agents' capability to adapt to new negotiation protocols) the aforementioned SimIS toolkit is

[12]The multi-attributive AO protocol would simply state the allocatedMemory as a secondary negotiable SLO, but would not differ otherwise.

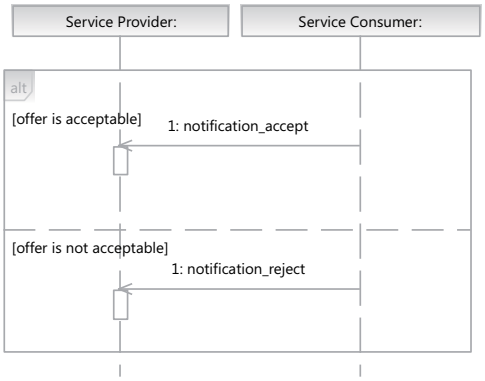


Figure 4.6.: Sequence Diagram representing a FPA Protocol

```
<ExtendedSLATemplate>
 <slaTemplateID>fixedPriceAuctionTemplate</slaTemplateID>
 <slaTemplate>
  <SLO propertyID="sla/price">
   <value>10.0</value>
  </SLO>
  ...
 </slaTemplate>
 <negotiationProtocol>
  <context>
   ...
   <serviceConsumer>
    <maximumNumberOfAgents>
     1
    </maximumNumberOfAgents>
    <minimumNumberOfAgents>
     1
    </minimumNumberOfAgents>
    <admissionRestriction admissionRestrictionForm="open"/>
   </serviceConsumer>
  </context>
   ...
  <process>
   <serviceConsumer>
    <protocolStep>
     <event from="serviceProvider" to="serviceConsumer">
      <messageType>admission</messageType>
     </event>
     <possibleAction from="serviceConsumer"
      to="serviceProvider">
      <messageType>notification_accept</messageType>
     </possibleAction>
    </protocolStep>
   </serviceConsumer>
  </process>
 </negotiationProtocol>
</ExtendedSLATemplate>
```

Figure 4.7.: EST Document defining a FPA Protocol

employed. Before detailing the actual simulation settings, some characteristics of the prototype, stemming from the usage of SimIS, have to be clarified:

REPAST, and thus SimIS, is a time-discrete simulator, hence all computations occur in distinct action steps, called *ticks*. On the one hand, this eases the sequencing of actions in a (although only simulated) massively decentralized setting by avoiding race

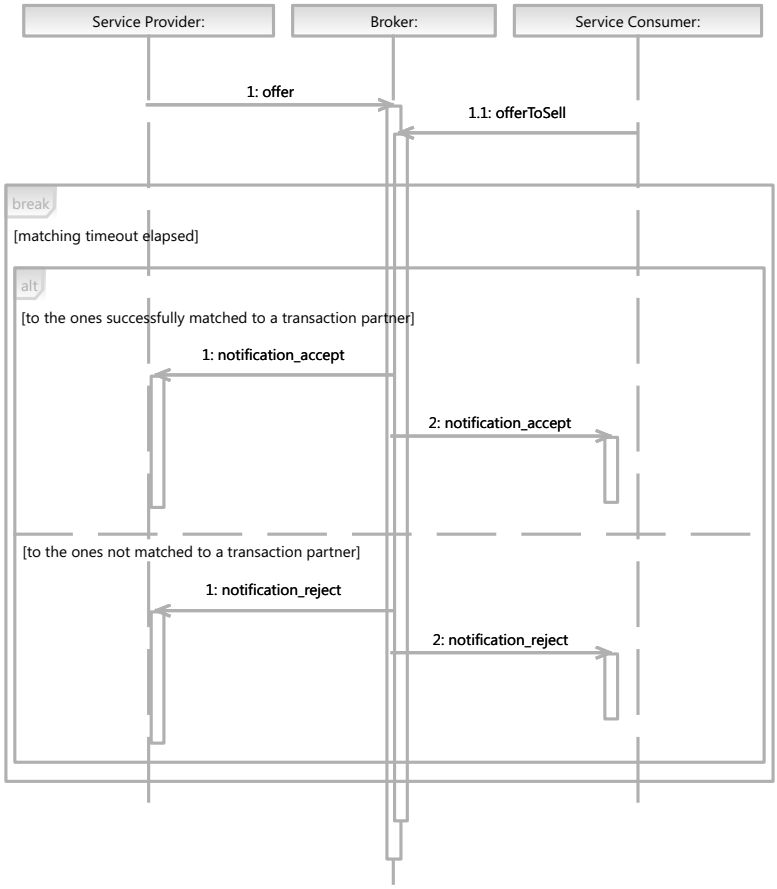


Figure 4.8.: Sequence Diagram representing a CM (Double Auction) Protocol

conditions. On the other hand, it inhibits direct investigations of technical performance metrics, such as response time. Given the nature of a proof-of-concept prototype such aspects are of minor interest, however, as the feasibility of the approach represents the main result to be proven. Thus, SimIS was assumed appropriate for the research project at hand.

In order to at least approximate real-world timing sequences, individual steps throughout the discovery and negotiation processes have been mapped to a corresponding amount of ticks, necessary for their completion. Each hop a message traverses from sender to recipient takes a distinct amount of ticks (depending on the bandwidth associated with the respective link during parameterization). The reception and internal interpretation of a message (including the creation of a potential answer message) has been designed to take exactly one tick.

The internal architecture of the BabelNEG system (building on the SimIS toolkit) has already been sketched in subsection 4.1. A set of IL, connected with bi-directional links, represent the topology of server nodes in the IoS setting. Each of these connections is

```
<ExtendedSLATemplate>
  <slaTemplateID>doubleAuctionTemplate</slaTemplateID>
  <slaTemplate>
    ...
  </slaTemplate>
  <negotiationProtocol>
    <context>
      <serviceProvider>
        <maximumNumberOfAgents>
          -1
        </maximumNumberOfAgents>
          ...
        <admissionRestriction admissionRestrictionForm="open"/>
      </serviceProvider>
      ...
    </context>
    <negotiationObject>
      <negotiableSLO propertyID="sla/price">
        <values>single</values>
      </negotiableSLO>
    </negotiationObject>
      ...
    <process>
      <serviceConsumer>
        <protocolStep>
          <event from="negotiationCoordinator"
            to="serviceConsumer">
            <messageType>admission</messageType>
          </event>
          <possibleAction from="serviceConsumer"
            to="negotiationCoordinator">
            <messageType>offer</messageType>
          </possibleAction>
        </protocolStep>
      </serviceConsumer>
    </process>
  </negotiationProtocol>
</ExtendedSLATemplate>
```

Figure 4.9.: EST Document defining a CM (Double Auction) Protocol

associated with a logical bandwidth attribute delimiting the amount of data that can cross this link in a single tick[13].

Located on this topology, implemented as AL agents, are the service management agents mentioned in section 3, SPs, SCs, RAs and NCs. Each SC is accompanied by one GN component, handling the negotiation processes. Correspondingly, each SP is accompanied by a strategy component, geared towards the protocol offered by this SP. This way the adaptability of a SC, and the GN module respectively, to a variety of different market situations can be assessed.

The SP agent class has been implemented as a generic wrapper of the individual strategy modules, regardless of the supported protocol. It simply passes all messages received during the negotiation phase to the strategy module, which is protocol-dependent.

In addition to regular SPs a second type is needed that is able to interact with a broker. Because of the modular nature of the SP agents, this type (called *Independent Service Provider (ISP)* in the following) only differs slightly from the regular SP, as it simply delegates the negotiation-relevant message handling to an external node (the NC)

[13]Incorporating the time-discrete paradigm, this bandwidth is specified in terms of message elements per tick.

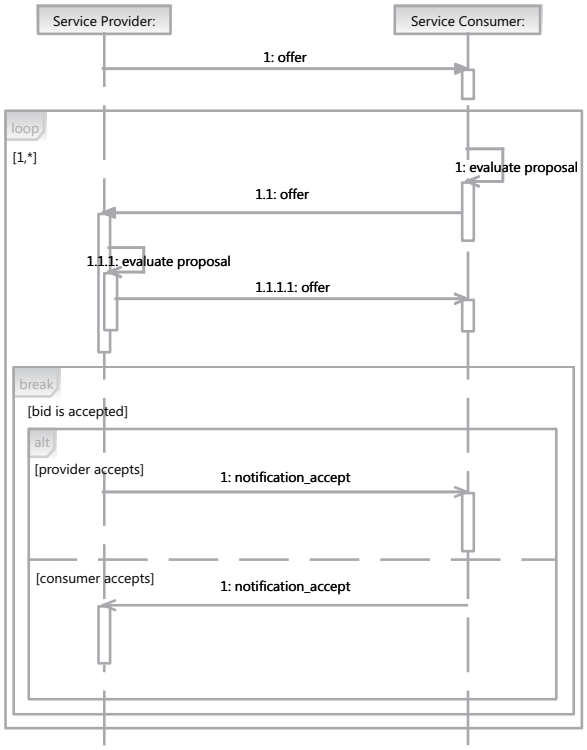


Figure 4.10.: Sequence Diagram representing an AO / MAO Protocol

instead of the internal strategy module. The just mentioned brokers are finally assumed to support one distinct auction protocol for one distinct service type each, allowing for structured broker discovery processes.

The quantity and distribution of all those agents, thus defining the simulation setting, are specified in a configuration file that is parsed during simulation startup. Additionally, a second file defines all available ST and EST combinations offered in this market. During startup the respective agents are instantiated (the SPs, ISPs and NCs are additionally parameterized with the ST and EST types they support) and located on the topology as stated in the configuration files. In a second step the service description documents are generated and passed to the SPs / ISPs for publication[14].

In order to allow for random demand generation at the SC agents, all STs available in the system are additionally stored at a *DemandHelper* node. During the actual simulation, each SC, once triggered internally by a timeout mechanism, requests a random service request from this helper module. This results in a system where only services

[14]Due to the Java-based implementation these documents are internally represented as Java objects. However, a parser has been developed capable of creating these objects from respective documents coded in XML.

```
<ExtendedSLATemplate>
  <slaTemplateID>alternateOffersTemplate</slaTemplateID>
  <slaTemplate>
    ...
  </slaTemplate>
  <negotiationProtocol>
    <context>
      ...
      <serviceConsumer>
        <maximumNumberOfAgents>
          1
        </maximumNumberOfAgents>
        <minimumNumberOfAgents>
          1
        </minimumNumberOfAgents>
        <admissionRestriction admissionRestrictionForm="open"/>
      </serviceConsumer>
    </context>
    <negotiationObject>
      <negotiableSLO propertyID="sla/price">
        <values>single</values>
      </negotiableSLO>
    </negotiationObject>
```

(a)

```
...
    <process>
      <serviceConsumer>
        <protocolStep>
          <event from="serviceProvider" to="serviceConsumer">
            <messageType>offer</messageType>
          </event>
          <possibleAction from="serviceConsumer"
            to="serviceProvider">
            <messageType>offer</messageType>
          </possibleAction>
          <possibleAction from="serviceConsumer"
            to="serviceProvider">
            <messageType>notification_accept</messageType>
          </possibleAction>
        </protocolStep>
      </serviceConsumer>
    </process>
  </negotiationProtocol>
</ExtendedSLATemplate>
```

(b)

Figure 4.11.: Sample EST Document for the AO Protocol

actually present on the market are demanded in the simulated user requests[15].

Simulation Parameters and their Usage within the Agents

For each simulation run a set of parameters have to be set: The one global parameter is the amount of ticks the simulations should run. This delimits the duration of the experiments. Additionally, there exists a variety of local parameters, applied only within particular agent classes.

Broker Agent (acting as an independent NC) The broker only has one parameter, the timeout that defines the time intervals, after which a matching step is conducted ($matchingTimeout$)[16].

SP / ISP Agent Both SPs and ISPs internally use the following timeout types:

- *findRegistryTimeout*, used during registry discovery.
- *publicationTimeout*, used during the publication process.
- *executionTimeout*, used after the negotiation when the service is actually invoked.

[15] Since the focus of my work lies on the investigation of the adaptability of the developed agents, requests for not available service types would not provide any additional value within the simulations. Hence, they were omitted.

[16] Due to the time-discrete simulations all timeouts are expressed as an amount of ticks that is discounted every tick.

- *auctionTimeout*, used to delimit the time an agent waits for the first joinNegotiation message to arrive before declaring this negotiation attempt to be failed and start over.

When accompanied with a EA, DA or (multi-attributive) AO strategy, a SP also needs a timeout delimiting the time it waits for answers from the SCs during a running negotiation (*biddingTimeout*) before declaring it to be aborted or starting a new round (EA and DA).

Finally, the ISP employs two additional timeouts, the *findNCTimeout* (employed in the broker discovery) and the *negotiationTimeout* (delimiting the maximum time the agent waits for an answer message from the broker).

Apart from the timeouts, a set of parameters defines the way the individual SLOs offered should be treated during the negotiation.

The *estimatedMarketValue* defines the valuation an agent assigns to a given SLO. This value is also used for calculating the starting offers. Depending on the value preference (stated in the *theHigherTheBetter* boolean parameter, also given for each SLO) these starting values are set to double or half the estimatedMarketValue. In an EA for example, given the estimated market value of the service to be sold is 10 USD, the starting value would be set to 5 USD.

The *marketValueAdoption* rate (only given for the price SLO[17]) defines how an agent adapts its market valuation of a given service over time. Whenever an agent was not able to reach an agreement (e.g. when a negotiation was aborted due to elapsing timeouts or if no mutually acceptable offers could be found), it reduces its valuation for the negotiated service. This will improve its chances for winning the next negotiation it engages in. When an agent wins a negotiation, it acts vice versa, trying to get an even better deal the next time. This very simple learning algorithm provides the simulation runs with a dynamic element, also preventing single agents from never being able to reach an agreement, just because of their wrong valuation of a given service.

SPs are also parameterized with a *concessionStep* value (one per SLO), used in bargaining protocols. This defines the value an agent is willing to concede from its last offer in order to converge to a mutually acceptable agreement.

Finally, all SPs and ISPs offer a parameter that delimits the distribution of broad-cast messages, the *maximumHops* factor.

SC Agent On the other hand, the SC agents define the following timeout parameters:

- *discoveryTimeout*, delimiting the amount of ticks the SC can take to find a SI and retrieve the respective EST and ST documents before this request is declared unsuccessful or a fitting SI is chosen to start the negotiation phase.

- *findRegistryTimeout*, used during registry discovery.

[17] Learning is only introduced for the price attribute, as this generates enough dynamic for the simulation runs and the other SLOs are assumed not to be as dynamically adaptable as the actual valuation of a service.

- *joinNegotiationTimeout*, delimiting the time this SC waits for an answer to a joinNegotiation message.
- *executionTimeout*, used during the service invocation.
- *waitForExecutionTimeout*, needed in case of a brokered negotiation (time the SC waits before trying to invoke the service; this guarantees that the ISP has received and processed the notification message from the broker in the meantime).

Finally, the SC is also given a *marketValueAdoptionRate*, *concessionStep* and *reserve values* respectively[18] as well as a maximumHops parameter.

Demonstration Metrics

When assessing the results of a simulation run, three distinct metrics are of primary interest:

- the absolute number of successfully executed negotiations per protocol.
- the minimum and maximum number of successfully executed negotiations per protocol per SC agent.
- the minimum number of different protocols a SC could successfully take part in.

The first metric describes a global view on the simulation experiment, stating how many negotiation processes could successfully be finished for each protocol type, present in the setting (as denoted by a distinct EST).

The second investigates this result on a per-agent basis as it tracks the minimum and maximum amount of successful negotiations per protocol any one SC in the system could achieve. This gives a hint on how the successful negotiations were spread over the agent population.

The final metric shows how diverse the used negotiation protocols were in the view of the individual agents, stating how many different protocols the agents have been involved in.

Experiments and Interpretation

Now the results achieved in simulating various market configurations are presented.

During all experiments a total amount of 10 IL agents was employed in a full meshed network. Since I wanted to investigate the feasibility of run time protocol-generity with as few side effects as possible, I tried to configure the simulation settings in a way they resemble a perfect market. A full-meshed topology especially supports the perfect

[18] These values are provided by the DemandHelper module per request (in form of SLOConstraint objects). However, all received market valuations are internally overwritten with the current market valuation for that service type, as learned throughout several negotiations (given the agent has already accumulated respective knowledge). This way the demand generation is adapted to the experience the agent has already made.

distribution of market information (as coded within the service description documents) among the participants. More complex topologies could in the worst case result in some of the services available not being found and thus negotiated for by the SC agents, leading to a decrease in the total amount of successful negotiation. Hence, full meshed topologies were considered appropriate for the research question at hand.

Each of the connections was parameterized with a bandwidth value ensuring that a complete message of the size used in these experiments traverses one connection in one tick. The simulation itself was delimited to 20.000 ticks and for all experiments 50 SCs were present.

The maximumHop value for all agents was constantly set to 5 hops. The timeouts employed in the respective agent classes were set to the following values:

1. matchingTimeout (NC): 40 ticks
2. findRegistryTimeout (SP, ISP, SC): 10 ticks
3. executionTimeout (SP, ISP, SC): 10 ticks
4. publicationTimeout (SP, ISP): 10 ticks
5. findNCTimeout (ISP): 10 ticks
6. negotiationTimeout (ISP): 50 ticks
7. discoveryTimeout (SC): 30 ticks
8. joinNegotiationTimeout (SC): 10 ticks
9. waitForExecutionTimouet (SC): 5 ticks
10. auctionTimeout (SP): 60 ticks
11. biddingTimeout (SP with EA, DA, AO or MAO): 30 ticks

Throughout the simulation experiments, two SLOs were used, the price and the allocated memory, representing a qualitative service aspect. Most negotiation protocols employed are only single-attributive, thus the price would be sufficient. The multi-attributive AO protocol is the only one also using the second SLO, which was chosen because it exhibits the contrary preference direction than price: whereas SCs want to achieve a low price they will potentially try to negotiate a higher memory amount they can use for their invocations (the preference for SPs is exactly vice versa). This aspect is reflected in the *theHigherTheBetter* parameters set for these characteristics at both SPs / ISPs and SCs. An offer is considered rejectable by a negotiator agent whenever the value of a respective SLO (price or allocated memory) is smaller than 30% of the lower or higher than 170% of the upper reservation value.

The starting values are 10 USD for price and 20 GB for allocatedMemory. Due to the learning algorithm the price value will adapt over time, as already shown. The *marketValueAdaptionRate* was set to 5% (meaning that the valuation will be in-/decreased by

5% of their current value). The concession steps for both SLOs were defined to be 1 USD / GB (SP) and 4 USD / GB (SC), respectively. This reflects the potentially higher urgency on the SC and thus more hesitant concessions on the SP side.

All just mentioned parameters are fixed throughout all experiments. Given the focus on adaptability to new protocols, such strategic aspects are of minor interest, once a reasonable configuration has found. The parameters, which are changed from one experiment to the other, are now shown along with the achieved results.

Overall, six simulation experiments have been conducted. Each was configured with 50 SCs, 25 SPs / ISPs and 1 RA. For each simulation run the three demonstration metrics introduced above have been logged and are given as (bar-)graphs in the following.

Only AO: During the first experiment all 25 SPs offered their services over the AO protocol sketched in subsection 4.2.2[19].

AO and EA: In a second step only 15 SPs used the AO protocol and the other 10 used the EA.

AO, EA and DA: During the third experiment 9 SPs offering the AO and either 8 offering the EA and DA were present.

AO, EA, DA and CM: For the fourth run, one additional agent has been introduced, a broker (NC role). This agent in place, a new configuration of 8 SPs offering the AO, either 6 offering the EA and DA and an additional 5 ISPs (using the brokered CM) was set.

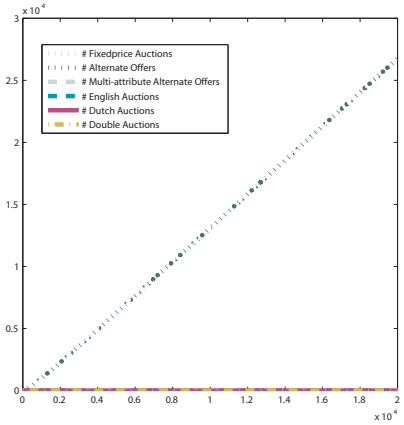
AO, EA, DA, CM and FPA: In the next step, 5 SPs of either AO, EA, DA, CM (ISP) and FPA were present.

AO, EA, DA, CM, FPA and MAO: During the final experiment 5 ISPs and 4 SPs for each of the other protocols (AO, MAO, EA, DA and FPA) have been configured, again summing up to 25 just as with all other experimental settings.

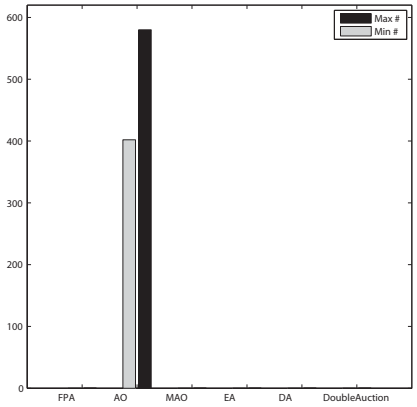
The actual values for the result metrics are of secondary significance. Among others, the specifics of the different protocols (for example the usage of bidding rounds with pre-defined durations in contrast to continuous offer exchanges), the random selection of SI documents to start a negotiation with, or the dynamic market valuations determine how many negotiations of a given type are actually finished.

The main statement to be proven with these experiments is that the GN node is at all able to adapt to different protocols, only based on their description in the EST documents (proof-of-concept). Based on the results achieved, this assertion can be approved. With the described escalation of different negotiation protocol types present throughout the experiments, each of the newly introduced protocols has successfully been integrated in

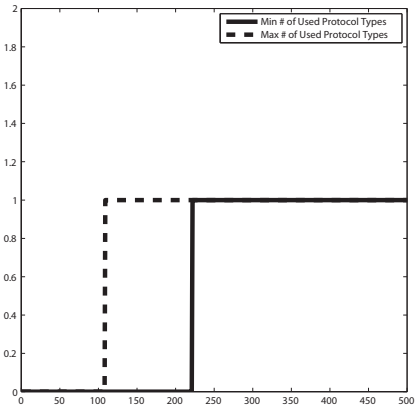
[19]Throughout all simulation experiments the six protocols introduced in subsection 4.2.2 have been used.



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol



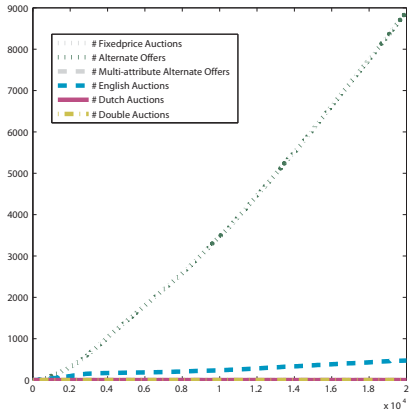
(c) Range of used Protocol Types

Figure 4.12.: Results of using the following Protocols: AO

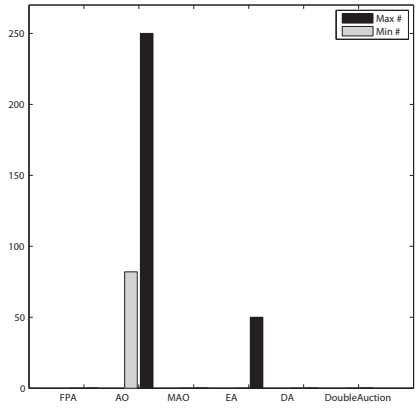
the actual market behavior; each protocol type present has been executed by a significant number of agents.

Given the relatively small setting, it is not surprising, that not all experiments resulted in every agent having been able to win a negotiation in more than one type of protocol (again also affected by the random selection of SIs and thus negotiation protocols for each service request). During the last experiment (investigating a setting with all six different protocols present) however, each agent could successfully take part in at least two negotiation protocol types (see figure 4.17(c)).

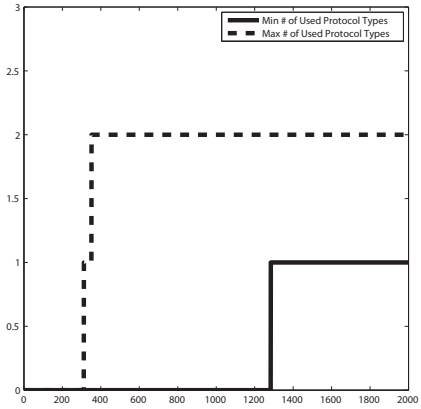
On the other hand, in each simulation run there exist SC agents that have successfully taken part in all available protocols. This fact is shown in figures 4.12(c), 4.13(c), 4.14(c), 4.15(c), 4.16(c) and 4.17(c) respectively, where the maximum amount of different protocol types a single agent has been able to take part in is always the amount of



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol



(c) Range of used Protocol Types

Figure 4.13.: Results of using the following Protocols: AO and EA

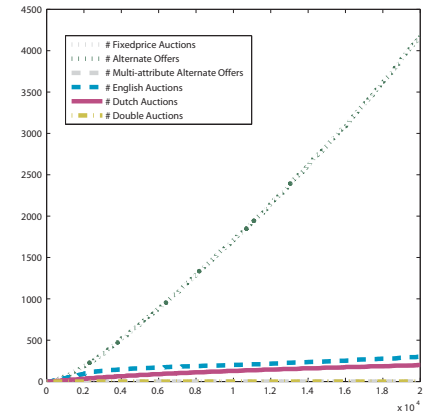
protocols at all present in the market.

Having now shown the adaptability of the GN module to the introduced negotiation protocols (and respective description documents), the requirements stated for this thesis can be considered fulfilled and the demonstration step thus completed. The subsequent evaluation step is described in the next section.

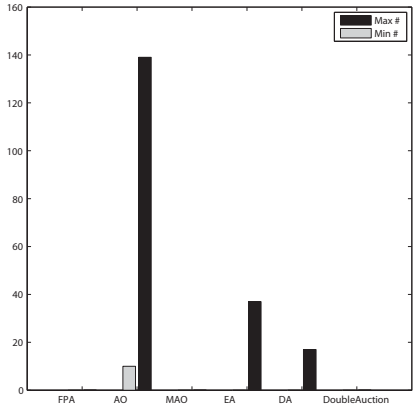
4.4. Evaluation based on Meffert's Theory on Flexibility

Following the rationale of Hevner et al. (2004) and Peffers et al. (2008) (see section 1.3 for a detailed discussion), the evaluation of a scientific artifact comprises its assessment with regard to a set of quality aspects.

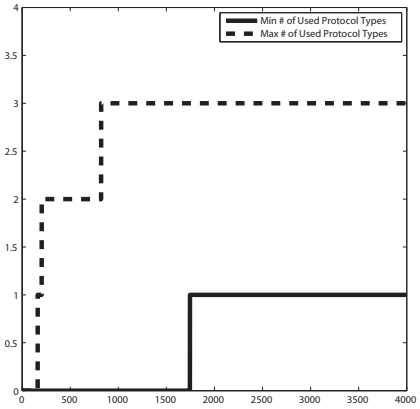
To this end, Hevner et al. do not distinguish between a demonstration and an eval-



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol

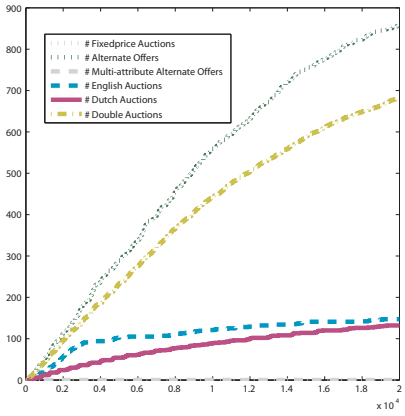


(c) Range of used Protocol Types

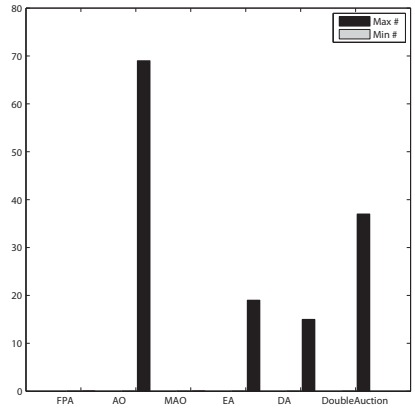
Figure 4.14.: Results of using the following Protocols: AO, EA and DA

uation step; they basically claim the need for a structured assessment of the designed solution in terms of its effectiveness of solving the research problem, the quality of the solution and the utility gain achievable by using the respective artifact in the envisioned problem setting. Peffers et al., on the other hand, distinguish between a basic demonstration phase, during which the effectiveness of the designed solution and an actual evaluation phase during which the quality of this solution is assessed.

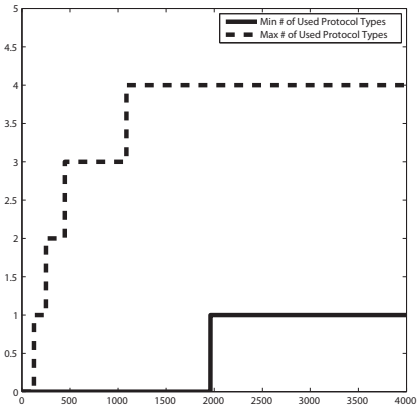
In this thesis, I mainly build on the rationale of Peffers et al., as it provides a very structured assessment framework. However, it also results in some inaccuracies when actually executed: The authors claim that demonstrating an artifact simply means to "use [the artifact] in experimentation, simulation, a case study, proof, or other appropriate activity" (Peffers et al. 2008, p. 90). All assessments on the basis of measurable quality metrics as well as comparisons with the stated requirements are assumed to be-



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol

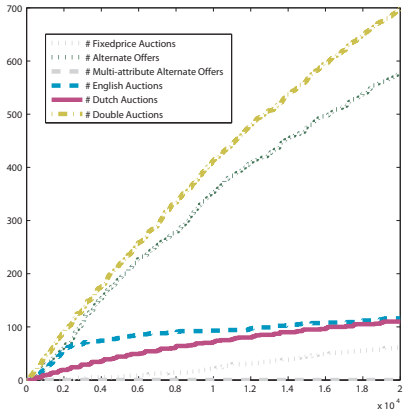


(c) Range of used Protocol Types

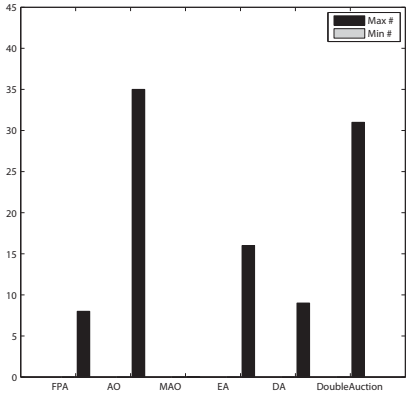
Figure 4.15.: Results of using the following Protocols: AO, EA, DA and CM (double auction)

long to the evaluation phase. In my opinion this contradicts with their initial statement of evaluation being the means to assess the quality of a designed solution, after having shown that it solves the stated problem at all (effectiveness) in the demonstration step.

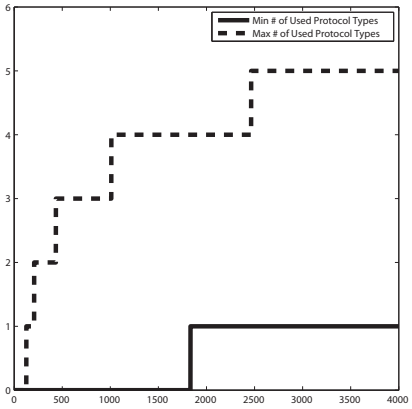
Throughout this thesis, this assertion has been used as a conceptual basis for distinguishing which assessment steps are shown in which section (demonstration or evaluation): During the last section the capability of my infrastructure to solve the stated requirements has been demonstrated. In addition to using the prototype system in a simulated environment, some quality aspects have also been assessed as a comparison was made with the stated requirements. All these steps aimed at proving the effectiveness of the BabelNEG approach. In the following I will consequently elaborate on "how well [it] supports a solution to the problem" (Peffers et al. 2008, p. 92).



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol



(c) Range of used Protocol Types

Figure 4.16.: Results of using the following Protocols: AO, EA, DA, CM (double auction) and FPA

A fundamental problem, occurring during the evaluation of a proof-of-concept prototype system, is that due to its innovative nature there are no other systems, solving the same problem, one could directly compare it to. It is by definition the first prototype aiming at this specific problem. Thus, an evaluation step must build on a unilateral assessment of the solution's quality.

To this end, a theory proposed by Meffert (1985) can be employed. In this work, the author elaborates on *flexibility* as a fundamental management paradigm[20]. He identifies three types of flexibility, with which a company can counter external events, such as changes in the market or demand structures:

[20]Flexibility is also often referred to as a distinctive advantage of IT systems building on software agents, when compared to traditional ones (Kirn 2006).

- *Processual Flexibility*: Speed of action in terms of planning and realization.
- *Structural Flexibility*: Capability of organizational structures, personnel and management to adjust to new market situations.
- *Activity Flexibility*: Set of possible actions making up the room of maneuver for the company.

For all three types, a high degree of achievement is desirable, as it strengthens the company's market position and thus its long term business value. For my work, especially the last type is of interest, as it describes how many options a given enterprise has when acting on the market. Transferred to the IoS scenario, and the research problem stated for this thesis, it refers to the amount of potential transaction partners to be found on the service market. From a reverse perspective, this type of flexibility quantifies the degree to which the technological and policy infrastructure of this service market restricts the involved providers and consumers in their economic behavior.

In the vocabulary of Meffert's theory, the goal of my dissertation is thus to increase the activity flexibility of IoS participants, in that it enlarges the set of potential business partners by reducing restrictions posed by the market infrastructure (particularly on the applied negotiation protocol)[21]. To this end, the quality of my infrastructure denotes the degree to which this activity flexibility could be increased (see figure 4.18).

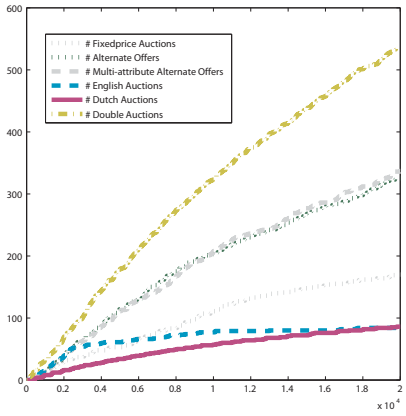
The theoretical optimum of activity flexibility, regarding the stated research problem, would thus be that each SP is able to negotiate with every SP potentially present in the IoS and, consequently, can take part in every possible negotiation protocol[22]. As already stated, an exhaustive list of possible negotiation protocols does not exist, so this optimum remains a theoretical one.

Nevertheless, it provides a reference for assessing the developed system. As already illustrated, the protocol-generic negotiation strategy is able to process, and adapt to, all negotiation protocols, that can be described within an EST document. The achieved activity flexibility is thus dependent on this document's capability to capture all relevant information.

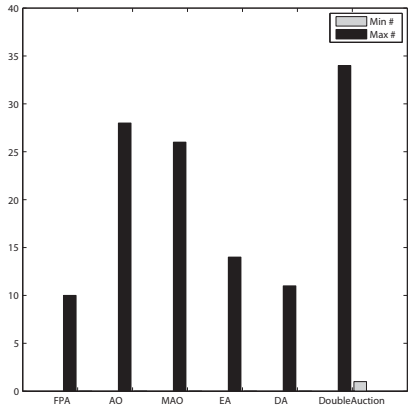
A very high quality of expressiveness could be achieved by building on state automata, a concept widely used to describe communication protocols. The events and actions used therein have been identified to be the negotiation related messages. After a thorough literature analysis this set could be proven to be exhaustive for the majority of actually used negotiation protocols in electronic settings. Up to now, no protocol was found, which could not be mapped to the identified set of messages. Summarizing, the flexibility achieved in describing the actual negotiation process can be viewed as significantly high. If, in the future, new negotiation messages or other types of events become necessary,

[21] Without my system in place, every SC can only negotiate with the subset of SPs, offering the adequate negotiation protocol; my system tries to dispose this restriction posed on the selection of transaction partners.

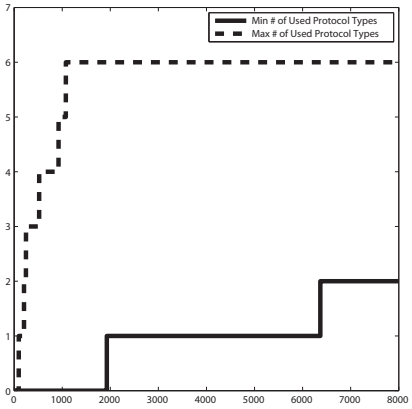
[22] This would be the case in unsupported negotiations between humans, where the two negotiators can talk to each other and resolve all occurring ambiguities regarding the negotiation and agreement process.



(a) Amount of successful Negotiations per Protocol



(b) Min / Max Amount of successful Negotiations per Protocol



(c) Range of used Protocol Types

Figure 4.17.: Results of using the following Protocols: AO, EA, DA, CM (double auction), FPA and MAO

these could even be integrated into my language quite easily by simply adding them to the respective EST schema.

The elements within my description language, best suitable for identifying the distance to the theoretical optimum activity flexibility are the ones for which external rule languages are needed. These elements represent placeholders for specific restrictions of the negotiation protocol, which cannot be described with simple parameters. The very feature of allowing expressive rule languages within these parameters can, however, lead to incompatibilities between the SPs and SCs whenever the used languages are not known by both sides. More generally, it can be stated that the developed infrastructure provides the more flexibility the less external rule languages it employs when describing a protocol or the more commonly known, and thus understandable, the used rule

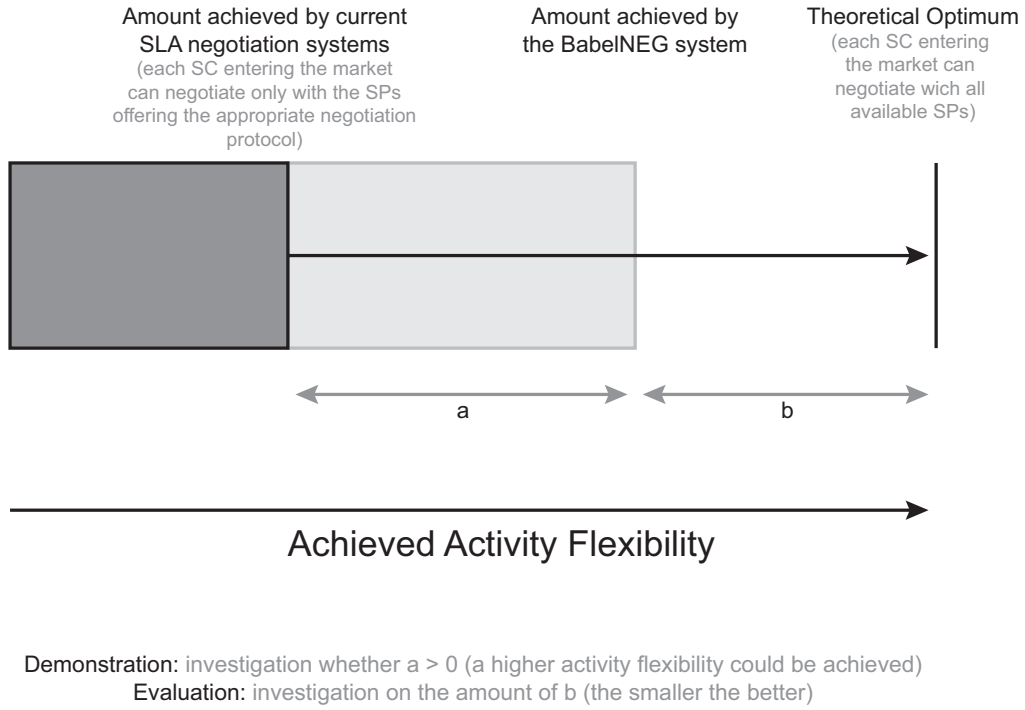


Figure 4.18.: Evaluation based on Meffert's Activity Flexibility Concept

language is.

The second aspect with a potential to limit the achieved activity flexibility is the ontology, needed as a conceptual basis for my infrastructure. Only if the agents understand what SLOs they are negotiating about, they can do so. Thus, the vocabulary used during the negotiation has a crucial effect on the activity flexibility that can be reached with my system. Only if a comprehensive ontology of terms is known, or at least accessible for all participants of the market, a high flexibility can be achieved. The more comprehensive the underlying ontology is, the higher the achievable flexibility.

Hence, the developed system can be attested a high degree of activity flexibility, and thus a high quality of the solution with regard to its evaluation, given a restrictive use of preferably common-use rule languages (or no such use at all) and a comprehensive system of underlying ontologies.

During the demonstration step a set of six different negotiation protocols of very differing nature have been employed, none of which needed any external rule expressions to be described. It can thus be assumed that the first restriction is fulfilled to a very high degree. Even if needed, a well-known rule language (several of such are in use at present) would be able to reduce the implied flexibility loss to a minimum.

On the other hand, an industry-wide ontology (or at least one covering a particular industry domain) can also be assumed to be present, given the many research work done in that area (see for example W3C's Semantic Web initiative[23], in particular the ongoing work on the OWL Web Ontology Language[24]) and the inherent need for a common vocabulary amongst business partners. Therefore, this second restriction can also be regarded as insignificant with regard to the achieved flexibility of the BabelNEG framework.

4.5. Assessment of the Communication Overhead Introduced by the BabelNEG System

From another perspective, my system could be compared with traditional SLA negotiation systems only offering a single, fixed negotiation protocol. In the light of (Christensen 1997) the BabelNEG system could thus be seen as a disruptive technology, fulfilling the same functionality than traditional ones (it also allows to electronically negotiate SLAs with a distinct negotiation protocol), but also offering some additional features (in this case the possibility to adapt to new protocols at run time).

In comparing it with traditional infrastructures, one would have to measure its performance in a relevant metric. For negotiation systems one such metric is the amount of communication traffic (negotiation messages) needed until a negotiation is reached.

When comparing my system with one exclusively using one distinct protocol, the discovery and negotiation phases are best compared separately:

During the negotiation phase the amount of sent messages heavily depends on the protocol applied and the strategies of the involved negotiators. An AO protocol between two agents with very small concession steps could for example last quite long (many messages would have to be exchanged until a both agents have conceded to an agreement acceptable for both). Contrarily, two agents being matched by the broker in a CM would in total produce exactly four messages (both would have sent an *offer* / *offerToSell* and received a notification message).

To this end, let $N_{p_r s_r}$ be the amount of messages needed to reach an agreement in a reference SLA negotiation system r, configured with a protocol p_r and set of employed strategies s_r. Consequently $N_{p_b s_b}$ denotes the amount of necessary messages within the BabelNEG system (employing protocol p_b and strategies s_b).

If one would now configure the BabelNEG system with the same negotiation protocol p as the infrastructure it is compared to and equip the involved agents with the exact same strategies s in both systems, BabeLNEG would perfectly mimic the other system with regard to the negotiation phase. Given the EST document can capture all communication-relevant aspects of the applied protocol (which was demonstrated in section 4.2), the involved agents would use the same communication procedures as in the reference system. When also equipped with the same strategies (the second parameter influencing the produced communication traffic) the exact same amount of messages

[23] http://www.w3.org/standards/semanticweb/
[24] http://www.w3.org/standards/techs/owl

would be needed in a given negotiation.

More formally: $p_r = p_b \bigwedge s_r = s_b \Rightarrow N_{p_r s_r} = N_{p_b s_b}$.

During the discovery phase on the other hand, my system introduces additional communication traffic, as the agents need to look up the various description documents, which would possibly not be mandatory in a system employing only a single protocol.

In order to quantify this communication overhead, let D_r be the number of messages needed for a SC in the reference system to locate a given service instance, to start a negotiation with. On the other hand D_b denotes the amount of messages produced within BabelNEG for the same task. For both Systems I will now conduct an analysis of both the best and worst case scenario[25].

The best case for the reference system denotes a situation, in which a service instance of the requested type is already known to the requesting SC. In that case it directly contacts this instance and starts a negotiation. D_{r_b} (denoting the amount of messages needed during the discovery phase in the reference system in the best case scenario) is thus 0. In the worst case on the other hand, the SC only knows the service type of which an actual instance is needed. To retrieve such an instance a request message is sent to the registry, which in turn answers with a respective result message. Thus, the amount of discovery messages produced in the worst case D_{r_w} is 2.

Within the BabelNEG, the best possible scenario occurs similarly whenever the SC already knows a SI along with its respective EST document. In this case it can also simply contact the SP stated in the SI and start a negotiation with it. Thus, $D_{b_b} = 0$; exactly like with the reference system.

In the worst case however, the SC within BabelNEG does only know the ST for which it requests an actual instance; no EST is known beforehand. In the current system implementation an initial request is sent to the registry in such a case, requesting SIs fitting the searched type. After that, the SC requests the EST document for each of the such returned SIs, choses one and starts a negotiation with it. Given there are n SIs known at the registry (and all of them are consequently mentioned to the SC by the registry) this means that $2 * n$ messages are produced; a request and a response message for all SIs found. Incorporating the first communication with the registry (concerning the available SIs) this results in a total amount or communication traffic of $D_{b_w} = 2 * n + 2$.

During an average discovery process this number would regularly be lower since the agents store retrieved EST documents and will thus already know some of the needed ESTs for in the future retrieved SIs. Also there are several potential optimization steps, such as sequentially requesting ESTs instead of in parallel or requesting several ESTs within one message; these mechanisms would be incorporated in the system once it matures from a proof-of-concept to an actual software product for everyday use.

Given the average amount of messages needed in most of the negotiation protocols (especially in AO and EA / DA protocols) and the very low probability of the sketched worst case occurring (most of the time at least one of the ST or EST documents would

[25]For this analysis I assume that a registry node is already known to the requesting agent (the traffic produced during registry discovery would depend on the applied discovery architecture, which is orthogonal to the analysis done now). Also I assume that at least one service instance of the requested type is available and open for negotiation.

already be known to the requesting SC), the communication overhead implied by my system can be assumed marginal when compared to the drastically increased activity flexibility.

In this chapter the BabelNEG system (a prototype implementation of the mechanisms presented in chapter 3) has been assessed with regard to its effectiveness and efficiency. First, its Java-based implementation in the context of the SimIS toolkit has been shown. After that the capability of the developed description documents to capture all relevant service aspects in a machine processable way has been demonstrated. In order to provide assertions on whether BabelNEG is able to fulfill all stated requirements a comprehensive conceptual and simulative demonstration has been conducted, the results of which have been shown in sections 4.2 and 4.3. The chapter closes with an investigation of the efficiency of the proposed solution, based on the limitations of the developed description documents and the communication overhead introduced by BabelNEG.

Summarizing the results, the developed prototype has been able to fulfill all requirements stated in section 2.1. The quality of the proposed solution is also considerably high, as shown in the light of Meffert's flexibility theory. In the remaining chapter I provide a summary of the results achieved and some remarks on future work and ways to implement the developed ideas into a commercial software product.

5. Lessons Learned and Future Steps

5.1. Summary and high-level Interpretation of Results

During my dissertation project I developed a novel service infrastructure for the structured discovery and protocol-generic negotiation of electronic SLA documents.

The need for such an artifact has been deduced from a detailed scenario analysis, extrapolating past and current developments in distributed computing. This step has resulted in a conceptual model for the future IoS, acting as the problem domain for the remainder of my work. Based on economic theory, the need for both negotiation processes as such and the possibility to adapt to different protocols at run time has been inferred.

After having motivated my work that way, the research process applied throughout this project has been defined. A DS method has been chosen, due to the research question at hand. In that, I followed the methodological guidelines proposed by Hevner et al. (2004), Peffers et al. (2008) as well as Gregor and Jones (2007); them being the most influential works on DS research methodologies as of today.

Following these guidelines, the requirements for my prototype, being deduced from the scenario model and underlying theories, have been identified. These claims are further used as criteria for the assessment of my system.

I have subsequently discussed conceptual foundations for my work, comprising theoretical principles for the design, formulation, discovery, negotiation and subsequent usage of (electronic) SLAs, especially focusing on distributed infrastructures of electronic services. Related research efforts, targeting the research problem, as stated for my dissertation, have been identified next and subsequently described with special focus on their potential to solve the stated problem. This has been assessed by comparing each individual project with the identified requirements.

In chapter 3, the actual infrastructure design, thus the proposed solution to the stated research problem, has been presented. The underlying idea is to decouple the good to be sold (the SLA) from the negotiation protocol, thus enabling a SP to apply different negotiation protocols for the same service over time (by simply creating new combinations of SLA and protocol). On the other hand, for the SC side a protocol-generic negotiation component has been designed, capable of adapting to different protocols, as offered by the SPs. The conceptual copula between these two sides are a set of structured service description documents, defining not only the service-relevant functional and non-functional parameters, but also the applied negotiation protocol in a machine-readable way.

Throughout the last chapter, a thorough assessment of the infrastructure design has been done, including the actual implementation of the developed mechanisms and data

structures in a Java-based simulation environment, the conceptual demonstration of the system's effectiveness (with regard to the stated requirements) as well as a simulative demonstration step proving the adaptability of the SC agents.

Based on these assessment steps the following results can be stated:

- The expressiveness of the designed data structures could be shown on the basis of a set of representative negotiation protocols originating in scientific literature.
- During the simulation runs the developed service agents have been able to adapt to formerly unknown negotiation protocols, just by parsing these protocol descriptions.

Hence, the developed system is able to fulfill the stated requirements, thus proving its effectiveness to solve the stated research problem.

5.2. Critical Reflection on the Applied Research Method

As with every research project, my dissertation has several points which could be criticized methodically. The first is probably the fact that I set out to solve a scientific problem not actually present currently; the scenario anticipated is not yet existent. This inevitably leads to uncertainties regarding the deduced research problem and ultimately the motivation of my work in general.

Nevertheless, many innovative artifacts face this same problem: they anticipate future scenarios towards which they are geared and aim at solving the problems occurring therein. The designers can only try to, as accurately as possible, make a prediction on how the future will probably look like and base their work on this scenario. I followed this exact procedure. What I hope makes my prediction valid, is its conceptual grounding in ongoing trends within the distributed computing discipline, as can be witnessed in both academia and industry today.

However, there is still a chance that the future Internet will not look exactly like the IoS setting sketched in section 1.1.1. In that case, hopefully at least some aspects of my scenario model will be true (which is quite reasonable to assume), potentially leading to only minor losses in the utility gained by my infrastructure or consequently minor needs for changes to it.

The second point, which could be criticized is the assessment of the developed artifact. The demonstration of the effectiveness of the developed description language and respective documents cannot claim completeness. It was based on a set of different protocols that were chosen due to a mutually as high as possible dissimilarity. For a complete demonstration this assessment would have to be done for all possible negotiation protocols used today. Unfortunately, there is no such exhaustive list, as already stated. The conceptual demonstration as present now could potentially increase its claim on completeness by gathering negotiation protocols from all current and future service infrastructures (originating in industry and research) and investigate the language's expressiveness with those. In my work, a reasonable amount of (research) projects and

protocols proposed in scientific literature has been used; paying tribute to the time constraints a dissertation project poses. A more comprehensive evaluation will have to take place over the next years, in which the developed infrastructure is hopefully used extensively.

On the other hand, simulation as a tool for the demonstration of a proof-of-concept prototype is arguable. The underlying model and simulation tools always only "simulate" some real-world setting without a proof that they do it correctly. This is enough for a proof of feasibility, as aimed at with this thesis. More sophisticated assertions on technical quality metrics, such as scalability or robustness, demand an actual, physically distributed prototype system. Only then, a reliable assessment of the technical quality of the developed infrastructure can be made.

Finally, my dissertation project suffers from it being as innovative as it is. No other infrastructure solves the exact same problem stated for my work, thus resulting in a lack of alternatives the BabelNEG system could be directly compared to. As of today the one promising candidate for such a direct competitor project is SLA@SOI[1], which is currently work in progress. Once finished it would be interesting to compare it with my infrastructure, especially with regard to the expressiveness of the protocol descriptions and thus the resulting activity flexibility achieved by both.

5.3. From Prototype to Product

On an organizational level, a necessary precondition for an application of my ideas is for a respective company's IS to follow the SO paradigm. Only when strictly building on loosely coupled, fine-granular IT services, buying, negotiating about, combining and re-selling such computational functionalities is reasonable.

Additionally, the economic strategies applied in the market behavior of the management agents must be defined. A detailed specification of concession step values, reserve prices etc. is needed for each agent and potentially even each service to be sold / purchased.

With regard to the binding of automatically negotiated "contracts", a juristical framework (e.g. in terms of a framework contract governing a consortium of involved business partners) is needed as a common ground for all actors on the electronic market. Given such a foundation is present, the negotiation agents can be regarded as representatives of their host organizations, enabling them to bindingly act in the resulting service economy.

The vocabulary used in the SLAs and service description documents must be understandable for all market participants for an automated economy to work. Thus, a ontology of service concepts (such as quality metrics or functional descriptions) must be defined, as already stated in the last chapters. It should cover all market-relevant information concepts along with the respective semantics of such concepts.

On a technical level, some fundamental infrastructure components are needed within the company IS to successfully integrate with my system: A graphical user interface (for requesting or selling services on the market manually) or an equivalently extended

[1] http://sla-at-soi.eu/

orchestration engine (for doing so automatically) are needed as links between the traditional service infrastructures and the BabelNEG system.

Also, a distributed infrastructure of market brokers and discovery nodes is needed to assure a stable service economy. These nodes could either be maintained by the SC and SP companies or by third party intermediaries. Similarly, a technical security system is indispensable, especially for transactions in a business context.

Given these preconditions are met within the participating companies and general market infrastructure, the negotiation infrastructure described in chapter 3, and the resulting prototypical implementation as has been assessed in chapter 4, can be directly integrated with the currently used company ISs. For this, the defined management agents and respective strategy modules must be implemented and granted access to both the internal service instances (for guarantee enforcement and monitoring purposes) and the external communication channels (including security and discovery system).

For a comprehensive service management infrastructure, software components supporting the other phases of the life cycle (apart from discovery and negotiation) are of course also needed.

Given these conditions are met the BabelNEG system can provide significant economic advantages for both individual companies and the digital service economy as a whole.

It drastically reduces entry barriers of the IoS, especially from the SC's point of view, as respective enterprises can easily implement SC agents instantly capable of interacting with all possible SPs present in the service economy. There is no need to implement different negotiator agents for different service markets (and thus negotiation protocols). In some cases this would be too costly or inefficient, requesting from the SC company to choose the protocols respective agents are implemented for. Given the high dynamic of the IoS setting, the decision about such a self-restriction to a subset of the available SPs as potential transaction partners is not only a highly complex task, but could also likely render itself wrong over time. With the BabelNEG system in place, no such decision has to be made and SCs can flexibly adapt to new SPs and negotiation protocols.

This is especially interesting for small and medium sized companies acting on niche markets. They generally do not have the funds to constantly re-evaluate their decisions on potential markets to engage in and to re-design their negotiation components. Nevertheless they can deliberately profit from an IoS scenario as enabled by the BabelNEG infrastructure, as it allows them to purchase needed services (even on very short notice) and concentrate the majority of their in-house investments on their core business instead.

The BabelNEG system can also lead to completely new service products to be created very quickly by combining individual basic services purchased over the IoS. No risky investments on in-house IT infrastructures have to be made for such a new offering, but the providing company can "try out" the market success at first with external services before deciding to internally providing them in case of success.

Given the BabelNEG system is in place, SPs can also easily adjust the employed negotiation protocols to changes in the market configuration or their offered services. Once introduced and described within a respective EST document, they can be sure to assume SCs able to interpret it and engage in respective negotiations subsequently.

On the economy's point of view, the BabelNEG system enables a set of new business

models to emerge within the IoS, such as market makers, trusted third parties or even providers of best practice negotiation strategies on demand. Given the much higher amount of compatible agents in a IoS setting based on BabelNEG, there is a much higher possibility of exploiting economies of scale from such businesses. This is especially the case in a digital setting, since a respective service, once implemented, produces almost no incremental costs in case the amount of users increases, but contrarily a very high amount of marginal gains.

5.4. Future Work

Given the dynamic IoS setting, an ever changing application scenario can be foreseen for the BabelNEG infrastructure. In this last section, I will shortly sketch what adaptions / extensions to the current system design could be introduced in order to increase its overall adaptability to such new application settings.

The most fundamental dynamics, which can be anticipated for the future IoS, are a changing set of services and service metrics; especially in case of an ongoing trend to automate individual business functionalities and the emergence of juristical foundations for cross-organizational automated service transactions. This development can already be captured in the current system architecture, as it only affects the service vocabulary used within the negotiations. The actual agent roles, messages and negotiation protocols in general do not differ just because the negotiated metrics change.

Similarly, optimization of negotiation strategies with regard to a) individual negotiation protocols or even b) when introducing strategies optimized for a whole set of different protocols (which would be favored by a protocol-generic approach as presented in this thesis) does not affect the system architecture and communication mechanisms designed therein.

If more sophisticated negotiation protocols emerge, an extension of the description documents proposed in this thesis could become necessary. A potential adjustment, increasing their expressiveness, could be the introduction of sub-states within the negotiation process. This would result in an altered process-element within the EST document; not only the incoming event will determine the possible actions, but rather a tuple of incoming event and current state. This results in a direct equivalence to the state machine paradigm.

Apart from these dynamics of the underlying scenario, some fundamental extensions could be introduced, increasing the general efficiency of BabelNEG:

The first potential extension would be the introduction of a dedicated *Information Service* role, responsible for distributing negotiation-relevant data[2]. Which data can be accessed can already be described within the EST document and in the current infrastructure proposal the NC is assumed to distribute such information. However, a comprehensive support for information distribution, potentially allowing for protocols,

[2]In the current prototype, this aspect was considered out of scope, as the primary goal was to investigate the adaptability of the SCs to new protocols.

in which the negotiating agents have to actively request some information, on e.g. the current highest bid during their negotiation behavior, is still to be designed.

Also, a *publish / subscribe system* could be another interesting extension to my system. It could build on the already used query data structure (used for querying SIs, ESTs or STs from the RA at the moment) and would allow a SC agent to post the need for a service of a given type (and / or offering a given protocol) whenever no such service could be found right away. To this end, it would define the search criteria, create a respective query object and post this *interest* to a RA. Whenever a SI fitting the stated query is published at the RA, a *notification* message is sent to the respective SC, indicating that an instance of the required service is finally available on the market.

In order to assure timeliness of the stored interests a lease-based mechanism could be employed. Each interest, registered with the RA, must be renewed after certain time intervals. Whenever a lease is not renewed in time the respective interest is deleted from the publish / subscribe infrastructure.

A minor extension to the current system could also allow for the integration of reverse negotiations (and thus eventually for protocol-generity, or the need for such, at the SP side). Instead of publishing SI documents, representing a service offered by a SP, the SC agents could offer an extended SI document representing a *demand* for a service, the respective SC currently needs. This demand document would exhibit the same internal structure as an SI. However, no link to a WSDL document would be present, as the actual service implementation fulfilling this demand (if any of such is ever found), is not known at publication time. Additionally, the EST to be applied for this demand is only optional. If existent it denotes that the service to be found must not only fit the stated ST but also adhere to the stated EST (i.e. a distinct protocol is requested), if not, only the type of the service is specified as a criteria in the demand.

Especially the discovery architecture is very basic in the current version of the system. This is due to the already mentioned pragmatic reasons. Future versions should incorporate a more sophisticated and thus more robust and scalable discovery mechanism than a single registry node. The most promising approach for this is probably a P2P-based architecture, given such systems excel in scalability and resilience to node failures (especially structured ones).

One idea to integrate the current prototype with a P2P-based discovery architecture is to introduce a DHT which connects all registry nodes and in which all service description documents are stored. Whenever a service request is submitted to a registry node (node within the DHT ring), the respective documents are retrieved and returned to the requestor.

A new RA can simply join this system by discovering a DHT node already present (broadcast discovery) and then join the DHT ring as a neighbor of this node. Internally, this results in the re-distribution of the stored data and routing information. In contrast, the absence of a RA (which has left the DHT) is directly noticed by its neighbor, which then triggers the re-arrangement of the DHT data accordingly (this represents a standard mechanism for DHTs).

All of these extensions could help to increase the efficiency and overall functionality of the developed system. Nevertheless, the prototype as currently available already

demonstrates the feasibility of automated protocol-generity in SLA negotiations, which was the primary goal of this thesis.

A. Appendices

A.1. Service Description Schema Documents

A.1.1. Service Type Document

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://xml.SLANeg.org/schema/SLANeg"
            xmlns="http://xml.SLANeg.org/schema/SLANeg"
            elementFormDefault="qualified">

    <xsd:complexType name="PropertiesType">
        <xsd:sequence>
            <!-- value domain of the property -->
            <xsd:element name="domain" type="xsd:String" />

            <!-- description of the metric's semantics -->
            <xsd:element name="declaration" type="xsd:String"
                minOccurs="0" maxOccurs="1" />
        </xsd:sequence>

        <!-- metric identifier as uniquely defined in an industry-
            wide ontology -->
        <xsd:attribute name="propertyID" type="xsd:anyURI" />
    </xsd:complexType>

    <!-- defines the functionality of a class of services -->
    <xsd:element name="ServiceType">
        <xsd:complexType>
            <xsd:sequence>
                <!-- qualified name of the service type -->
                <xsd:element name="serviceTypeID" type="xsd:anyURI
                    " />

                <!-- UDDI-like description of the service type (
                    URL or in-line description) -->
                <xsd:element name="serviceDescription" type="
                    xsd:anyType" />

                <!-- non-functional service properties -->
                <xsd:element name="property" type="PropertiesType"
                     maxOccurs="unbounded" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
```

```
36 </xsd:schema>
```

A.1.2. Extended SLA Template Document

```
1  ?<?xml version="1.0" encoding="UTF-8"?>
2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3              xmlns ="http://xml.SLANeg.org/schema/SLANeg"
4              targetNamespace="http://xml.SLANeg.org/schema/SLANeg"
5              elementFormDefault="qualified">
6
7      <!-- SLA template -->
8      <xsd:complexType name="SLATemplateType">
9          <xsd:sequence>
10             <xsd:element name="SLO" type="SLOType" minOccurs="0"
                   maxOccurs="unbounded"/>
11         </xsd:sequence>
12     </xsd:complexType>
13
14     <xsd:complexType name="SLOType">
15         <xsd:sequence>
16             <!-- (initial) value for this SLO -->
17             <xsd:element name="value" type="xsd:anySimpleType" />
18         </xsd:sequence>
19
20         <!-- ID of the service property this SLO is defined upon (
               originates in the respective ServiceType) -->
21         <xsd:attribute name="propertyID" type="xsd:anyURI" />
22     </xsd:complexType>
23
24     <!-- negotiation protocol description -->
25     <xsd:simpleType name="AdmissionRestrictionFormType">
26         <xsd:restriction base="xsd:string">
27             <xsd:enumeration value="open"/>
28             <xsd:enumeration value="restricted"/>
29         </xsd:restriction>
30     </xsd:simpleType>
31
32     <xsd:complexType name="AdmissionType">
33         <xsd:sequence>
34             <!--this element is present if restricted admission is
                   defined-->
35             <xsd:element name="admissionRestrictionRule" type="
                   xsd:anyType" minOccurs="0" maxOccurs="1"/>
36         </xsd:sequence>
37
38         <xsd:attribute name="admissionRestrictionForm" type="
               AdmissionRestrictionFormType"/>
39     </xsd:complexType>
40
41     <xsd:complexType name="RoleContextType" mixed="true">
42         <xsd:sequence>
43             <xsd:element name="maximumNumberOfAgents" type="
                   xsd:integer" minOccurs="0" maxOccurs="1"/>
44
45             <xsd:element name="minimumNumberOfAgents" type="
                   xsd:integer" minOccurs="0" maxOccurs="1"/>
```

```
46
47              <xsd:element name="admissionRestriction" type="
                    AdmissionType"/>
48          </xsd:sequence>
49      </xsd:complexType>
50
51      <xsd:complexType name="ContextType">
52          <xsd:sequence>
53              <xsd:element name="serviceProvider" type="
                    RoleContextType"/>
54
55              <xsd:element name="serviceConsumer" type="
                    RoleContextType"/>
56          </xsd:sequence>
57      </xsd:complexType>
58
59      <xsd:simpleType name="ValuesType">
60          <xsd:restriction base="xsd:string">
61              <xsd:enumeration value="single"/>
62              <xsd:enumeration value="multiple"/>
63          </xsd:restriction>
64      </xsd:simpleType>
65
66      <xsd:complexType name="NegotiableSLOType">
67          <xsd:sequence>
68              <xsd:element name="values" type="ValuesType" />
69          </xsd:sequence>
70
71          <!-- ID of the service property the SLO is defined upon --
                >
72          <xsd:attribute name="propertyID" type="xsd:anyURI" />
73      </xsd:complexType>
74
75      <xsd:complexType name="NegotiationObjectType">
76          <xsd:sequence>
77              <xsd:element name="negotiableSLO" type="
                    NegotiableSLOType" minOccurs="0" maxOccurs="
                    unbounded" />
78          </xsd:sequence>
79      </xsd:complexType>
80
81      <xsd:simpleType name="ProgressFormType">
82          <xsd:restriction base="xsd:string">
83              <xsd:enumeration value="ascending"/>
84              <xsd:enumeration value="descending"/>
85          </xsd:restriction>
86      </xsd:simpleType>
87
88      <xsd:complexType name="ProgressType" mixed="true">
89          <xsd:sequence>
90              <xsd:element name="progressForm" type="
                    ProgressFormType"/>
91
92              <xsd:element name="delta" type="xsd:anySimpleType"
```

```
                minOccurs="0" maxOccurs="1"/>
93          </xsd:sequence>
94      </xsd:complexType>
95
96      <xsd:complexType name="ThresholdType">
97          <xsd:choice>
98              <xsd:element name="lowerBound" type="xsd:anySimpleType
                    "/>
99
100             <xsd:element name="upperBound" type="xsd:anySimpleType
                    "/>
101         </xsd:choice>
102     </xsd:complexType>
103
104     <xsd:complexType name="AttributeRestrictionType" mixed="true">
105         <xsd:choice>
106             <xsd:element name="progress" type="ProgressType"/>
107
108             <xsd:element name="threshold" type="ThresholdType"/>
109
110             <xsd:element name="restrictionRule" type="xsd:anyType"
                    />
111         </xsd:choice>
112
113         <!-- service property this restriction applies to -->
114         <xsd:attribute name="propertyID" type="xsd:anyURI" />
115     </xsd:complexType>
116
117     <xsd:complexType name="OfferRestrictionsType" mixed="true">
118         <xsd:sequence>
119             <xsd:element name="attributeRestriction" type="
                    AttributeRestrictionType" minOccurs="0" maxOccurs="
                    unbounded"/>
120
121             <xsd:element name="generalRestriction" type="
                    xsd:anyType" minOccurs="0" maxOccurs="unbounded"/>
122         </xsd:sequence>
123     </xsd:complexType>
124
125     <xsd:simpleType name="MatchingFormType">
126         <xsd:restriction base="xsd:string">
127             <xsd:enumeration value="forwarded"/>
128             <xsd:enumeration value="defined"/>
129         </xsd:restriction>
130     </xsd:simpleType>
131
132     <xsd:complexType name="OfferAllocationPolicyType">
133         <xsd:sequence>
134             <!--this element is present if defined matching is
                    specified-->
135             <xsd:element name="matchingRule" type="xsd:anyType"
                    minOccurs="0" maxOccurs="1"/>
136         </xsd:sequence>
137
```

```
138         <xsd:attribute name="matchingForm" type="MatchingFormType"
                />
139     </xsd:complexType>
140
141     <xsd:simpleType name="TransparencyType">
142         <xsd:restriction base="xsd:string">
143             <xsd:enumeration value="public"/>
144             <xsd:enumeration value="protected"/>
145             <xsd:enumeration value="none"/>
146         </xsd:restriction>
147     </xsd:simpleType>
148
149     <xsd:complexType name="InformationPolicyType" mixed="true">
150         <xsd:sequence>
151             <xsd:element name="negotiationTransparency" type="
                    TransparencyType"/>
152
153             <!--this element is present if negotiationTransparency
                     is not set to none-->
154             <xsd:element name="negotiationContent" type="
                    xsd:anyType" minOccurs="0" maxOccurs="1"/>
155
156             <xsd:element name="statusTransparency" type="
                    TransparencyType"/>
157
158             <!--this element is present if statusTransparency is
                    not set to none-->
159             <xsd:element name="statusContent" type="xsd:anyType"
                    minOccurs="0" maxOccurs="1"/>
160         </xsd:sequence>
161     </xsd:complexType>
162
163     <xsd:simpleType name="RoleNameType">
164     <!-- role abstraction needed for reusability of the protocol
            descriptions -->
165         <xsd:restriction base="xsd:string">
166             <xsd:enumeration value="serviceProvider"/>
167             <xsd:enumeration value="serviceConsumer"/>
168             <xsd:enumeration value="negotiationCoordinator"/>
169         </xsd:restriction>
170     </xsd:simpleType>
171
172     <xsd:simpleType name="MessageType">
173         <xsd:restriction base="xsd:string">
174             <xsd:enumeration value="callForBids"/>
175             <xsd:enumeration value="offer"/>
176             <xsd:enumeration value="still_interested"/>
177             <xsd:enumeration value="notification_accept"/>
178             <xsd:enumeration value="notification_reject"/>
179             <xsd:enumeration value="admission"/>
180         </xsd:restriction>
181     </xsd:simpleType>
182
183     <xsd:complexType name="EventActionType" >
```

```
184         <xsd:sequence>
185             <xsd:element name="messageType" type="MessageType" />
186             </xsd:sequence>
187 
188         <!-- message receiving events -->
189         <xsd:attribute name="from" type="RoleNameType" />
190 
191         <!-- message sending actions -->
192         <xsd:attribute name="to" type="RoleNameType" />
193     </xsd:complexType>
194 
195     <xsd:complexType name="ProtocolStepsType">
196         <xsd:sequence>
197                 <xsd:element name="protocolStep" maxOccurs="
                        unbounded">
198 
199                     <xsd:complexType>
200                         <xsd:sequence>
201                             <xsd:element name="event" type="
                                    EventActionType" minOccurs="0" />
202 
203                             <xsd:element name="possibleAction"
                                    type ="EventActionType" maxOccurs="
                                    unbounded"/>
204                         </xsd:sequence>
205                     </xsd:complexType>
206                 </xsd:element>
207         </xsd:sequence>
208     </xsd:complexType>
209 
210     <xsd:complexType name="ProcessType">
211         <xsd:sequence>
212                 <xsd:element name="serviceConsumer" type="
                        ProtocolStepsType" minOccurs="0" maxOccurs="
                        unbounded" />
213             </xsd:sequence>
214     </xsd:complexType>
215 
216     <xsd:complexType name="NegotiationProtocolType" mixed="true">
217         <xsd:sequence>
218             <xsd:element name="context" type="ContextType" />
219 
220             <xsd:element name="negotiationObject" type="
                    NegotiationObjectType" />
221 
222             <xsd:element name="offerRestrictions" type="
                    OfferRestrictionsType"/>
223 
224             <xsd:element name="offerAllocationPolicy" type="
                    OfferAllocationPolicyType"/>
225 
226             <xsd:element name="informationPolicy" type="
                    InformationPolicyType"/>
227 
```

```
228             <xsd:element name="process" type="ProcessType" />
229         </xsd:sequence>
230     </xsd:complexType>
231
232     <!-- defines the available non-functional aspects of a servics
            and how to negotiate about them -->
233     <xsd:element name="ExtendedSLATemplate">
234         <xsd:complexType mixed="true">
235             <xsd:sequence>
236                 <!-- qualified name of this extended SLA template
                        -->
237                 <xsd:element name="slaTemplateID" type="xsd:anyURI
                        " />
238
239                 <!-- set of initial service level objectives -->
240                 <xsd:element name="slaTemplate"  type="
                        SLATemplateType" />
241
242                 <!-- negotiation protocol description -->
243                 <xsd:element name="negotiationProtocol" type="
                        NegotiationProtocolType" />
244             </xsd:sequence>
245         </xsd:complexType>
246     </xsd:element>
247 </xsd:schema>
```

A.1.3. Service Identificator Document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://xml.SLANeg.org/schema/SLANeg"
            xmlns="http://xml.SLANeg.org/schema/SLANeg"
            elementFormDefault="qualified">

    <!-- identifies and describes a given service by refering to
        the respective description documents -->
    <xsd:element name="ServiceIdentificator">
        <xsd:complexType mixed="true">
            <xsd:sequence>
                <!-- qualified name of the service instance -->
                <xsd:element name="serviceID" type="xsd:anyURI"/>

                <!-- link to the service's type -->
                <xsd:element name="serviceTypeID" type="xsd:anyURI
                    "/>

                <!-- link to the extended SLA template offered for
                     this service -->
                <xsd:element name="slaTemplateID" type="xsd:anyURI
                    "/>

                <!-- link to the WSDL file applied for this
                    service for subsequent invocation -->
                <xsd:element name="wsdlFile" type="xsd:anyURI"/>

                <!-- link to the negotiation coordinator -->
                <xsd:element name="negotiationCoordinator" type="
                    xsd:anyURI" />

                <!-- link to the service provider (link to the
                    service consumer would be necessary in reverse
                    negotiations -->
                <xsd:element name="serviceProvider" type="
                    xsd:anyURI" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

A.2. Activity Diagrams of the Service Management Agents

A.2.1. Service Consumer

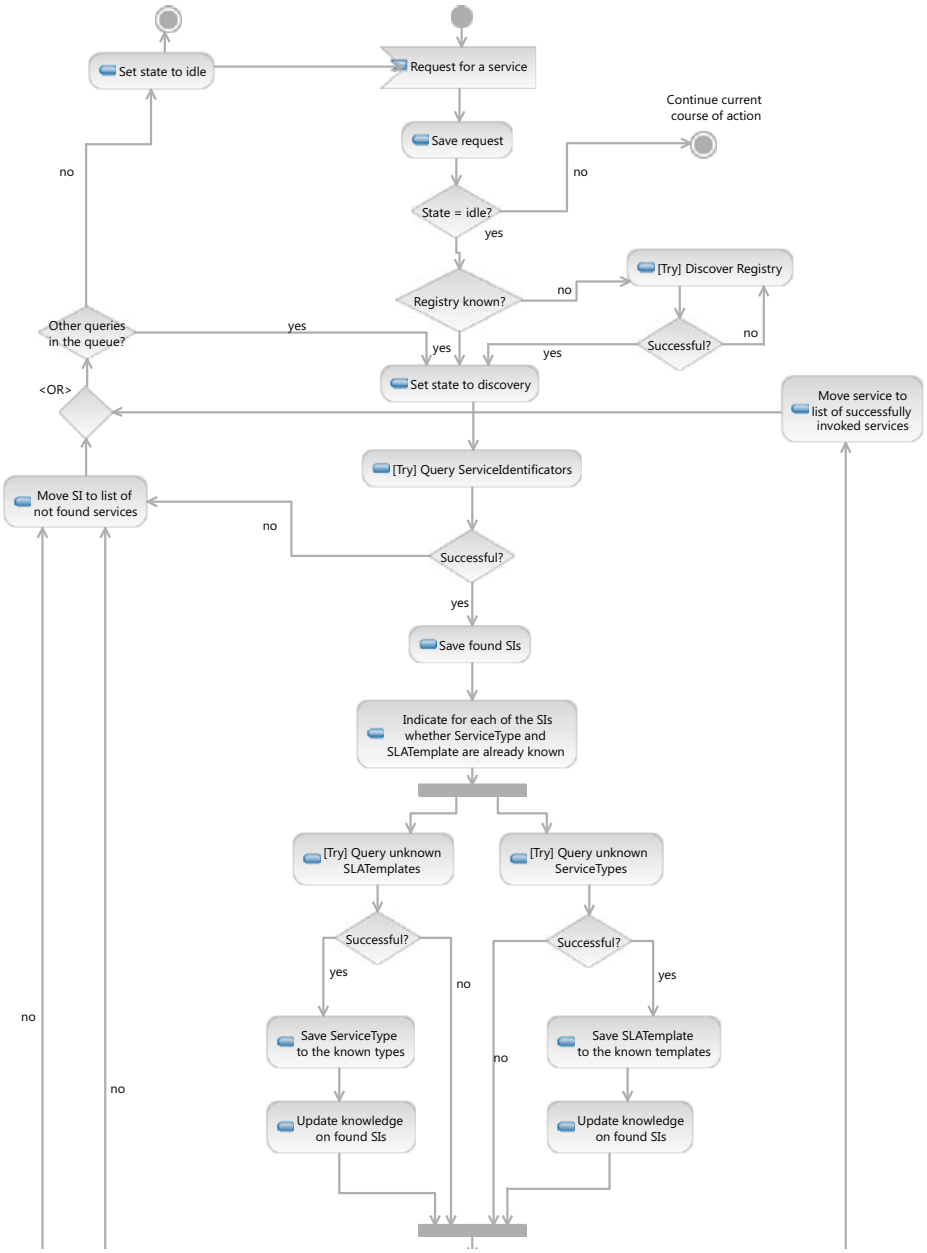


Figure A.1.: Activity Diagram: SC (part1)

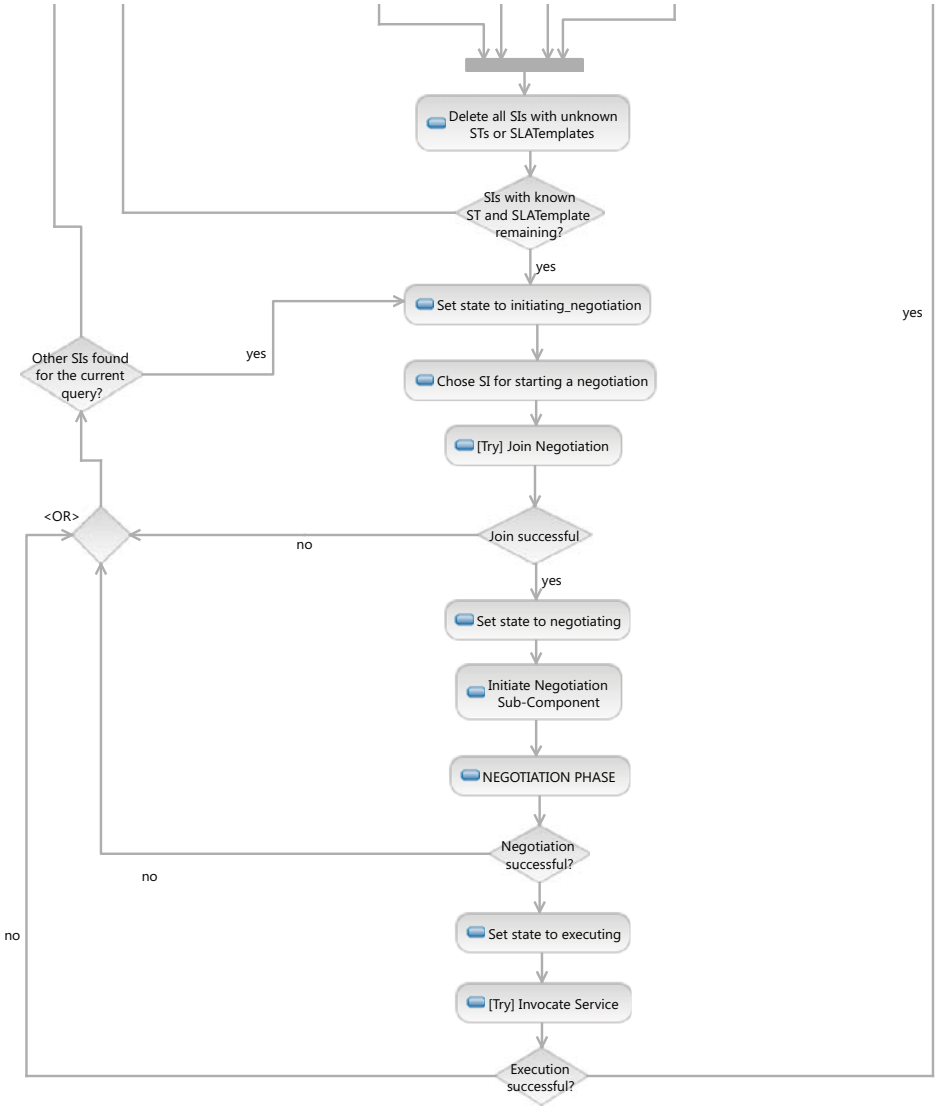


Figure A.2.: Activity Diagram: SC (part2)

A.2.2. Service Provider

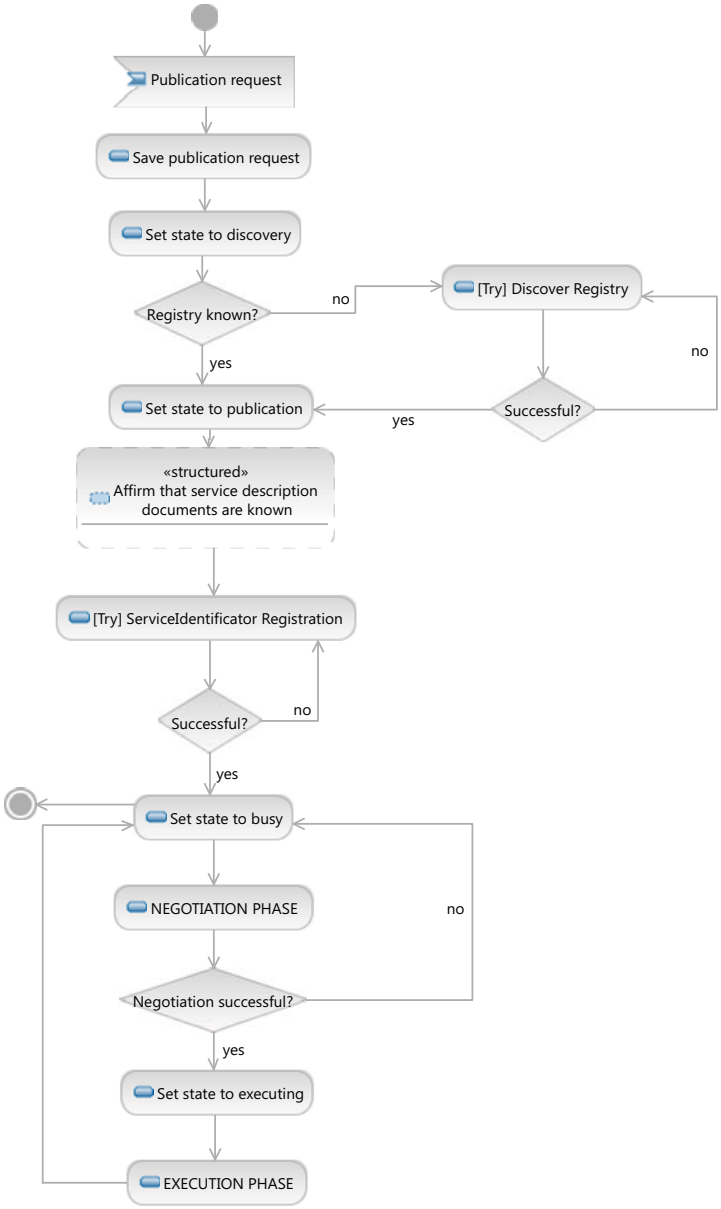


Figure A.3.: Activity Diagram: SP

A.2.3. Sub-diagram (SP): Affirming that Service Description Documents are Known at the Registry

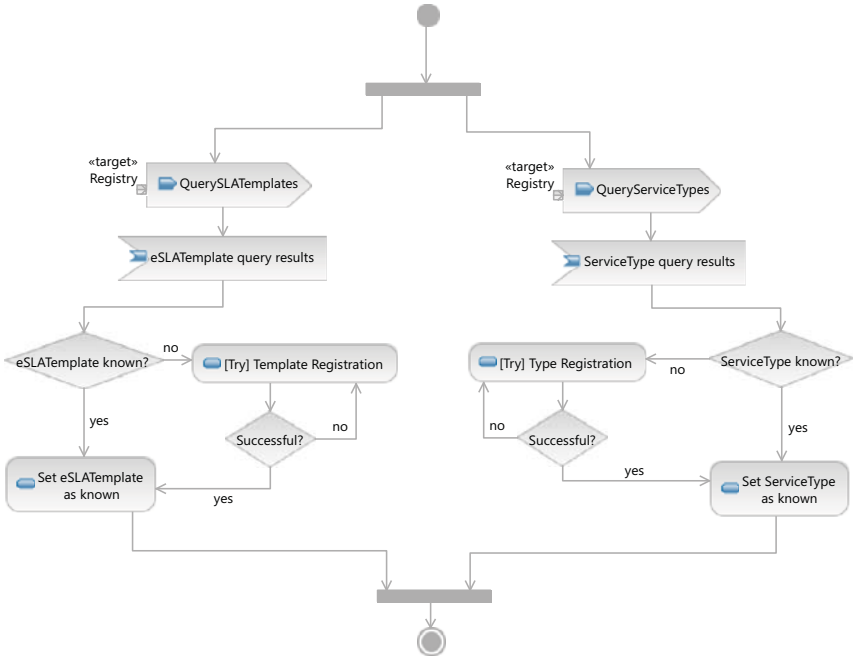


Figure A.4.: Sub-diagram for the SP: Affirmation that EST and ST Documents are known at the Registry

Bibliography

Aberer, Karl and Manfred Hauswirth (2002). *An Overview on Peer-to-Peer Information Systems*. last checked: 2011-05-03. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.681.

Andrieux, Alain et al. (2007). *Web Services Agreement (WS-Agreement) Specification, Version 03/2007*. last checked: 2011-05-03. Open Grid Forum (OGF). URL: http://www.ogf.org/documents/GFD.107.pdf.

Ardaiz, Oscar et al. (2006). "The Catallaxy Approach for Decentralized Economic-based Allocation in Grid Resource and Service Markets". In: *Journal of Applied Intelligence* 25.2, pp. 131–145. ISSN: 0924-669X. DOI: http://dx.doi.org/10.1007/s10489-006-9650-9.

Ardaiz, Oscar et al. (2007). *Third Year Report of WP3: Proof-of-Concept Prototype*. Project Deliverable. ISSN: 1864-9300, last checked: 2011-05-03. CATNETS Consortium. URL: http://opus.ub.uni-bayreuth.de/volltexte/2007/372/.

BREIN (2010). *Publishable final Activity Report*. Project Deliverable. last checked: 2011-05-03. BREIN Consortium. URL: http://www.eu-brein.com/index.php?option=com_docman&task=doc_download&gid=86.

Balakrishnan, Hari et al. (2003). *Looking up Data in P2P Systems*. last checked: 2011-05-04. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.9709.

Barros, Alistair, Marlon Dumas, and Peter Bruza (2005). *The Move to Web Service Ecosystems*. BPTrends Newsletter, 3(3). last checked: 2011-05-04. URL: http://www.bptrends.com/publicationfiles/12-05-WP-WebServiceEcosystems-Barros-Dumas.pdf.

Bartolini, C., C. Preist, and N. R. Jennings (2005). "A Software Framework for Automated Negotiation". In: *Proceedings of the Conference on Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications (SELMAS 2004)*. Ed. by R. Choren et al. Lecture Notes on Computer Science. Springer Publishing, pp. 213–235.

Beatty, John et al. (2005). *Web Services Dynamic Discovery (WS-Discovery)*. last checked: 2011-05-04. BEA Systems, Canon, Intel, Microsoft Cooperation and webMethods. URL: http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf.

Begin, Marc-Elian (2008). *An EGEE Comparative Study: Grids and Clouds, Evolution or Revolution?* presented at OGF 23, June 1. - 6. 2008, Barcelona, Spain. last checked: 2011-05-04. URL: http://www.sixsq.com/internal/events/cloud-vs-grid-comparative-study.

Bellwood, Tom et al. (2004). *Universal Description, Discovery and Integration v3.0.2*. Standard. last checked: 2011-05-04. OASIS UDDI Specification Technical Commit-

tee. URL: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm.

Benyoucef, Morad and Stefanie Rinderle (2006). "Modeling e-Negotiation Processes for a Service Oriented Architecture". In: *Group Decision and Negotiation* 15.5, pp. 449–467. DOI: 10.1007/s10726-006-9038-6.

Benyoucef, Morad and Marie-Helene Verrons (2008). "Configurable e-Negotiation Systems for large scale and transparent Decision Making". In: *Group Decision and Negotiation* 17.3, pp. 211–224. DOI: 10.1007/s10726-007-9073-y.

Berger, Thomas (2005). "Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen". last checked: 2011-05-04. PhD thesis. Technical University of Darmstadt. URL: http://tuprints.ulb.tu-darmstadt.de/570/.

Bichler, Martin (2001). *The Future of eMarkets - Multi-Dimensional Market Mechanisms*. Cambridge: Cambridge University Press.

Bichler, Martin and Jayant R. Kalagnanam (2006). "Software Frameworks for Advanced Procurement Auction Markets". In: *Communications of the ACM* 49.12. DOI: http://doi.acm.org/10.1145/1183236.1183239.

Bichler, Martin, Gregory Kersten, and Stefan Strecker (2003). "Towards a Structured Design of Electronic Negotiations". In: *Group Decision and Negotiation* 12.4, pp. 311–335.

Blau, Benjamin, Jochen Stösser, and Carsten Block (2008). "How to trade Electronic Services? – Current Status and Open Questions". In: *Proceedings of the Conference on Group Decision and Negotiation 2008, Coimbra, Portugal, June 17th - 20th*.

Blau, Benjamin et al. (2009). "Service Value Networks". In: *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC '09), Washington, DC, USA*. IEEE Computer Society, pp. 194–201. ISBN: 978-0-7695-3755-9. DOI: http://dx.doi.org/10.1109/CEC.2009.64.

Block, Carsten and Dirk Neumann (2008). "A Decision Support System for Choosing Market Mechanisms in e-Procurement". In: *Negotiation, Auctions and Market Engineering, Lecture Notes in Business Information Processing* 2, pp. 44–57. DOI: http://dx.doi.org/10.1007/978-3-540-77554-6_3.

Boag, Scott et al. (2007). *XQuery 1.0: An XML Query Language*. Recommendation. last checked: 2011-05-04. World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/xquery/.

Boehm, B. (1986). "A Spiral Model of Software Development and Enhancement". In: *SIGSOFT Software Engineering Notes* 11.4, pp. 14–24. ISSN: 0163-5948. DOI: http://doi.acm.org/10.1145/12944.12948.

Boehmann, T. and H. Krcmar (2004). "Grundlagen und Entwicklungstrends im IT-Servicemanagement". In: *HMD - Praxis der Wirtschaftsinformatik* 237. last checked: 2011-05-05, pp. 7–21. URL: http://hmd.dpunkt.de/237/01.php.

Booth, D. et al. (2004). *Web Services Architecture*. Standard. last checked: 2011-05-04. World Wide Web Consortium. URL: http://www.w3.org/TR/ws-arch/.

Borissov, Nikolay, Dirk Neumann, and Christof Weinhardt (2009). "Automated bidding in Computational Markets: An Application in Market-based Allocation of Comput-

ing Services". In: *Journal of Autonomous Agents and Multi-Agent Systems* 21.2, pp. 115–142. DOI: 10.1007/s10458-009-9112-y.

Borissov, Nikolay et al. (2009). "Message Protocols for Provisioning and Usage of Computing Services". In: *Proceedings of the 6th International Workshop on Grid Economics and Business Models 2009 (GECON 09)*. Ed. by Jörn Altmann, Rajkumar Buyya, and Omer F. Rana. Vol. 5745/2009. Lecture Notes in Computer Science, pp. 160–170.

Brandic, Ivona et al. (2008a). "Towards a Meta-Negotiation Architecture for SLA-Aware Grid Services". In: *Proceedings of the Workshop on Service-Oriented Engineering and Optimizations 2008. In conjunction with International Conference on High Performance Computing 2008 (HiPC2008), Bangalore, India*.

Brandic, Ivona et al. (2008b). *Towards a Meta-Negotiation Architecture for SLA-aware Grid Services*. Technical Report. last checked: 2011-05-03. University of Melbourne. URL: http://www.gridbus.org/reports/meta-negotiation2008.pdf.

Bray, Tim et al. (2006). *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. last checked: 2011-05-04. World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/2006/REC-xml-20060816.

Bucher, Tobias, Christian Riege, and Jan Saat (2008). "Evaluation in der gestaltungsorientierten Wirtschaftsinformatik - Systematisierung nach Erkenntnisziel und Gestaltungsziel". In: *Proceedings of the Multikonferenz Wirtschaftsinformatik 2008 (MKWI 2008), Munich, 26.-28. Februar 2008*. Vol. 120. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Wilhelms-University Münster. ISSN 1438-3985, pp. 69–86.

Bui, Tung, Alexandre Gachet, and Hans-Juergen Sebastian (2006). "Web Services for Negotiation and Bargaining in Electronic Markets: Design Requirements, Proof-of-Concepts, and Potential Applications to e-Procurement". In: *Group Decision and Negotiation* 15.5, pp. 469–490. DOI: 10.1007/s10726-006-9039-5.

Buyya, Rajkumar, David Abramson, and Srikumar Venugopal (2005). "The Grid Economy". In: *Proceedings of the IEEE* 93, pp. 698–714.

Buyya, Rajkumar et al. (2009). "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility". In: *Future Generation Computer Systems* 25.6, pp. 599 –616. ISSN: 0167-739X. DOI: http://dx.doi.org/10.1016/j.future.2008.12.001.

Chakravorty, R. et al. (2003). *Dynamic SLA-based QoS Control for Third Generation Wireless Networks: The CADENUS Extension*. last checked: 2011-05-04. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.2357.

Chhetri, Mohan Baruwal et al. (2007). "ASAPM - An Agent-Based Framework for Adaptive Management of Composite Service Lifecycle". In: *Proceedings of the IEEE / WIC / ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, 2007*, pp. 444–448.

Christensen, Clayton M. (1997). *The Innovator's Dilemma – When Technologies Cause Great Firms to Fail*. Boston, Massachusetts: Harvard Business School Press, p. 225. ISBN: 978-0875845852.

Cramton, Peter, Yoav Shoham, and Richard Steinberg (2006). "Introduction to Combinatorial Auctions". In: *Combinatorial Auctions*, pp. 1–14.

Dan, Asit, Heiko Ludwig, and Giovanni Pacifici (2003). *Web Service Differentiation with Service Level Agreements*. last checked: 2011-05-04. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.4338.

Detken, Kai-Oliver (2004). "Service Discovery - Automatisches Auffinden mobiler Dienste". In: *NET* 6, pp. 34–36.

Dimitrakos, Theo (2009). *Integration of Technical Results of the BEinGRID project*. Project White Paper. last checked: 2011-05-03. BEinGRID Consortium. URL: http://www.it-tude.com/bigpicture.html.

Dubin, Robert (1978). *Theory Building, Revised Edition*. English. Free Press, London.

Ermolayev, Vadim and Natalya Keberle (2006). "A Generic Ontology of Rational Negotiation". In: *Proceedings of the 5th International Conference on Information Systems Technology and their Applications (ISTA)*, pp. 51–65.

Eymann, Torsten, Werner Streitberger, and Sebastian Hudert (2007). "CATNETS - Open Market Approaches for Self-Organizing Grid Resource Allocation". In: *Proceedings of the 4th International Workshop on Grid Economics and Business Models (GECON 2007), Rennes, France*. Ed. by J. Altmann D.J. Veit. Lecture Notes in Computer Science (LNCS) 4685. Springer Publishers, Berlin, pp. 176–181. DOI: http://dx.doi.org/10.1007/978-3-540-74430-6.

Ferstl, Otto K. and Elmar J. Sinz (2008). *Grundlagen der Wirtschaftsinformatik*. 6. Edition. ISBN: 978-3-486-58755-5. Oldenbourg Verlag München.

Foster, I. et al. (2002a). *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Open Grid Service Infrastructure WG, Global Grid Forum. last checked: 2011-05-04. URL: http://www.globus.org/alliance/publications/papers/ogsa.pdf.

Foster, Ian (2002). *What is the Grid? A Three Point Checklist*. last checked: 2011-05-04. URL: http://dlib.cs.odu.edu/WhatIsTheGrid.pdf.

Foster, Ian (2005). "Service-Oriented Science". In: *Science* 308.5723, pp. 814–817. DOI: 10.1126/science.1110411. eprint: http://www.sciencemag.org/cgi/reprint/308/5723/814.pdf.

Foster, Ian and Carl Kesselman, eds. (2004). *The Grid: Blueprint for a New Computing Infrastructure, 2nd Edition*. Morgan Kaufmann Publishers.

Foster, Ian, Carl Kesselman, and Steven Tuecke (2001). "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". In: *International Journal of Supercomputer Applications* 15, pp. 200–222. DOI: http://dx.doi.org/10.1177%2f109434200101500302.

Foster, Ian et al. (2002b). "Grid Services for Distributed System Integration". In: *Computer* 35.6, pp. 37–46. ISSN: 0018-9162. DOI: http://dx.doi.org/10.1109/MC.2002.1009167.

Frutos, Henar Munoz and Ioannis Kotsiopoulos (2009). "BREIN: Business Objective Driven Reliable and Intelligent Grids for real Business". In: *International Journal of Interoperability in Business Information Systems* 3.1. ISSN: 1862-6378, pp. 39–41.

Gamma, Erich et al. (1995). *Design Patterns - Elements of Resuable Object-Oriented Software*. Addison-Wesley Publishing.

Glass, Robert L., V. Ramesh, and Iris Vessey (2004). "An Analysis of Research in Computing Disciplines". In: *Communications of the ACM, Issue on Wireless Sensor Networks* 47.6. ISSN:0001-0782, pp. 89–94. DOI: http://dx.doi.org/10.1145/990680.990686.

Goldberg, David (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goo, Jahyun, Rajiv Kishore, and H. R. Rao (2009). "The Role of Service Level Agreements in Relational Management of Information Technology Outsourcing: An Empirical Study". In: *MIS Quarterly* 33.1. last checked: 2011-05-05, pp. 119–145. URL: http://www.acsu.buffalo.edu/~rkishore/papers/Goo-MISQ-33-1-2009.pdf.

Gradwell, Peter and Julian Padget (2005). "Markets vs Auctions: Approaches to Distributed Combinatorial Resource Scheduling". In: *Multiagent and Grid Systems* 1. last checked: 2011-05-05, pp. 251–262. URL: http://portal.acm.org/citation.cfm?id=1233712.

Gregor, Shirley and David Jones (2007). "The Anatomy of a Design Theory". In: *Journal of the Association for Information Systems* 8.5. ISSN: 1536-9323, pp. 312–335.

Gudgin, Martin et al. (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/2007/REC-soap12-part1-20070427.

Hartmann, Stephan (1996). "The World as a Process: Simulations in the Natural and Social Sciences". In: *Modelling and Simulaton in the Social Sciences from the Philosophy of Science Point of View*. Ed. by Rainer Hegselmann, Ulrich Mueller, and Klaus G. Troitzsch. last checked: 2011-05-04. Dordrecht: Kluwer Academic Publishers, pp. 77–100. URL: http://philsci-archive.pitt.edu/archive/00002412/01/Simulations.pdf.

Hasselmeyer, Peer (2005). "On Service Discovery Process Types". In: *Proceedings of the 3rd International Conference on Service Oriented Computing (ICSOC05)*.

Hasselmeyer, Peer et al. (2006a). "Towards Autonomous Brokered SLA Negotiation". In: *Exploiting the Knowledge Economy: Issues, Applications, Case Studies (eChallenges 2006), Barcelona, Spain* 3. ISBN 978-1-58603-682-9, pp. 44–51.

Hasselmeyer, Peer et al. (2006b). "Towards SLA-Supported Resource Management". In: *High Performance Computing and Communications*. Vol. 208/2006. Lecture Notes in Computer Science (LNCS). last checked: 2011-05-04, pp. 743–752. URL: http://www.springerlink.com/content/n756m230705t66g3/.

Hasselmeyer, Peer et al. (2007). "Implementing an SLA Negotiation Framework". In: *Proceedings of the eChallenges Conference (e-2007)*.

Haykin, Simon (1994). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0023527617.

Hevner, Alan R. et al. (2004). "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1, pp. 75–105.

Hudert, Sebastian (2006). *A Proposal for a Web Services Agreement Negotiation Protocol Framework*. Bamberger Beitraege zur Wirtschaftsinformatik und Angewandten Informatik 69. last checked: 2011-05-03. University of Bamberg. URL: https://www.opus-bayern.de/uni-bamberg/frontdoor.php?source_opus=323.

Hudert, Sebastian (2009). "A Protocol-generic Infrastructure for Electronic SLA Negotiations in the Internet of Services". In: *Proceedings of the Doctoral Consortium, co-located with the 9th International Conference Wirtschaftsinformatik 2009, February 23 - 24 2009, Vienna, Austria*. Ed. by Torsten Eymann. Vol. 40. Bayreuther Arbeitspapiere zur Wirtschaftsinformatik.

Hudert, Sebastian (2010). "From Service Markets to Service Economies - An Infrastructure for Protocol-generic SLA Negotiations". In: *Grids and Service-Oriented Architectures for Service Level Agreements (Proceedings of the Dagstuhl Seminar on Grids and Service-Oriented Architectures for Service Level Agreements)*. Ed. by Philipp Wieder, Ramin Yahyapour, and Wolfgang Ziegler. Vol. 1. ISBN: 978-1-4419-7319-1. Springer, pp. 145–156.

Hudert, Sebastian and Torsten Eymann (2010). "Coping with Global Information Systems - Requirements for a Flexible SLA Discovery and Negotiation Infrastructure for the Future Internet of Services". In: *Proceedings of the Multikonferenz Wirtschaftsinformatik 2010. Göttingen, 23th - 25th February 2010*. Ed. by Matthias Schumann et al. Univ.-Verl. Göttingen; Niedersächsische Staats-und Universitätsbibliothek.

Hudert, Sebastian and Torsten Eymann (2011a). "A protocol-generic Infrastructure for electronic SLA Negotiations in the Internet of Services". In: *Proceedings of the Fall 2010 Future SOC Lab Day*. Ed. by Christoph Meinel et al. Vol. 42. ISBN 978-3-86956-114-1.

Hudert, Sebastian and Torsten Eymann (2011b). "The BabelNEG System - A prototype Infrastructure for protocol-generic SLA Negotiations". In: *Proceedings of the 10th International Conference on Wirtschaftsinformatik (WI 2011), 16th - 18th of February, Zurich, Switzerland*. Ed. by Abraham Bernstein and Gerhard Schwabe, pp. 704–713.

Hudert, Sebastian, Heiko Ludwig, and Guido Wirtz (2006). *A Negotiation Protocol Framework for WS-Agreement*. Research Report RC24094. last checked: 2011-05-03. IBM. URL: `http://domino.research.ibm.com/library/cyberdig.nsf/papers/2F154C6C066AF94F8525721B0056C019/\$File/rc24094.pdf`.

Hudert, Sebastian, Heiko Ludwig, and Guido Wirtz (2007). "A Negotiation Protocol Framework for WS-Agreement". In: *Proceedings of the 15.TTG/GI Fachtagung Kommunikation in Verteilten Systemen (KIVS 07), February 26th to March 2nd, 2007, Bern, Switzerland*.

Hudert, Sebastian, Heiko Ludwig, and Guido Wirtz (2008). "Negotiating Service Levels – A Generic Negotiation Framework for WS-Agreement". In: *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE08), 2008, Redwood City, San Francisco Bay, USA*, pp. 587–592.

Hudert, Sebastian, Heiko Ludwig, and Guido Wirtz (2009). "Negotiating SLAs - An Approach for a Generic Negotiation Framework for WS-Agreement". In: *Journal of Grid Computing* 7.2. ISSN: 1570-7873 (Print) 1572-9814 (Online), pp. 225–246. DOI: `10.1007/s10723-009-9118-3`.

Hudert, Sebastian, Christoph Niemann, and Torsten Eymann (2010). "On Computer Simulation as a Component in Information Systems Research". In: *Global Perspectives on Design Science Research (Proceedings of the DESRIST 2010, St. Gallen,*

Switzerland). Vol. 6105/2010. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 167–179. DOI: 10.1007/978-3-642-13335-0.

Hudert, Sebastian et al. (2009). "A Negotiation Protocol Description Language for Automated Service Level Agreement Negotiations". In: *Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC 09), Vienna, Austria*. URL: http://portal.acm.org/citation.cfm?id=1603226.

Hung, Patrick C. K., Haifei Li, and Jun-Jang Jeng (2004). "WS-Negotiation: An Overview of Research Issues". In: *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*. IEEE Computer Society, p. 10033.2. ISBN: 0-7695-2056-1.

IT Governance Institute (2007). *COBIT 4.1*. ISBN: 1933284722. ISACA.

Jennings, N. R. et al. (2001). "Automated Negotiation: Prospects, Methods and Challenges". In: *Group Decision and Negotiation* 10.2, pp. 199–215.

Jennings, Nicholas R. (2000). "On Agent-based Software Engineering". In: *Artificial Intelligence* 177.2, pp. 277–296. DOI: http://dx.doi.org/10.1016/S0004-3702(99)00107-1.

Joita, L. et al. (2007). "Service Level Agreements in Catallaxy-Based Grid Markets". In: *Proceedings of the Usage of Service Level Agreements in Grids Workshop, in conjunction with The 8th IEEE International Conference on Grid Computing (Grid 2007), Austin, Texas, USA*.

Jonker, Catholijn M., Valentin Robu, and Jan Treur (2007). "An Agent Architecture for Multi-attribute Negotiation using Incomplete Preference Information". In: *Autonomous Agents and Multi-Agent Systems* 15.2, pp. 221–252. ISSN: 1387-2532. DOI: http://dx.doi.org/10.1007/s10458-006-9009-y.

Karaenke, Paul and Stefan Kirn (2007). "Service Level Agreements: An Evaluation from a Business Application Perspective". In: *Proceedings of the eChallenges 2007 Conference, Den Haag, Niederlande*, pp. 104–111.

Karten, Naomi (2003). *How to establish Service Level Agreements*. Naomi Karten Associates.

Kelaskar, M. et al. (2002). "A Study of Discovery Mechanisms for Peer-to-Peer Applications". In: *Proceedings of the 2 nd IEEE/ACM International Symposium on Cluster Computing and the Grid Workshop on Global and Peer-to-Peer on Large Scale Distributed Systems*, pp. 444–445.

Keller, Alexander and Heiko Ludwig (2003). "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services". In: *Journal of Network and Systems Management* 11.1, pp. 57–81. ISSN: 1064-7570. DOI: http://dx.doi.org/10.1023/A:1022445108617.

Keller, Alexander et al. (2002a). "Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture". In: *In Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE Press, pp. 513–528.

Keller, Alexander et al. (2002b). "Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture". In: *Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE Press, pp. 513–528.

Kersten, Gregory E., Ka Pong Law, and Stefan Strecker (2004). *A Software Platform for Multiprotocol E-Negotiations*. Research Paper INR04/04. last checked: 2011-05-03. InterNeg. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.6777&rep=rep1&type=pdf.

Kersten, Gregory E. and Sunil J. Noronha (1999). "WWW-based Negotiation Support: Design, Implementation, and Use". In: *Decision Support Systems* 25.2, pp. 135–154. ISSN: 0167-9236. DOI: http://dx.doi.org/10.1016/S0167-9236(99)00012-3.

Kertesz, Attila, Gabor Kecskemeti, and Ivona Brandic (2009). "An SLA-based Resource Virtualization Approach for On-demand Service Provision". In: *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing (VTDC '09), New York, NY, USA*. Barcelona, Spain: ACM, pp. 27–34. ISBN: 978-1-60558-580-2. DOI: http://doi.acm.org/10.1145/1555336.1555341.

Kim, Jin Baek and Arie Segev (2005). "A Web Services-enabled Marketplace Architecture for Negotiation Process Management". In: *Decision Support Systems* 40.1, pp. 71–87. ISSN: 0167-9236. DOI: http://dx.doi.org/10.1016/j.dss.2004.04.005.

Kirn, S. (2006). "Multiagent Engineering - Theory and Applications in Enterprises". In: ed. by S. Kirn et al. ISBN: 3-540-31406-7. Springer. Chap. Flexibility of Multiagent Systems, pp. 53–69.

Klemperer, Paul (1999). "Auction Theory: A Guide to The Literature". In: *Journal of Economic Surveys* 13.3, pp. 227–287.

König, Stefan, Sebastian Hudert, and Torsten Eymann (2010). "Socio-Economic Mechanisms to Coordinate the Internet of Services – the Simulation Environment SimIS". In: *Journal of Artificial Societies and Social Simulation (JASSS)* 13.2. URL: http://jasss.soc.surrey.ac.uk/13/2/6.html.

Kotsiopoulos, Ioannis et al. (2008). "Using Semantic Technologies to Improve Negotiation of Service Level Agreements". In: *Collaboration and the Knowledge Economy: Issues, Applications, Case Studies (Proceedings of the eChallenges'08, Stockholm, Sweden)*. Ed. by Paul Cunningham and Miriam Cunningham. ISBN 978-1-58603-924-0. IOS Press.

Kotsokalis, Costas et al. (2010). *SLA@SOI Deliverable D.A5a - Foundations for SLA Management*. Project Deliverable. last checked: 2011-05-03. SLA@SOI Consortium. URL: http://sla-at-soi.eu/results/deliverables/.

Lai, Guoming et al. (2004). *Literature Review on Multi-attribute Negotiations*. Research Report CMU-RI-TR-04-66. last checked: 2011-05-03. Carnegie Mellon University, Robotics Institute. URL: http://www.ri.cmu.edu/publication_view.html?pub_id=4868.

Lamanna, Davide D., James Skene, and Wolfgang Emmerich (2003). "SLAng: A Language for Defining Service Level Agreements". In: *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03), Washington, DC, USA*. IEEE Computer Society. ISBN: 0769519105.

Lambea, Juan et al. (2009). *SLA@SOI Deliverable D.A2a - Business SLA Management*. Projcet Deliverable. last checked: 2011-05-03. SLA@SOI Consortium. URL: http://sla-at-soi.eu/results/deliverables/.

Lambea, Juan et al. (2010). *SLA@SOI Deliverable D.A2a - Business SLA Management (full lifecycle)*. Project Deliverable. last checked: 2011-05-03. SLA@SOI Consortium. URL: http://sla-at-soi.eu/results/deliverables/.

Laria, G. et al. (2009). *BREIN Deliverable D 4.1.3 v2 - Final BREIN Architecture*. Project Deliverable. last checked: 2011-05-03. BREIN Consortium. URL: http://www.eu-brein.com/index.php?option=com_docman&task=doc_download&gid=63.

Leff, Avraham et al. (2003). "Service-level Agreements and Commercial Grids". In: *IEEE Internet Computing* 7.4, pp. 44–50. DOI: 10.1109/MIC.2003.1215659.

Li, Cuihong, Joseph Giampapa, and Katia Sycara (2003). *A Review of Research on Bilateral Negotiations*. Technical Report CMU-RI-TR-03-41. last checked: 2011-05-03. Carnegie Mellon University, Robotics Institute. URL: http://www.cs.cmu.edu/\~softagents/papers/negotiation_lit.pdf.

Lochner, Kevin M. and Michael P. Wellman (2004). "Rule-Based Specification of Auction Mechanisms". In: *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 818–825.

Lomuscio, Alessio R., Michael Wooldridge, and Nicholas R. Jennings (2003). "A Classification Scheme for Negotiation in Electronic Commerce". In: *Group Decision and Negotiation* 12.1, pp. 31–56.

Ludwig, Andre et al. (2006). "A Framework for Automated Negotiation of Service Level Agreements in Services Grids". In: *Lecture Notes in Computer Science, Proceedings of the Workshop on Web Service Choreography and Orchestration for Business Process Management, 2006*. Vol. 3812/2006.

Ludwig, Heiko et al. (2003a). "A Service Level Agreement Language for Dynamic Electronic Services". In: *Journal of Electronic Commerce Research* 3, pp. 43–59. DOI: http://dx.doi.org/10.1023/A:1021525310424.

Ludwig, Heiko et al. (2003b). *Web Service Level Agreement (WSLA): Language Specification Version 1.0*. Research Report 1.0. last checked: 2011-05-03. IBM Research. URL: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

Ludwig, Heiko et al. (2005). "Template-based Automated Service Provisioning - Supporting the Agreement-Driven Service Life-Cycle". In: *Lecture Notes in Computer Science - Proceedings of the International Conference on Service Oriented Computing (ICSOC2005)*. Vol. 3826/2005, pp. 283–295.

Mantar, Haci et al. (2006). "A Bandwidth-Broker Based Inter-domain SLA Negotiation". In: *Autonomic Management of Mobile Multimedia Services*. Ed. by Ahmed Helmy et al. Vol. 4267. Lecture Notes in Computer Science. last checked: 2011-05-04. Springer Berlin / Heidelberg, pp. 134–140. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.6580&rep=rep1&type=pdf.

March, Salvatore T. and Gerald F. Smith (1995). "Design and Natural Science Research on Information Technology". In: *Decision Support Systems* 15.4, pp. 251–266. DOI: 10.1016/0167-9236(94)00041-2.

Martin, John C. (2002). *Introduction to Languages and the Theory of Computation*. Third Edition. ISBN: 978-0072322002. McGraw-Hill Science/Engineering/Math.

Masche, Philipp, Paul Mckee, and Bryce Mitchell (2006). "The Increasing Role of Service Level Agreements in B2B Systems". In: *Proceedings of the International Conference on Web Information Systems, Setubal*, pp. 473–478.

Matros, Raimund et al. (2008). *SORMA Deliverable 1.1a: Requirements Description*. Project Deliverable. last checked: 2011-05-03. SORMA Consortium. URL: `http://www.im.uni-karlsruhe.de/sorma/deliverables.htm`.

Mayerl, Christian et al. (2005). "Methode für das Design von SLA-fähigen IT-Services". In: *Proceedings of Kommunikation in Verteilten Systemen (KiVS2005), Kaiserslautern, Germany*.

Meffert, Heribert (1985). "Größere Flexibilität als Unternehmungskonzept". In: *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung (ZfbF)* 37.2. (german), pp. 121–137.

Meyer, Bertrand (1992). "Applying "Design by Contract"". In: *Computer* 25.10, pp. 40–51. ISSN: 0018-9162. DOI: `http://dx.doi.org/10.1109/2.161279`.

Meyer, Harald et al. (2007). *ASG Deliverable 6.V-1: Reference Architecture: Requirements, Current Efforts and Design*. Project Deliverable. last checked: 2011-05-03. ASG Consortium. URL: `http://asg-platform.org/cgi-bin/twiki/view/Public/PublicDeliverables`.

Mitchell, Bryce and Paul Mckee (2005). "SLAs A Key Commercial Tool". In: *Proceedings of eChallenges 2005, Ljubljana, Slovenia*.

Mobach, D. G. A. et al. (2005). "A Two-Tiered Model of Negotiation based on Web Service Agreements". In: *Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS'05)*, pp. 202–213.

Molderez, Jean-Francois et al. (2006). *AssessGrid Deliverable 1.1: Requirements Analysis*. Project Deliverable. last checked: 2011-05-03. AssessGrid Consortium. URL: `http://www.assessgrid.eu/index.php?id=318`.

Mora, Juan et al. (2009). *BREIN Deliverable D5.3.4 - Usage Patterns and Best Practice*. Project Deliverable. last checked: 2011-05-03. BREIN Consortium. URL: `http://www.eu-brein.com/index.php?option=com_docman&task=cat_view&gid=39&Itemid=31`.

Neumann, Dir et al. (2003). "Applying the Montreal Taxonomy to State of the Art E-Negotiation Systems". In: *Group Decision and Negotiation* 12.4. ISSN: 0926-2644 (Print) 1572-9907 (Online), pp. 287–310. DOI: `10.1023/A:1024871921144`.

Neumann, Dirk, Jochen Stösser, and Christof Weinhardt (2007). "Bridging the Grid Adoption Gap - Developing a Roadmap for Trading Grids". In: *Proceedings of the 20th Bled eConference, Bled, Slovenia, 4th - 6th of June*. ISBN: 978-961-232-204-5.

Neumann, Dirk et al. (2008). "A Framework for Commercial Grids - Economic and Technical Challenges". In: *Journal of Grid Computing* 6.3. ISSN: 1570-7873, pp. 325–347. DOI: `10.1007/s10723-008-9105-0`.

North, Michael J., Nicholson T. Collier, and Jerry R. Vos (2006). "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit". In: *ACM Transactions on Modeling and Computer Simulation* 16.1, pp. 1–25. ISSN: 1049-3301. DOI: `http://doi.acm.org/10.1145/1122012.1122013`.

Object Constraint Language (2010). last checked: 2011-05-04. Object Management Group (OMG). URL: http://www.omg.org/spec/OCL/2.2/PDF/.

Office of Government Commerce (2007). *Service Strategy Book*. ISBN: 9780113310456. TSO (The Stationery Office).

Olmedo, Vicente et al. (2007). *Akogrimo Deliverable D4.2.4 - Consolidated Integrated Services Design and Implementation Report*. Project Deliverable. last checked: 2011-05-03. Akogrimo Consortium. URL: http://www.akogrimo.org/modules235a.html?name=UpDownload&req=viewdownload&cid=5.

Ouelhadj, D. et al. (2005). "A Multi-agent Infrastructure and a Service Level agreement Negotiation Protocol for Robust Scheduling in Grid Computing". In: *Advances in Grid Computing* 3470/2005, pp. 651–660.

Overhage, Sven and Peter Thomas (2005). "WS-Specification: Ein Spezifikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards". In: *Proceedings of the Wirtschaftsinformatik 2005, Bamberg*. Ed. by Sven Eckert und Tilman Isselhorst Otto K. Ferstl Elmar J. Sinz. ISBN: 978-3-7908-1574-0, pp. 1539–1558.

Padgett, James, Karim Djemame, and Peter Dew (2005). "Grid Service Level Agreements Combining Resource Reservation and Predictive Run-time Adaption". In: *4th UK e-Science All Hands Meeting (AHM'05)*.

Padgett, James et al. (2006). *AssessGrid Deliverable 1.3: System Architecture Specification and Developed Scenarios*. Project Deliverable. last checked: 2011-05-03. AssessGrid Consortium. URL: http://www.assessgrid.eu/index.php?id=318.

Papazoglou, Michael P. and D. Georgakopoulos (2003). "Service-Oriented Computing". In: *Communications of the ACM* 46.10, pp. 25–28.

Paprzycki, Marcin et al. (2004). "Implementing Agents Capable of Dynamic Negotiation". In: *Proceedings of the Conference on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC04), Timioara, Romania*. Mirton Press, pp. 369–380.

Parkin, Michael, Rosa M. Badia, and Josep Martrat (2008). *A Comparison of SLA Use in Six of the European Commissions FP6 Projects*. Technical Report TR-0129. last checked: 2011-05-03. CoreGRID. URL: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0129.pdf.

Paurobally, Shamimabi, Valentina Tamma, and Michael Wooldridge (2007). "A Framework for Web Service Negotiation". In: *ACM Transactions on Autonomous and Adaptive Systems* 2.4, p. 14. ISSN: 1556-4665. DOI: http://doi.acm.org/10.1145/1293731.1293734.

Peffers, Ken et al. (2008). "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3, pp. 45–77. DOI: http://dx.doi.org/10.2753/MIS0742-1222240302.

Popper, Karl Raimund (2002). *The Logic of Scientific Discovery*. Taylor & Francis. ISBN: 978-0415278447.

Pruitt, Dean G. (1981). *Negotiation Behavior*. Academic Press, New York.

Rana, Omer et al. (2008). "Monitoring and Reputation Mechanisms for Service Level Agreements". In: *Proceedings of the 5th international workshop on Grid Economics and Business Models (GECON '08)*. Berlin, Heidelberg: Springer-Verlag, pp. 125–

139. ISBN: 978-3-540-85484-5. DOI: http://dx.doi.org/10.1007/978-3-540-85485-2_10.

Resinas, Manuel, Pablo Fernandez, and Rafael Corchuelo (2006). "A Conceptual Framework for Automated Negotiation Systems". In: *Proceedings of the 7th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2006)*, pp. 1250–1258. DOI: 10.1007/11875581_148.

Resinas, Manuel, Pablo Fernandez, and Rafael Corchuelo (2010). "Automatic Service Agreement Negotiators in Open Commerce Environments". In: *International Journal of Electronic Cornmerce* 14.3, pp. 93–128.

Rolli, Daniel et al. (2006). "A Descriptive Auction Language". In: *Electronic Markets* 16.1, pp. 51–62. DOI: 10.1080/10196780500491436.

Rosenberg, Igor and Ana Juan (2009). *White Paper: Integrating an SLA Architecture based on Components*. Project White Paper. last checked: 2011-05-03. BEinGRID Consortium. URL: http://www.it-tude.com/slawhitepaper.html.

Royce, W. (1987). "Managing the Development of Large Software Systems: Concepts and Techniques". In: *Proceedings of the 9th international conference on Software Engineering (ICSE '87), Los Alamitos, CA, USA*. Monterey, California, United States: IEEE Computer Society Press, pp. 328–338. ISBN: 0-89791-216-0.

Ruggaber, Rainer (2007). "Internet of Services SAP Research Vision". In: *Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '07), Washington, DC, USA*. IEEE Computer Society, p. 3. ISBN: 0-7695-2879-1. DOI: http://dx.doi.org/10.1109/WETICE.2007.152.

Russell, Stuart and Peter Norvig (1995). *Artificial Intelligence - A Modern Approach*. Ed. by Stuart Russell and Peter Norvig. Prentice Hall, Upper Saddle River, New Jersey.

SORMA (2007). *SORMA Deliverable 1.2: Framework for Self-Organizing Resource Management*. Project Deliverable. last checked: 2011-05-03. SORMA Consortium. URL: http://www.im.uni-karlsruhe.de/sorma/deliverables.htm.

SORMA (2008). *SORMA Deliverable 2.2: Final Specification and Design Documentation of the SORMA Components*. Project Deliverable. last checked: 2011-05-03. SORMA Consortium. URL: http://www.im.uni-karlsruhe.de/sorma/deliverables.htm.

SORMA (2009). *SORMA Deliverable 5.2: Economic Middleware and Grid Operating System Extensions*. Project Deliverable. last checked: 2011-05-03. SORMA Consortium. URL: http://www.im.uni-karlsruhe.de/sorma/deliverables.htm.

Schaaf, Thomas (2008). "IT-gestütztes Service-Level-Management - Anforderungen und Spezifikation einer Managementarchitektur". last checked: 2011-05-04. PhD thesis. Ludwig-Maximilians-University Munich. URL: http://edoc.ub.uni-muenchen.de/9534/.

Scheithauer, Gregor and Matthias Winkler (2008). *A Service Description Framework for Service Ecosystems*. Bamberger Beitraege zur Wirtschaftsinformatik und Angewandten Informatik 78. last checked: 2011-05-03. University of Bamberg. URL: http://www.opus-bayern.de/uni-bamberg/volltexte/2009/173/.

Schnizler, Björn et al. (2005a). *Environmental Analysis for Application Layer Networks*. Project Deliverable. last checked: 2011-05-03. CATNETS Consortium. URL: http://www.catnets.uni-bayreuth.de/index.php?id=13.

Schnizler, Björn et al. (2005b). *First Year Report of WP1: Theoretical and Computational Basis*. Project Deliverable. last checked: 2011-05-03. CATNESTS Consortium. URL: http://www.catnets.uni-bayreuth.de/index.php?id=13.

Schopf, Jennifer M. and Bill Nitzberg (2002). "Grids: The top ten questions". In: *Scientific Programming* 10 (2). last checked: 2011-05-04, pp. 103–111. ISSN: 1058-9244.

Schroth, Christoph and Till Janner (2007). "Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services". In: *IT Professional* 9.3, pp. 36–41. ISSN: 1520-9202. DOI: http://doi.ieeecomputersociety.org/10.1109/MITP.2007.60.

Seidel, Jan et al. (2007). *Using SLA for Resource Management and Scheduling - A Survey*. Technical Report TR-0096. last checked: 2011-05-03. CoreGRID. URL: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0096.pdf.

Silaghi, Gheorghe Cosmin, Alvaro E. Arenas, and Luis Moura Silva (2007). *Reputation-based Trust Management Systems and their Applicability to Grids*. Technical Report TR-0064. last checked: 2011-05-03. CoreGrid. URL: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0064.pdf.

Simon, Herbert Alexander (1996). *The Sciences of the Artificial*. Cambridge, Massachusetts: MIT Press. ISBN: 978-0-262-69191-8.

Skene, J., D. Lamanna, and W. Emmerich (2004). "Precise Service Level Agreements". In: *Proceedings of the 26th International Conference on Software Engineering, Edinburgh, United Kingdom*. IEEE Computer Society Press, pp. 179–188.

Smith, Adam (1976). *An Inquiry into the Nature and Causes of the Wealth of Nations*. Ed. by R. H. Campbell, A. S. Skinner, and W. B. Todd. First published 1776. Oxford University Press.

Smith, R. G. (1980). "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". In: *IEEE Transactions on Computers* 29.12, pp. 1104–1113. ISSN: 0018-9340. DOI: http://doi.ieeecomputersociety.org/10.1109/TC.1980.1675516.

Snelling, D. and A. Anjomshoaa (2007). *NextGRID Architectural Concepts*. Technical Report. last checked: 2011-05-03. NextGRID. URL: http://eprints.ecs.soton.ac.uk/17602/1/09.pdf.

Snelling, Dave, Mike Fisher, and Achim Basermann (2005). *An Introduction to the NextGRID Vision and Achievements V1.0*. Technical Report. last checked: 2011-05-03. NextGRID. URL: http://www.nextgrid.org/download/publications/NextGRID_Architecture_White_Paper.pdf.

Srinivasan, Latha and Jem Teadwell (2005). *An Overview of Service-oriented Architecture, Web Services and Grid Computing*. Technical Report. last checked: 2011-05-03. HP Software Global Business Unit. URL: http://h71028.www7.hp.com/ERC/downloads/SOA-Grid-HP-WhitePaper.pdf.

Steinmetz, Ralf and Klaus Wehrle (2004). "Peer-to-Peer-Networking & -Computing". In: *Informatik - Spektrum* 27.1, pp. 51–54. DOI: 10.1007/s00287-003-0362-9.

Stoica, Ion et al. (2001). "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". In: *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA*. San Diego, California, United States: ACM, pp. 149–160. ISBN: 1-58113-411-8. DOI: http://doi.acm.org/10.1145/383059.383071.

Ströbel, M. (2001). *Design of Roles and Protocols for Electronic Negotiations*. last checked: 2011-05-04. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.7302.

Ströbel, Michael (2000a). "Effects of Electronic Markets on Negotiation Processes". In: *Proceedings of the 8th European Conference on Information Systems*. Vol. 1, pp. 445–452.

Ströbel, Michael (2000b). "On Auctions as the Negotiation Paradigm of Electronic Markets". In: *Electronic Markets* 10.1, pp. 39–44. DOI: 10.1080/10196780050033962.

Ströbel, Michael and Christof Weinhardt (2003). "The Montreal Taxonomy for Electronic Negotiations". In: *Group Decision and Negotiation* 12.2, pp. 143–164.

Strecker, Stefan et al. (2006). "Electronic negotiation systems: The Invite prototype". In: *Multikonferenz Wirtschaftsinformatik 2006*. Gesellschaft für Informatik e.V. Berlin: GITO, pp. 315–331.

Streitberger, Werner et al. (2008). "On the Simulation of Grid Market Coordination Approaches". In: *Journal of Grid Computing, Special Issue on Grid Economics and Business Models* 6.3. ISSN: 1570-7873 (Print) 1572-9814 (Online), pp. 349–366. DOI: http://dx.doi.org/10.1007/s10723-007-9092-6.

Tamma, V., M. Wooldridge, and I. Dickinson (2002). *An Ontology for Automated Negotiation*. last checked: 2011-05-04. URL: http://www.computer.org/portal/web/csdl/doi/10.1109/SNPD.2007.27.

Taylor, Steve et al. (2009). "Business Collaborations in Grids: The BREIN Architectural Principals and VO Model". In: *Proceedings of the 6th International Workshop on Grid Economics and Business Models (GECON '09)*. Berlin, Heidelberg: Springer-Verlag, pp. 171–181. ISBN: 978-3-642-03863-1. DOI: http://dx.doi.org/10.1007/978-3-642-03864-8_14.

Terracina, Annalisa et al. (2007). *Akogrimo Deliverable 4.3.4 - Consolidated Report on the Implementation of the Infrastructure Services Layer*. Project Deliverable. last checked: 2011-05-03. Akogrimo Consortium. URL: http://www.akogrimo.org/modules235a.html?name=UpDownload&req=viewdownload&cid=5.

Theilmann, Wolfgang et al. (2009). *SLA@SOII Deliverable D.A1a - Framework Architecture*. Project Deliverable. last checked: 2011-05-03. SLA@SOI Consortium. URL: http://sla-at-soi.eu/results/deliverables/.

Theilmann, Wolfgang et al. (2010). *SLA@SOI Deliverable D.A1a - Framework architecture (full lifecycle)*. Project Deliverable. last checked: 2011-05-03. SLA@SOI Consortium. URL: http://sla-at-soi.eu/results/deliverables/.

Theseus (2009). *TEXO – Business Webs im Internet der Dienste*. last checked: 2011-05-04, (german). URL: http://theseus-programm.de/anwendungsszenarien/texo/default.aspx.

Thompson, Henry S. et al. (2004). *XML Schema Part 1: Structures, Second Edition*. last checked: 2011-05-04. World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028.

Tosic, Vladimir, Kruti Patel, and Bernard Pagurek (2002). "WSOL - Web Service Offerings Language". In: *Lecture Notes in Computer Science* 2512/2002. ISSN 0302-9743, pp. 57–67.

Tosic, Vladimir et al. (2002). "Web Service Offerings Language (WSOL) and Web Service Composition Management (WSCM)". In: *Proceedings of the Object- Oriented Web Services Workshop at the Conference on Object-Oriented Programming, Systems, Languages & Applications 2002 (OOPSLA 2002), Seattle, USA*.

Tosic, Vladimir et al. (2004). "Web Service Offerings Infrastructure (WSOI) - a management infrastructure for XML Web services". In: *Proceedings of the Network Operations and Management Symposium (NOMS 2004)*. Vol. 1, pp. 817–830.

Treadwell, J. (2007). *Open Grid Services Architecture, Glossary of Terms, Version 1.6*. Information Document. last checked: 2011-05-04. Open Grid Forum. URL: http://www.ogf.org/documents/GFD.120.pdf.

Unified Modelling Language (UML) 2.0 (2005). Object Management Group. URL: http://www.omg.org/spec/UML/.

Veit, Daniel et al. (2007). *Third Year Report of WP 1: Theoretical and Computational Basis*. Project Deliverable. last checked: 2011-05-03. CATNETS Consortium. URL: http://www.catnets.uni-bayreuth.de/index.php?id=13.

Venugopal, Srikumar, Xingchen Chu, and Rajkumar Buyya (2008). "A Negotiation Mechanism for Advance Resource Reservations Using the Alternate Offers Protocol". In: *Proceedings of the 16th International Workshop on Quality of Service (IWQoS 2008)*, pp. 40–49. DOI: 10.1109/IWQOS.2008.10.

Waeldrich, Oliver et al. (2010). "WS-Agreement Negotiation". In: last checked: 2010-10-05. URL: https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.graap-wg/docman.root.current_drafts.ws_agreement_negotiation_specifi/doc16041/1.

Walls, Joseph G., George R. Widmeyer, and Omar A. El Sawy (1992). "Building an Information System Design Theory for Vigilant EIS". In: *Information Systems Research* 3.1, pp. 36–59. DOI: 10.1287/isre.3.1.36. eprint: http://isr.journal.informs.org/cgi/reprint/3/1/36.pdf.

Walsh, William E., Michael P. Wellman, and Fredrik Ygge (2000). "Combinatorial Auctions for Supply Chain Formation". In: *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00)*. New York, NY, USA: ACM, pp. 260–269. ISBN: 1-58113-272-7. DOI: http://doi.acm.org/10.1145/352871.352900.

Wesner, Stefan et al. (2005). *Akogrimo Deliverable 2.2.1 Volume 1 – The Socio-Economic Environment*. Project Deliverable. last checked: 2011-05-03. Akogrimo Consortium. URL: http://www.akogrimo.org/modules235a.html?name=UpDownload&req=viewdownload&cid=5.

Westerinen, A. et al. (2001). *Terminology for Policy-Based Management*. Request for Comments. last checked: 2011-05-04. Internet Engineering Task Force (IETF). URL: http://www.ietf.org/rfc/rfc3198.txt.

Wettig, Steffen and Eberhard Zehendner (2004). "A Legal Analysis of Human and Electronic Agents". In: *Artificial Intelligence and Law* 12.1, pp. 111–135. ISSN: 0924-8463. DOI: http://dx.doi.org/10.1007/s10506-004-0815-8.

Wilde, Thomas and Thomas Hess (2007). "Forschungsmethoden der Wirtschaftsinformatik - Eine empirische Untersuchung". In: *Wirtschaftsinformatik* 49.4, pp. 280–287.

Wilson, Michael (2007). *TrustCoM Project Final Report*. Project Deliverable. last checked: 2011-05-03. TrustCoM Consortium. URL: http://epubs.cclrc.ac.uk/bitstream/5109/trustcom-finalreport_en.pdf.

Wooldridge, M. (1997). "Agent-based Software Engineering". In: *IEEE Proceedings Software Engineering* 144.1, pp. 26–37. DOI: http://dx.doi.org/10.1049/ip-sen:19971026.

Wooldridge, Michael J. (2005). *An Introduction to MultiAgent Systems*. Chichester: John Wiley & Sons, Ltd.

Wurman, P., W. Walsh, and M. Wellman (1998a). *Flexible Double Auctions for Electronic Commerce: Theory and Implementation*. last checked: 2011-05-04. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1798.

Wurman, Peter R., William E. Walsh, and Michael P. Wellman (1998b). "Flexible Double Auctions for Electronic Commerce: Theory and Implementation". In: *Decision Support Systems* 24, pp. 17–24.

Wurman, Peter R., Michael P. Wellman, and William E. Walsh (1998). "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents". In: *Second International Conference on Autonomous Agents (Agents'98)*.

Wurman, Peter R., Michael P. Wellman, and William E. Walsh (2001). "A Parametrization of the Auction Design Space". In: *Games and Economic Behavior* 35.1-2, pp. 304–338.

Wurman, Peter R., Michael P. Wellman, and William E. Walsh (2002). "Specifying Rules for Electronic Auctions". In: *AI Magazine* 23, p. 2002.

Ziegler, Wolfgang et al. (2008). *Considerations for negotiation and monitoring of Service Level Agreements*. Technical Report TR-0167. last checked: 2011-05-03. CoreGRID. URL: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0167.pdf.

d'Andria, Francesco et al. (2006). "Exploiting the Knowledge Economy". In: ed. by Paul Cunningham and Miriam Cunningham. IOS Press. Chap. An Enhanced Strategy for SLA Management in the Business Context of New Mobile Dynamic VO.

Curriculum Vitae

Personal Information

Sebastian Hudert
Preuschwitzer Str. 23
95445 Bayreuth

Email: sebastian.hudert@gmail.com

Diplom-Wirtschaftsinformatiker (Univ.)
Born on March 3rd 1981 in Schweinfurt, Germany
Marital status: unmarried
Nationality: German

Vita

since 11/2006	**Research and Teaching Assistant and PhD Candidate**, Department of Information Systems Management (BWL VII), Prof. Dr. Torsten Eymann, University of Bayreuth
10/2001 - 09/2006	**Academic Studies in Information Systems** (Diploma, equivalent to a Master's Degree) at the Otto-Friedrich University Bamberg
09/2005 - 01/2006	**Exchange semester** at the Dublin City University, Dublin, Ireland
09/2000 - 07/2001	**Civilian Service** at the Kolping Educational Center Schweinfurt
06/2000	**School Leaving Examination (Abitur)** at the Walther Rathenau Gymnasium Schweinfurt

Bayreuth, October 25, 2011

Visions of the next-generation Internet of Services are driven by digital resources traded on a global scope. For the resulting economic setting, automated on-line techniques for handling services and resources themselves, for advertising and discovering as well as for the on-the-fly negotiation of proper terms for their use are needed. Hence, a flexible infrastructure for the respective management of services and associated service level agreements is mandatory.

This thesis presents the results of my dissertation project. They comprise a service infrastructure, able to support the structured discovery and protocol-generic negotiation of electronic service level agreements (SLAs) and thus services themselves.

ISSN 1864-9300