Multiagent resource allocation in service networks

A dissertation
submitted to the Faculty of Business, Economics and Social Sciences of
the University of Hohenheim

in partial fulfillment of the requirements for the degree of
Doctor of Economics (Dr. oec.)

by

Paul Karänke

March 2014

Principal Adviser: Prof. Dr. Stefan Kirn
Co-Adviser: Prof. Dr. Frank Leymann
Chair of the Examination Committee: Prof. Dr. Christian Ernst
Dean: Prof. Dr. Dirk Hachmeister
Date of the Oral Examination: March 17, 2014

Abstract

The term *service network* (SN) denotes a network of software services in which complex software applications are provided to customers by aggregating multiple elementary services. These networks are based on the *service-oriented computing* (SOC) paradigm, which defines the fundamental technical concepts for software services over electronic networks, e.g., Web services and, most recently, Cloud services. For the provision of software services to customers, software service providers (SPs) have to allocate their scarce computational *resources* (i.e., hardware and software) of a certain quality to customer requests. The SOC paradigm facilitates interoperability over organizational boundaries by representing business relationships on the software system level. *Composite software services* aggregate multiple software services into software applications. This aggregation is denoted as service composition. The loose coupling of services leads to SNs as dynamic entities with changing interdependencies between services.

For composite software services, these dependencies exist across SN tiers; they result from the procurement of services, which are themselves utilized to produce additional services, and constitute a major problem for resource allocation in SNs. If these dependencies are not considered, the fulfillment of agreements may become unaccomplishable (*overcommitment*). Hence, the consideration of service dependencies is crucial for the allocation of service providers' resources to fulfill customer requests in SNs.

However, existing resource allocation methods, which could consider these dependencies – such as combinatorial auctions with a central auctioneer for the whole SN – are not applicable, since there are no central coordinating entities in SNs. The application of an allocation mechanism that does not consider these dependencies might negatively affect the actual service delivery; results are penalty payments as well as a damage to the reputation of the providers.

This research is conducted in accordance to the design science paradigm in information system research. It is a problem-solving paradigm, which targets the *construction and evaluation* of IT artifacts. The objectives of this research are to develop and evaluate an allocation protocol, which can consider multi-tier service dependencies without the existence of central coordinating entities. Therefore, an interaction protocol engineering (IPE) perspective is applied to solve the problem of multi-tier dependencies in resource allocation. This approach provides a procedure model for designing interaction protocols

for multiagent systems, and is closely related to the well-established area of communication protocol engineering.

Automated resource allocation in SNs is analyzed in this research by representing the actors as *autonomous software agents* in the software system. The actors delegate their objectives to their software agents, which conduct the negotiations for service provision on their behalf. Thus, these agents communicate concerning the resource allocation; in this process, the sequence of communication interactions is crucial to the problem addressed. Interaction protocols define a structured exchange of defined messages between agents; they facilitate agent conversations.

When multiple agents have to reach agreements by negotiation and bargaining, such as in case with allocating scarce resources, game theory provides means to formalize and analyze the most rational choice of actions for the interacting agents. Based on a formal framework for resource allocation in SNs, this research first performs a game-theoretic problem analysis; it is concerned with the existence, as well as the complexity of computing optimal allocations. In addition, Nash equilibria are analyzed for optimal allocations. Second, a distributed, auction-based allocation protocol, which prevents overcommitments and guarantees socially optimal allocations for single customer requests under certain assumptions, is proposed. Therefore, a game-theoretic model and an operationizable specification of the protocol are presented. Third, it is formally verified that the protocol enables multi-tier resource allocation and avoids overcommitments by proofs for the game-theoretic model and by model checking for the interaction protocol specification; using the model checker SPIN, safety properties like the absence of deadlock are as well formally verified as the protocol enabling multi-tier resource allocation. Fourth, the efficacy and the benefits of the proposed protocol are demonstrated by multiagent simulation for concurrent customers. The experimental evaluation provides evidence of the protocol's efficiency compared to the socially optimal allocation as a centralized benchmark in different settings, e.g., network topologies and different bidding policies.

Acknowledgments

Sincere thanks to Julia for her dedicated support.

I thank my dissertation adviser Stefan Kirn for the supervision and for making this work possible. I also thank my co-adviser Frank Leymann. Thanks go to my colleagues in Hohenheim, especially to Jörg Leukel for his scientific advice. Last but not least, I thank my parents and my family.

Contents overview

Contents

List of Figures

List of Tables

List of abbreviations

AI	artificial intelligence
BDI	belief-desire-intention
BPMN	business process model and notation
BPRO	best price resources only
CFP	call for proposals
CNP	Contract net protocol
CP	composition pattern
DAI	distributed artificial intelligence
DPS	distributed problem solving
ERF	external resources first
FSM	finite state machine
IaaS	infrastructure as a service
IPE	interaction protocol engineering
IRF	internal resources first
IS	information system
LTL	linear temporal logic
MARA	multiagent resource allocation
MAS	multiagent system
MTCNP	Multi-tier contract net protocol
NE	Nash equilibrium
PaaS	platform as a service
QoS	quality of service
SaaS	software as a service
SLA	service level agreement
SN	service network
SOA	service-oriented architecture
SOC	service-oriented computing
SP	service provider
WS	Web service
WS-BPEL	Web services business process execution language

List of symbols

$\mathcal{A}$	set of agents
$\mathcal{A}_C \subseteq \mathcal{A}$	set of customer agents
$\mathcal{A}_{SP} \subseteq \mathcal{A}$	set of service provider agents
Λ	set of service provider agent tiers
$\lambda \in \Lambda$	service provider agent tier
$\mathcal{A}_{SP,\lambda}$	set of service providers in tier λ
$i, j, k, \ell, m, n \in \mathbb{N}_0$	indices
$a_i \in \mathcal{A}$	agent i
Ω	world state
$\Omega_i \subseteq \Omega$	environment of agent a_i
OF_i	delegated objective function of agent a_i
BEL_i	beliefs of agent a_i
$GOAL_i$	goals of agent a_i
Σ_i	set of possible actions of agent a_i
CT_i	control thread of agent a_i
$per_i : \Omega_i \times OF_i \to BEL_i$	perception function of agent a_i
$goal_i : BEL_i \times OF_i \times GOAL_i \to GOAL_i$	goal function of agent a_i
$act_i : GOAL_i \times OF_i \times \Sigma_i \to \Sigma_i$	action function of agent a_i
$\mathcal{T} = \{0, \ldots, T\}$	set of time periods
$T \in \mathcal{T}$	last time period
$t \in \mathcal{T}$	time period
$\mathcal{S}$	set of services
$\mathscr{P}(\mathcal{S})$	powerset of the set $\mathcal{S}$
$\mathcal{S}^t \subseteq \mathcal{S}$	set of contracted services in period t
$s_{ij} \in \mathcal{S}$	service provided by a_j to a_i
$v_i : \mathcal{S} \to \mathbb{R}$	valuation function of agent a_i
$c_j : \mathcal{S} \to \mathbb{R}$	cost function of agent a_j
$\mathcal{R}$	set of resources
$\mathcal{R}_j \subseteq \mathcal{R}$	set of resources of agent a_j

$\mathcal{O}$	set of offers
$\mathcal{O}^t \subseteq \mathcal{O}$	set of offers in period t
$o_{ij}^t \in \mathcal{O}$	offer from a_i for service s_{ij} in period t
$\mathcal{B}$	set of bids
$\mathcal{B}^t \subseteq \mathcal{B}$	set of bids in period t
$b_{ij}^t \in \mathcal{B}$	bid from a_j for service s_{ij} in period t
$w_j : \mathcal{S} \times \mathcal{R}_j \to \mathbb{R}$	capacity function of agent a_j
W_j	total capacity of agent a_j
$\psi : \mathcal{O} \times \mathcal{B} \to \mathbb{R}$	payment function
$x^t : \mathcal{S} \to \{0, 1\}$	allocation function for period t
$\varphi : \mathcal{S} \to \mathscr{P}(\mathcal{S})$	service dependency function
$z^t : \mathcal{S} \to \{0, 1\}$	dependency allocation function for period t
γ	penalty factor
$\rho_i^t : \mathcal{S} \to \mathbb{R}$	penalty payment function for a_i in period t

Chapter 1

Introduction

1.1 Resource allocation in service networks

The term *service network* (SN) denotes a network of software services in which complex software applications are provided to customers by aggregating multiple elementary services. These networks are based on the *service-oriented computing* (SOC) paradigm, which provides the fundamental technical concepts for software services over electronic networks, e.g., Web services and, most recently, Cloud services (Blau, van Dinther, Conte, Xu & Weinhardt 2009, Armbrust, Fox, Griffith, Joseph, Katz, Konwinski, Lee, Patterson, Rabkin, Stoica & Zaharia 2010). Studies suggest that Cloud-based SNs are gaining widespread acceptance and may provide economic benefits for businesses as well as entire economies (e.g., (Alford & Morton 2009, Hogan & Mohamed 2010, Vehlow & Golkowsky 2011, Alcatel-Lucent 2012)).

SOC enables the loose coupling of software services to provide composite software application services to customers. The main objective of the SOC paradigm is to facilitate interoperability over organizational boundaries, representing real-world business relationships on the software system level. *Composite software services* aggregate multiple software services into software applications. This aggregation is denoted as service composition. The loose coupling of services leads to SNs as dynamic entities with changing interdependencies between services (Papazoglou, Traverso, Dustdar & Leymann 2008).

The provision of software services in SNs requires non-consumable *resources* (hardware and software) that are limited and possessed by service providers (SPs). The SPs' resources contribute to the production of service applications in SNs by means of service provision and composition. For the supply of software services to customers, software SPs have to allocate their scarce computational resources of a certain quality to customer requests, i.e., re-

serve respective resources for software services on the contracted service level. In return, SPs receive monetary compensations for providing services to customers. Supply and demand for services are assigned by an *allocation mechanism* (Rasmusen 1989, Binmore 1992). Thus, the allocation mechanism of SPs' resources to fulfill customer requests is of major importance for SNs.

SNs can be regarded as a special kind of digital supply chains with a network structure, in which nodes constitute the software systems of actors – SPs and customers – and edges constitute formal representations of business relationships for software services. This research analyzes automated resource allocation in SNs by representing the actors as *autonomous software agents* (Wooldridge & Jennings 1995, Jennings 2000, Russell & Norvig 2003) in the software system. The actors delegate their objectives to their software agents, which conduct the negotiations for service provision on their behalf (Huhns, Singh, Burstein, Decker, Durfee, Finin, Gasser, Goradia, Jennings, Kiran Lakkaraju Nakashima, Van Dyke Parunak, Rosenschein, Ruvinsky, Sukthankar, Swarup, Sycara, Tambe, Wagner & Zavafa 2005). The representation of real-world business relationships requires formal contractual agreements. Explicit formal statements of the obligations and guarantees regarding software services in a business relationship are referred to as service level agreements (SLAs) (Verma 1999). A SLA defines a software service as part of a contract between a SP and a service customer in a SN. SOC-related SLA approaches aim at providing an abstraction of the service while facilitating measurement and monitoring of service properties agreed upon (Czajkowski, Foster & Kesselman 2004, pp. 264–265). Thus, agreements between software agents are represented by *SLA specifications*. Figure 1.1 shows a use case diagram for automated resource allocation in SNs in accordance to the actors' delegated objectives.

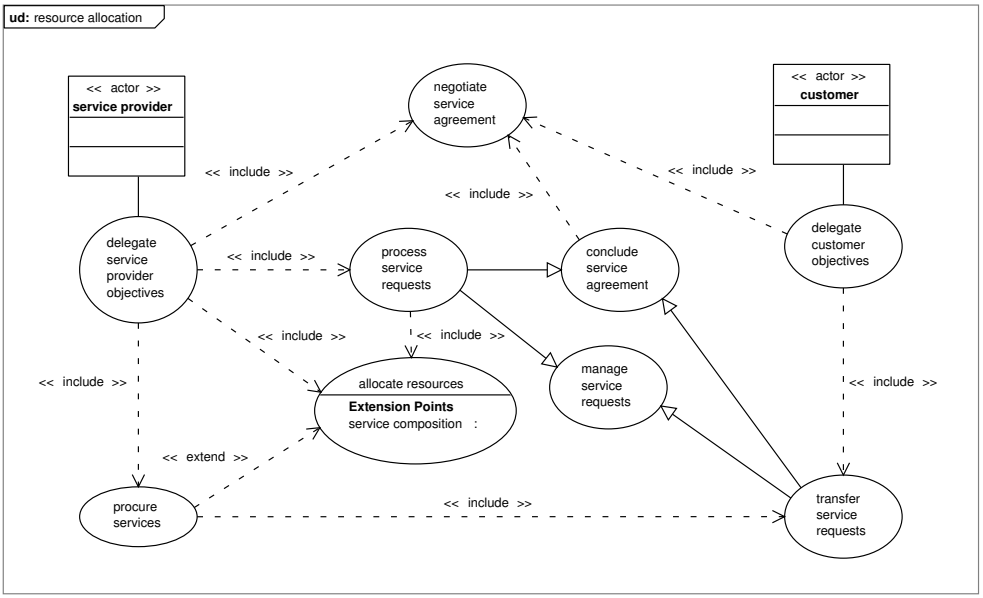

Figure 1.1: Resource allocation in service networks use case diagram.

For composite software services, service dependencies exist across SN tiers; they result from the procurement of services, which are themselves utilized to produce additional services, and constitute a major problem for resource allocation in SNs. The individual requirements of the service customer determine the requirements of agreements that have to be established in upstream SN tiers. This results in networks of agreements with service dependencies. If these dependencies are not considered, the fulfillment of agreements may become unaccomplishable (*overcommitment*). The application of an allocation mechanism that does not respect these dependencies might negatively affect the actual service delivery; results are penalty payments as well as a damage to the reputation of the providers. Hence, the consideration of service dependencies is crucial for the allocation of service providers' resources to fulfill customer requests in SNs.

However, existing resource allocation methods, which could respect these dependencies – such as combinatorial auctions with a central auctioneer for the whole SN – are not applicable, since there are no central coordinating entities in SNs. In addition, finding optimal allocations is often computationally infeasible in this setting (Bo & Lesser 2010). Thus, the objectives of this research are to develop and evaluate an allocation protocol which is able to consider multi-tier service dependencies without the existence of central coordinating entities.

Example As an illustrative example, a scenario with two customer agents is shown in figure 1.2. The service requested by the customer agents are data mining services in the same time frame, which are provided by SP agents over the network (software as a service, SaaS). The customer agents' requests include definitions of the services and respective parameters (required capacities in this example). The SP agents have to subcontract parts of the service to, respectively procure appropriate sub-services from, their suppliers in the next tier. In this example, both data mining SP agents require a specialized analysis service for their data mining service software. In addition, one of the data mining services along with the analysis service require data storage services.

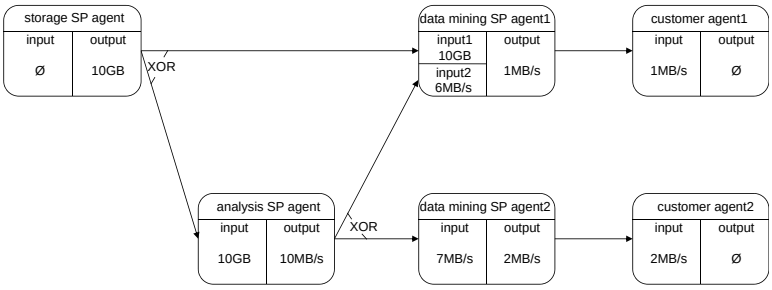

Figure 1.2: Service network example.

The multi-tier resource allocation has to ensure that agreements will either be established with the customer agents *and* with respective supplier agents in all tiers required or no agreements will be established at all. If a binding agreement between a SP agent and a customer agent is established before binding procurement agreements with the SP agents in the next tier are established, the SP agents may be unable to fulfill the agreements with the customer agents.

In this example, it is assumed that the storage SP agent can *either* provide its service to the analysis SP agent *or* data mining SP agent1, since the required capacity of the latter two exceeds the available capacity of the storage SP agent (10GB + 10GB > 10GB). Similarly, both data mining SP agent1 and data mining SP agent2 require the specialized analysis service. However, the analysis SP agent has insufficient capacity to serve both agents (6MB/s + 7MB/s > 10MB/s).

In this example scenario, just two possible allocations are feasible. Figure 1.3 shows the two feasible allocations of the example scenario in black, while unallocated services are depicted in gray.

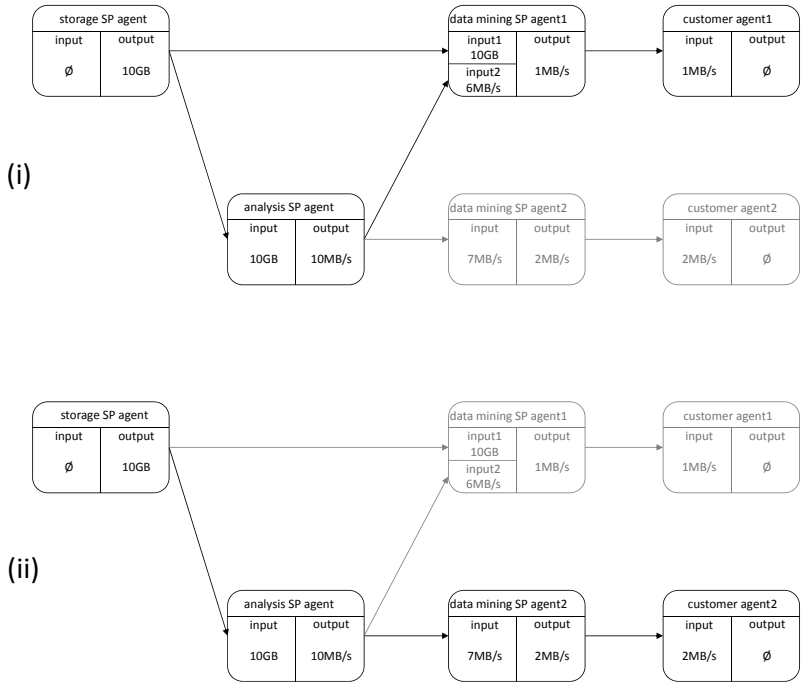

Figure 1.3: Feasible allocations example.

The allocation mechanism has to ensure that

1. data mining SP agent2 does not establish an agreement with customers agent2 if it cannot procure the analysis service (figure 1.3, (i)),
2. data mining SP agent1 does not establish an agreement with customers agent1 if it cannot procure both the analysis and storage services (figure 1.3, (ii)), and
3. the data mining SP agents must not purchase services from their upstream provider agents if the purchased services cannot be used for production by the data mining SP agents.

1.2 Research approach

Resource allocation is a well-established field in multiagent research (Chevaleyre, Dunne, Endriss, Lang, Lemaitre, Maudet, Padget, Phelps, Rodriguez-aguilar & Sousa 2006). Each actor in a SN (service providers and customers) is represented by an autonomous software agent (Huhns et al. 2005). These agents

communicate concerning the resource allocation in SNs. The sequence of communication interactions is crucial to the problem addressed, as stated in section 1.1. Interaction protocols define a structured exchange of defined messages between agents; they facilitate agent conversations (Huhns & Stephens 1999, Wooldridge 2009).

This research studies multi-tier dependencies in resource allocation from an interaction protocol engineering (IPE) perspective (Huget & Koning 2003). IPE is concerned with designing interaction protocols for multiagent systems, and is closely related to communication protocol engineering (Holzmann 1991). Although there are conceptual differences between the assumptions for communicating processes in distributed systems and communicating agents, especially regarding the level of abstraction (Jennings & Wooldridge 2000), prior research in multiagent systems suggests that methods and tools from the communication protocol area are also useful for technical aspects of developing and validating agent interaction protocols (Walton 2007, Giordano, Martelli & Schwind 2007); basic requirements apply for both communication and interaction protocols (e.g., absence of deadlock and unreachable states).

The IPE approach comprises five phases. In the analysis phase (1), this research studies resource allocation with regard to dependencies between different SN tiers and formally analyzes these dependencies. When multiple agents have to reach agreements by negotiation and bargaining, such as in case with allocating scarce resources, game theory (von Neumann & Morgenstern 1944, Rasmusen 1989, Binmore 1992) provides means to formalize and analyze the most rational choice of actions for the interacting agents (Rosenschein 1985, Rosenschein & Zlotkin 1994, Kraus 1997, Parsons & Wooldridge 2002). Based on a formal framework for resource allocation in SNs, this research performs a game-theoretic problem analysis; it is concerned with the existence, as well as the complexity of computing optimal allocations. In addition, Nash equilibria (Nash 1950) are investigated for optimal allocations.

In the formal description phase (2), the protocol is constructed and formally specified. In this thesis, a distributed, auction-based allocation protocol, which prevents overcommitments and guarantees socially optimal allocations for single customer requests under certain assumptions, is proposed. A game-theoretic specification of the protocol is provided, based on the formal framework. In addition, a specification is presented in UML sequence diagrams. The model checker (Clarke, Grumberg & Peled 1999) SPIN (Holzmann 1997) accepts protocol specifications in the verification language PROMELA (a Process Meta Language) (Holzmann 1991). Towards a formal description of the protocol in

PROMELA, finite state machines (FSMs) are constructed for the participants. Thus, states are assigned to the participants for every (alternative) interaction on the basis of the UML sequence diagrams.

The fulfillment of the requirements by the formal descriptions is evidenced in the verification phase (3). This research proves that the proposed protocol prevents overcommitments and guarantees socially optimal allocations for single customer requests under the assumption, that resources are non-substitutable between SPs, based on the game-theoretic model. Further, the model checker SPIN is applied to verify certain formal properties of the protocol description in PROMELA. These include safety (e.g., absence of deadlock) as well as liveness properties of the protocol, expressed in linear temporal logic (LTL). Thus, it is verified that the protocol allows multi-tier resource allocation.

The implementation phase (4) comprises the construction of an executable protocol. The proposed protocol is implemented as a reusable capability for the Jadex BDI agent system (Pokahr, Braubach & Lamersdorf 2005). The simulation experiments are executed using this implementation, along with Jadex BDI agents representing the actors in a dedicated simulation system.

Finally, the conformance test phase (5) validates the conformance of the implemented protocol with the properties defined in the analysis phase (Huget & Koning 2003). The efficacy and the benefits of the proposed protocol are demonstrated through multiagent simulation (Kluegl 2001) of different Cloud computing scenarios. In the simulation experiments, the efficiency of the protocol is compared to the optimal allocation as a centralized benchmark in different settings (e.g., network topologies or number of customer and SP agents) for three different SP bidding policies. Further, the experiments are executed with and without substitutable resources. SP agents can decide if they use own resources or procure sub-services from SP agents in the next tier in case of substitutable resources. Table 1.1 summarizes the methods applied.

Table 1.1: Methods used.

#	**IPE phase**	**methods used**
1	analysis	game theory
2	formal description	game theory, UML, FSMs, PROMELA (LTL)
3	verification	proofs, model checking (SPIN)
4	implementation	Jadex (java)
5	conformance test	multiagent simulation

1.3 Epistemological position

This research is conducted in accordance to the design science paradigm in information systems research proposed by Hevner, March, Park & Ram (2004). Research in the information systems discipline is mainly characterized by two paradigms: The behavioral-science approach – based on natural science research principles – aims at developing and justifying theories that explain or predict organizational and human behavior, regarding the utilization of information systems. In contrast, the design science paradigm – based on engineering and the sciences of the artificial (Simon 1996) – is a problem-solving paradigm. It targets the *construction and evaluation* of IT artifacts, enabling organizations to address information-related tasks (Hevner et al. 2004).

Hevner et al. (2004) propose seven guidelines for effective information system research; these are shown in table 1.2 and discussed with regard to this work in the following.

Table 1.2: Design-science research guidelines.
(Hevner et al. 2004, p. 83)

guideline	**description**
guideline 1: design as an artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
guideline 2: problem relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
guideline 3: design evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
guideline 4: research contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
guideline 5: research rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
guideline 6: design as a search process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
guideline 7: communication of research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Guideline 1: design as an artifact Design-science research in information systems results in purposeful IT artifacts addressing important organizational problems. IT artifacts are defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems) (Hevner et al. 2004, p. 77). In this thesis, a distributed resource allocation protocol is constructed as a method artifact. Further, an implementation of this protocol is presented as an instantiation artifact.

Guideline 2: problem relevance This thesis addresses multi-tier service dependencies in resource allocation, i.e., distributed resource allocation without the existence of central coordinating entities. The problem consists of contractual dependencies along the complete SN. If these dependencies are not considered, the fulfillment of agreements may be unaccomplishable, due to missing agreements with other actors; however, these agreements are required for the fulfillment (avoiding *overcommitment*). The application of an allocation mechanism that does not respect these dependencies might negatively affect the actual service delivery; results are penalty payments as well as a damage to the reputation of the providers.

Guideline 3: design evaluation The evaluation within the design science framework has to demonstrate the utility, quality, and efficacy of the constructed artifact by means of appropriate methods (Hevner et al. 2004, pp. 85–87). This research verifies the formal specification of the proposed protocol (i) by proofs and (ii) by model checking regarding safety (e.g., absence of deadlock), as well as liveness properties. The model checker SPIN (Holzmann 1997) is applied to verify that the protocol specification is sound and allows multi-tier resource allocation. This work evaluates the protocol implementation experimentally by multiagent-based simulation (Kluegl 2001), motivated by real-world application scenarios from the Cloud-computing domain.

Guideline 4: research contributions The developed artifact has to be innovative and provide a clear contribution; i.e., it has to be novel in terms of solving a previously unsolved problem or it has to solve a known problem in a more effective or efficient manner (Hevner et al. 2004). The contributions of this thesis are a formal framework and the specification and implementation of a novel interaction protocol for multi-tier resource allocation for composite service provision over multiple SN tiers. In contrast to current work, it avoids

overcommitments by service providers and does not require central entities for coordination.

Guideline 5: research rigor Research rigor refers to the application of rigorous methods for the construction and evaluation of design artifacts and addresses the way design-science research is conducted. Research rigor requires the effective use of the knowledge base, i.e., theoretical foundations and research methodologies (Hevner et al. 2004, p. 87–88). This work's theoretical foundations include an economic and a technical part. The economic part comprises game theory and service auctions in SNs. The technical part covers foundations of multiagent systems, service-oriented information systems, and SLAs. To solve the problem of multi-tier resource allocation, an interaction protocol engineering perspective (Huget & Koning 2003) is applied. This approach is closely related to the well-established area of communication protocol engineering (Holzmann 1991). For the verification of required protocol properties, formal methods from game theory (i.e., proofs) and communication protocol engineering (model checking) are applied. The simulation method used is multi-agent simulation (Kluegl 2001), which constitutes a mature method for the evaluation of interaction protocol implementations.

Guideline 6: design as a search process The design process can be regarded as a search process for an effective solution to a problem. Problem solving can be defined as utilizing available means to reach desired ends while satisfying laws existing in the environment (Simon 1996). Means qualify the set of actions and resources available to construct a solution; ends describe goals and constraints on the solution; and laws denote uncontrollable forces in the environment. However, it may not be possible to formulate appropriately and pose mathematically all relevant means, ends, or laws of an information system design problem to apply standard operations research techniques to determine an optimal solution. Assuming it is possible to do so, the size and complexity of the solution space will often render the problem computationally infeasible. For these problems, the search processes consist of determination of satisfactory (i.e., satisficing (Simon 1996)) solutions, without explicitly specifying all possible solutions (Hevner et al. 2004, p. 88–90). Regarding the formal specification of the protocol, this thesis formally verifies that the protocol is sound and allows multi-tier resource allocation. The simulation-based evaluation of the protocol implementation shows that the proposed solution is satisfactory with regard to the requirements addressed.

Guideline 7: communication of research Results of design-science research must be presented to technology- and management-oriented audiences in order to extend the research community's knowledge base and to make the artifact's benefits available to practitioners (Hevner et al. 2004, p. 90). A preliminary version of the protocol specification and a preliminary evaluation have been described and demonstrated to management-oriented audiences within the scope of the EU ICT project BREIN (Laria 2009, Jones 2010). The underlying software architecture of the system used for the simulation-based evaluation of the protocol implementation has been presented in (Karaenke, Micsik & Kirn 2009). The approach for quality of service (QoS) parameter aggregation has been published in (Karaenke & Leukel 2010, Karaenke, Leukel & Sugumaran 2013). A multi-tier contract net protocol specification for logistics service chains has been proposed in (Karaenke & Kirn 2010*a*). The generalization of this preliminary protocol to SNs and a formal verification by model checking have been partly described in (Karaenke & Kirn 2010*b*).

1.4 Outline

This thesis is structured in five chapters. It starts with an overview of multiagent systems and service networks, continues with a requirements analysis and related work, before presenting the proposed protocol along with its verification and validation.

Chapter 1: Introduction This chapter outlines the motivation and problem addressed in this thesis, its research approach and contributions, and discusses the epistemological position of this research.

Chapter 2: State of the art This chapter gives an overview of the fundamentals and basic concepts of multiagent systems, game theory in multiagent systems, service networks, and service-oriented computing. In addition, it provides a game-theoretic problem analysis based on a formal model for SNs, presents a specification of the requirements, and discusses related work.

Chapter 3: Design This chapter presents the protocol specification mathematically based on the game-theoretic model, in UML sequence diagrams, and in a PROMELA model. Different bidding policies for service providers are formally analyzed. Further, the protocol implementation and the simulation system are presented.

Chapter 4: Evaluation This chapter evaluates the proposed protocol. First, the protocol specification is evaluated based on the formal, game-theoretic model. Second, safety and liveness properties of the PROMELA model are verified by means of the model checker SPIN. Third, the results of the multiagent simulation, which evaluates the protocol implementation, are presented and discussed.

Chapter 5: Conclusions This chapter summarizes the results, draws major conclusions, outlines implications and limitations of the proposed protocol, and gives an outlook on future research.

Chapter 2

State of the art

This chapter provides the definitions of key concepts and underlying assumptions from multiagent systems, game theory in multiagent systems, and service networks. Based on a formal model for SNs, it uses game theory to derive a set of requirements that must be fulfilled to solve the problem addressed. Then it reviews relevant allocation approaches from the extant literature and assesses if and in how far they meet these requirements.

2.1 Definitions and assumptions

2.1.1 Autonomous agents & multiagent systems

This section introduces fundamental concepts from the distributed artificial intelligence (DAI) area. DAI can be divided into two primary research areas: distributed problem solving (DPS) and multiagent systems (MAS). DPS research addresses the division of problem solving processes among a set of entities, which can cooperate in dividing and sharing knowledge about problems and in developing solutions. Thus, DPS problems require a global perspective, even for understanding and stating the problem. In contrast, MAS research focuses on coordination of a number of intelligent agents to resolve conflicts between actions or to take advantage of the actions of other agents (Bond & Gasser 1988).

In the remainder of this section, concepts for a single computational entity, an agent, are presented. This discussion focuses on concepts from the MAS research area and agency in artificial intelligence (AI) in general. The literature analysis includes a definition of the term 'agent' – with focus on agency in the MAS research area – as well as discussions of different properties and types of architecture for agents. Further, systems with multiple interacting agents,

which form multiagent systems, and implications of agent properties for these systems, are presented and analyzed. This includes agent communication and coordination concepts in MAS.

2.1.1.1 Autonomous agent

Although the definition of the term 'agent' has been subject of comprehensive scientific discussions in AI since the 1980s, there is no generally accepted definition and the debate is ongoing; not only for agency in AI in general, but also for agency in MAS research in particular. This results from the fact that different attributes of agency are of varying importance in distinct application domains. For example, it may be desirable that agents learn (i.e., adopt their behavior) in some domains, while this is undesired in others (Wooldridge 2009). Franklin & Graesser (1997) discuss various different notions of agency, present a set of essential characteristics, and propose a taxonomy of autonomous agents.

This work provides a definition of *intelligent, autonomous software agents* – hereafter referred to as agents – from the MAS research area, based on related literature; it also discusses and analyzes properties as well as characteristics of agents in the remainder of this section.

Though there is a number of different definitions of the term 'agent' (e.g., (Ferber 1999)), the understanding of agency in this research is based on the *weak notion of agency* by Wooldridge & Jennings (1995), which constitutes the most commonly accepted definition. Wooldridge & Jennings (1995) also discuss a *stronger notion of agency*, considering human-like concepts like mentalistic notions (e.g., knowledge and obligations) or emotions (Wooldridge & Jennings 1995, p. 117). However, the stronger notion of agency is beyond the scope of this work.

Definition 2.1.1 (**agent**). *An* ***agent*** $a_i \in \mathcal{A}$ *is an* ***encapsulated software system*** *with* ***social ability*** *that is* ***situated*** *in an* ***environment****, and that is capable of* ***autonomous action*** *which is* ***expected to maximize*** *the agent's* ***delegated objective function*** *(Wooldridge & Jennings 1995, Jennings 2000, Russell & Norvig 2003).*

This definition and the understanding of agency in this work is based on the assumptions that (i) agents are encapsulated software systems; (ii) agents are situated in an environment; (iii) agents are autonomous from their environment and have delegated objective functions; (iv) agents are rational; (v) agents are capable of deliberation, reactive, and proactive behavior; (vi) agents have social ability; and (vii) agents can potentially improve their performance through learning. The following sections discuss these assumptions in detail.

Although *agent mobility* (i.e., mobile agents) is an important research field, it is beyond the scope of this research. That is, agent mobility is explicitly excluded. Moreover, this research assumes *persistence of agents* and the ability to conclude *binding agreements* (Rosenschein 1985, p. 30) as discussed further in section 2.1.1.2.

2.1.1.1.1 Encapsulated software system Agents are *encapsulated software systems*, i.e.,

- agents are identifiable entities with well-defined boundaries (Jennings 2000, p. 280) $\Leftrightarrow (\forall a_i, a_j \in A : a_i \neq a_j \Leftrightarrow i \neq j)$;
- *autonomous agents* are a concept of *computer science*, namely the *agent-based computing software paradigm* (Wooldridge 1997, Jennings 2000).

Encapsulation of complex data types in software modules provides an abstraction to hide (i) details of operations (procedural abstraction) and (ii) value representations (data abstraction). In addition, encapsulation is a protection mechanism to isolate changes in one software module from the remainder of the software system (Gannon, Hamlet & Mills 1987, p. 820). Encapsulation also plays an important role in the object-oriented software paradigm. Here, a class is separated into interface and implementation. This facilitates a separation of concerns, where the implementation details are hidden and the interface provides an abstract description of the class behavior (Booch 1993, pp. 46–47).

For agents, encapsulation also denotes an abstraction role that hides details of the realization. Therefore, an agent is an identifiability software system (e.g., module or class) with well-defined boundaries (Jennings 2000, p. 280). In definition 2.1.1, the identifiability is provided by the index i.

The concept of agency discussed in this work is fundamentally a concept of *computer science*, i.e., it considers the *agent-based computing software paradigm* (Wooldridge 1997, Jennings & Wooldridge 2000). Therefore, philosophical, social, and economical discussions of agency, which are beyond the capabilities of the Von Neumann architecture, are out of the scope of this research.

However, this work *does* utilize concepts from economic theory for agents, though the underlying assumptions are analyzed with regard to the applicability to *software systems*. For example, the set of possible actions of an agent is limited by, among others, the computational model and more specifically by the realization of an agent in software. That is, it cannot be assumed that an agent is capable of any imaginable action as is might be assumed in philosophical or social sciences (e.g., in principle, an agent cannot be assumed to have the capabilities to 'walk away' or terminate itself).

Further, in agent-based computing, several *metaphors* from other research disciplines such as social sciences are used (e.g., the 'social ability' of agents (Wooldridge & Jennings 1995, p. 116)). Hereby, it is essential to not over-interpret the implications the migrated concepts have for software systems. For example, 'social ability' is a metaphor which denotes the ability to coordinate joint and conflicting goals and actions with other agents required for, e.g., cooperative problem solving or negotiation (Wooldridge 1997, Castelfranchi 1998). However, it cannot be assumed that a 'society' of these communicating software components has the same properties as a human society. The same is true for autonomy and other properties of software agents. The interpretation of these properties for software systems are thus analyzed in the remainder of this section.

2.1.1.1.2 Situated in environment Agents are *situated in an environment*, i.e.,

- agents can *perceive their environment* through sensors $\Leftrightarrow$ there is a perception function per_i such that the environment $\Omega_i \subseteq \Omega$ is part of the domain of per_i for all agents $a_i \in A \Leftrightarrow \exists per_i : D_{per_i} \rightarrow CD_{per_i} \forall a_i \in A : \Omega_i \subseteq D_{per_i}$;
- agents can *execute actions in their environment* through effectors $\Leftrightarrow$ there is an action function act_i such that the set of possible actions Σ_i is part of the co-domain of act_i for all agents $a_i \in A \Leftrightarrow \exists act_i : D_{act_i} \rightarrow CD_{act_i} \forall a_i \in A : \Sigma_i \subseteq CD_{act_i}$ (Jennings 2000, Russell & Norvig 2003).

The *embeddedness* of the agent in its *environment*, i.e., being *situated* in the environment, is defined by an agent's ability to *perceive* the state of its environment through *sensors* and act on that environment through *effectors* (Russell & Norvig 2003, p. 32).

The concept of an agent's embodiment in its environment – which is adopted in the understanding of the term 'agent' in this work – is shown in figure 2.1. Agents generate actions to influence their environment (possibly including other agents) and receive feedback and other percepts through their sensors.

Agent environments can be further specified by general properties suggested by Russell & Norvig (2003) as follows. They can be *fully* or *partially observable*. Full observability denotes that an agent can perceive the *complete* state of the environment at any point in time through its sensors. An environment is *effectively* fully observable if an agent can perceive all *relevant* information for the choice of action. Relevance depends on the agent's delegated objective function.

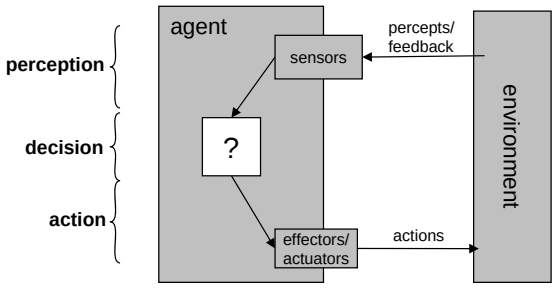

Figure 2.1: Agent and environment.
(after (Russell & Norvig 2003, p. 33) and (Wooldridge 2009, p. 22))

The environment can be *deterministic* or *stochastic*. In deterministic environments, agent actions have guaranteed effects, i.e., the resulting state of the environment is solely dependent on the current state and the agent's actions. In stochastic environments, there is uncertainty about the state that results from the agent's actions. If the environment is deterministic except for other agents' actions, it is denoted as *strategic*.

An agent's environment is qualified as *episodic* if the next episode does not depend on actions taken in the current episode. That is, the environment is divided into atomic episodes, which are independent of each other. In *sequential* environments, the current agents' decisions can affect all future decisions.

If the environment does not change while an agent is deliberating, it is denoted as *static* for this agent. If it can change during agent deliberation, it is denoted as *dynamic*. In dynamic environments, agents continuously have to decide what to do – in case they are still deliberating, they may also decide to do nothing.

The distinction of *discrete* and *continuous* environments can be applied to (i) the *state* of the environment, (ii) the way *time* is handled, and (iii) to the *perceptions* and *actions* of agents. An environment is state-discrete if it has a finite number of states, time-discrete if it has a finite set of points in time, and perception-/action-discrete if it has a finite set of perceptions/actions; i.e., the discreteness depends on the finiteness of the corresponding set of states, points in time, perceptions, and actions.

In addition, single-agent and multiagent environments can be distinguished – by the number of agents situated in the environment (Russell & Norvig 2003, pp. 40–44). However, this distinction is out of scope of this section, since it focuses on agents' relationships with their environments, independent of the number of agents situated in this environment. Multiagent environments are discussed in detail in section 2.1.1.2.

The agents' environments considered in this work are assumed to be partially observable, strategic (stochastic), sequential, dynamic, state-continuous, time-discrete, perception-/action-continuous, and multiagent.

Definition 2.1.2 (**agent environment**). *Let $a_i \in \mathcal{A}$ be an agent and Ω be the world state. a_i**'s environment** $\Omega_i \subseteq \Omega$ consists of anything an agent can **perceive** through its **sensors**, denoted by function per_i, and **act on** through its **effectors**, denoted by function act_i (Jennings 2000, Russell & Norvig 2003).*

2.1.1.1.3 Autonomy and delegation Agents are *autonomous from their environment* and have *delegated objective functions*, i.e., agents have

- delegated objective functions $OF_i \Leftrightarrow \exists OF_i \forall a_i \in A$;
- beliefs BEL_i, constituting the agents' interpretation of the perceptions received $\Leftrightarrow \exists BEL_i \forall a_i \in A$;
- states of parameters of their environment with positive influence on the objective functions, referred to as goals $GOAL_i \Leftrightarrow \exists GOAL_i \forall a_i \in A$;
- perception functions per_i that interpret the perceptions received in accordance to the delegated objective function $\Leftrightarrow \exists per_i : \Omega_i \times OF_i \rightarrow BEL_i \forall a_i \in A$;
- goal functions $goal_i$ that infer the goals in accordance to the beliefs, delegated objective function, and existing goals $\Leftrightarrow \exists goal_i : BEL_i \times OF_i \times GOAL_i \rightarrow GOAL_i \forall a_i \in A$;
- action functions act_i that infer the actions in accordance to the goals, delegated objective function, and planned actions $\Leftrightarrow \exists act_i : GOAL_i \times OF_i \times \Sigma_i \rightarrow \Sigma_i \forall a_i \in A$; and
- control threads CT_i which manage the internal execution of inference processes $\Leftrightarrow \exists CT_i \forall a_i \in A$.

The mentioned sets and functions *cannot be directly influenced by the environment* (including other agents) – this constitutes the autonomy (Wooldridge & Jennings 1995, Castelfranchi 1995, Luck & d'Inverno 1995, Jennings 2000, Russell & Norvig 2003, Wooldridge 2009).

Wooldridge (2009) identifies *autonomy* as a commonly accepted property of agents in MAS; it is a key characteristic that distinguishes agents from objects (Wooldridge 1997). Wooldridge & Jennings (1995) characterize agents' autonomy as follows: agents have control over their internal state and their behavior

(i.e., actions) without direct external interaction. However, the authors do not provide a comprehensive discussion of the autonomy concept for agents, but refer to Castelfranchi (1995).

Castelfranchi (1995) analyzes the fundamentals of agent autonomy and proposes different autonomy dimensions. He characterizes autonomy as an intrinsically relational concept, i.e., the autonomy of an entity is defined in relation to another entity. For autonomous agents, two main relations are identified: autonomy from the environment (physical context) and autonomy from other agents (social context). However, autonomy does not imply 'autism', i.e., although the environment (including other agents) cannot *directly* influence the behavior and internal state of agents, it does not mean that agents are totally unrelated and unconcerned about the environment. The behavior can be *indirectly* influenced by external stimuli, though it cannot be determined or imposed by them, i.e., agent behavior is not determined by a set of rigid and deterministic reflexes to external stimuli. The autonomy of an agent from stimuli it percepts through its sensors is referred to as *cognitive autonomy*.

Further, Castelfranchi (1995) distinguishes *executive autonomy* and *goal autonomy*. Executive autonomy is relative to the actions ('means') an agent performs, while goal autonomy is related to the goals ('ends') an agent tries to achieve. That is, executive autonomy refers to the autonomy *how* an agent tries to reach its goals, and goal autonomy refers to the autonomy *what* it tries to achieve.

In addition to not being directly influenceable by the environment, agent goals can only be influenced through agent beliefs (i.e., double indirect influence). Therefore, an agent's control over its beliefs constitutes an additional filter for the influences of the environment on the agent's goals (Castelfranchi 1995).

Luck & d'Inverno (1995) formally define autonomy of an agent by the existence of a 'motivation' of an agent. By motivation, the authors refer to non-derivable top-level goals, guarded by internal inaccessible rules. That is, autonomous agents evaluate their possible behavior not only regarding their environment, but also regarding non-derived, static top-level goals (Luck & d'Inverno 1995). Jennings (2000) and Wooldridge (2009) denote this 'motivation' by 'design objectives' (Jennings 2000, p. 280) and 'delegated objectives' (Wooldridge 2009, p. 21) respectively. Russell & Norvig (2003) characterize the 'motivation' of an agent in AI as an intended maximization of its performance measure (Russell & Norvig 2003, pp. 35–36). In this work, the 'motivation' of an agent is denoted as an agent's *delegated objective function*, i.e., it is a

delegated objective which is measurable and constitutes the criterion of success of the agent's actions in accordance to the delegated objectives.

Jennings (2000) states that autonomy implies that agents have a dedicated, persistent thread of control (Jennings 2000, p. 283).

Based on the literature analysis, the perception and action functions from definition 2.1.2 can be substantiated. Let $a_i \in \mathcal{A}$ be an agent, OF_i a_i's delegated objective function, BEL_i a_i's beliefs, $GOAL_i$ a_i's (operational) goals, Σ_i a_i's set of possible actions and Ω_i a_i's environment. Then, function $per_i : \Omega_i \times OF_i \rightarrow BEL_i$ interprets the perceptions received in accordance to the delegated objective function (i.e., relevance of perceptions). This interpretation and filtering function for perceptions realizes the *cognitive autonomy*, as the agent is autonomous in interpretation of the perceptions in accordance to its delegated objective function.

$goal_i : BEL_i \times OF_i \times GOAL_i \rightarrow GOAL_i$ infers agent a_i's operational (sub-) goals in accordance to the beliefs, delegated objective function, and existing goals. Thus, this function implements the *goal autonomy* of agent a_i.

$act_i : GOAL_i \times OF_i \times \Sigma_i \rightarrow \Sigma_i$ infers agent a_i's actions in accordance to the goals, delegated objective function, and planned actions. Therefore, this function realizes the *executive autonomy* of agent a_i. However, the set of actions Σ_i is necessarily limited due to (i) the computational model of a_i and (ii) the delegated objective function; i.e., it cannot be assumed that an agent is capable of any imaginable action.

Finally, to be autonomous in the elaborated sense, an agent a_i needs control of its internal execution thread, i.e., a control thread CT_i which manages the internal execution of inference processes (e.g., execution order). In this work, the control of this thread in combination with conceptual control of the inference processes is denoted as *control autonomy*. Figure 2.2 shows an agent, its environment, and the different forms of autonomy elaborated in this section.

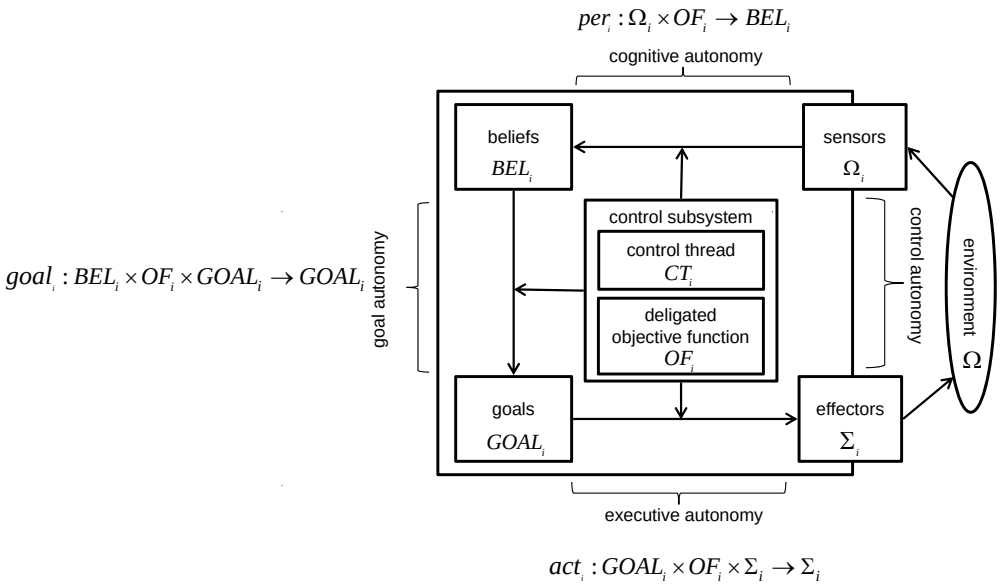

Figure 2.2: Agent, environment, and different forms of autonomy.

2.1.1.1.4 Rationality Agents are *rational*, i.e.,

- agents choose the actions that are expected to maximize the value of their delegated objective function ($\max_{(\sigma_{i,1},\ldots,\sigma_{i,n})\in\Sigma_i}(E(OF_i))$), though the rationality is limited by computational resources, and therewith referred to as *bounded rationality* or *bounded optimality* (Russell & Subramanian 1995, Russell & Norvig 2003).

Towards a definition of intelligence of agents, Russell & Norvig (2003) discuss the concept of *rational agents* – agents that do the right things. Moreover, they identify rationality as a central concept for AI. With regard to agents, rationality depends on performance measures, an agent's knowledge about the environment, an agent's possible actions, and an agent's perceptions received. Thereby, performance measures define the criterion of success of an agent's actions, regarding the environment and perceptions received. Thus, rational agents are maximizing their expected performance measures for any possible combination of perceptions received (Russell & Norvig 2003, pp. 35–36).

The agency concept in this work adopts the rationality for autonomous agents. An autonomous agent a_i is optimizing the expected value of its delegated objective function OF_i by taking this function into account for inference of beliefs from perceptions, goals from beliefs, and actions from goals.

However, *perfect rationality* – acting in a way to maximize the expected utility in every instant – is a concept from theory which is in general unrealistic in practice; given that computational resources of agents are limited, the

calculations for choosing appropriate actions is too time-consuming in most environments. The notion of rationality for an agent that *eventually* returns the perfectly rational choice, potentially too late to be of any value, is referred to as *calculative rationality*; i.e., a calculative rational agent will eventually return what *would have been* the optimal choice. The notion of calculative rationality is mostly applied when designing logical or decision-theoretic agents. Here the calculations of the optimal choice are not considered, but the optimality of actions is focused on.

Similarly, perfect rationality is neglected for humans by Simon (1957) in his *principle of bounded rationality*, as the capacity of the human mind is small as compared to the size of real-world problems requiring objectively rational behavior for their solution. This also applies for reasonable approximations of this objective rationality (Simon 1957, p. 198). Simon characterizes *bounded rationality* by deliberation just until an 'acceptable' (i.e., satisficing) solution is found (Simon 1982). However, the acceptance criterion is not formally defined, and satisficing is one of several alternatives to deal with bounded resources (Russell & Norvig 2003, pp. 972–973).

In AI, the constrained rationality of agents, limited by computational resources, is denoted as *bounded optimality*. A bounded optimal agent acts as satisfactorily as possible with the computational resources available. Bounded optimality specifies optimal *programs*, not optimal actions, since agent actions are generated by programs (Russell & Subramanian 1995).

2.1.1.1.5 Deliberation, reactivity, and proactiveness Agents are capable of *deliberation, reactive, and proactive behavior*, i.e.,

- *deliberation* denotes the ability of agents to determine which state of the world Ω_i is desirable to be achieved, i.e., infer the *goals* $GOAL_i$ for agent a_i;
- *reactivity* denotes the ability of agents to perceive the environment and adopt their beliefs, goals, and actions to changes in it;
- *proactiveness* denotes the ability of agents to determine actions and subgoals to achieve current agent goals without external triggers (Wooldridge & Jennings 1995, Wooldridge 1997).

The decision processes of agents which actions to perform in order to meet its delegated objectives (i.e., maximize the expected value of their delegated

objective functions) depend on the agents' architectures, that is, software architectures for decision-making systems embedded in environments (Wooldridge 2009). Thus, abstract agent architectures are briefly discussed in the following.

On an abstract architecture level, purely reactive agents and agents with state can be distinguished. Purely reactive agents, also referred to as simple reflex agents (Russell & Norvig 2003, pp. 46–48), decide what to do without a consideration of the complete perception sequence, i.e., they react to the latest perception only.

Agents with state have internal data structures in which they can track the percepts received. This information can be utilized for decision making (Wooldridge 2009, pp. 36–38). Russell & Norvig (2003) denote reactive agents with state as model-based reflex agents. These agents use an internal state to track changes in the environment which are not evident in the current percept (Russell & Norvig 2003, pp. 44–56). However, since *purely* reactive agents do not meet this work's definition, the agents considered in this research are explicitly *not* purely reactive.

In contrast to reactive agents, deliberative (i.e., deductive reasoning) agents are built on the *symbolic AI* paradigm – the classical approach of building AI systems. Thus, these agents have an explicitly represented, symbolic model of the world (i.e., of the environment and the desired behavior). Decisions are made, based on this model, via reasoning mechanisms. If the symbolic models constitute logical formulae, the reasoning corresponds to logical deduction or theorem proving (Wooldridge 2009, pp. 45–50).

Practical reasoning must be distinguished from logical reasoning and is directed towards determining which actions to perform; it consists of two distinct activities. First, it has to be determined which state of the world Ω_i is desirable to be achieved, i.e., infer the *goals* $GOAL_i$ for agent a_i. This process is referred to as deliberation. Second, it has to be determined how this state can be achieved by performing appropriate actions, i.e., infer the *actions* $\sigma_i \in \Sigma_i$ for agent a_i. This is referred to as means-end reasoning (Wooldridge 2009, pp. 65 ff.).

Reactive and deliberative agent architectures are not mutually exclusive. Hybrid architectures exist which combine reactive and deliberative behavior in a single architecture, integrating subsystems to deal with reactive and proactive behavior (Wooldridge & Jennings 1995, pp. 134–138). There is a class of architectures where these subsystems are organized in layers; at least one for reactive and proactive behavior each. The interacting layers are characterized by an either horizontal or vertical control and information flow. In a horizon-

tal layering, each layer is able to receive perceptions and produce effects (i.e., suggestions of which actions to perform) independent of other layers. In contrast, a vertical layering results in the processing of inputs and the production of outputs by at most one layer (Wooldridge 2009, pp. 92 ff.).

This work does not assume a specific organization of the agents' architectures subsystems in layers, though the agents considered in this thesis have a *hybrid* architecture; i.e., they are capable of both reactive and proactive behavior – discussed in detail in the following.

Wooldridge & Jennings (1995, p. 116) characterize *reactivity* of agents as having the ability to perceive the environment and respond in a timely fashion to changes in it. They define *proactiveness* of agents as being able to act without external triggers on their own initiative (i.e., goal-driven).

Wooldridge (1997) describes proactiveness as being able to *plan how to achieve goals* without external triggers, and, if necessary, to generate subsidiary goals. In relation to the plans of achieving goals and active goals themselves, reactivity constitutes the ability to not blindly executing plans or pursue goals. In contrast, in the event of circumstances in the environment which conflict with preconditions or invariants of active plans or goals, an agent should react to the changed situation accordingly in time for the response to be useful (Wooldridge 1997, pp. 27–28).

The capability to determine actions and sub-goals to achieve current agent goals is realized in function $goal_i \forall a_i \in A$. This goal inference is internally triggered by the control thread of an agent. It is done independently of actual changes in the environment – internal events (e.g., timers) can also initiate the goal reasoning process. The reactivity as described by Wooldridge (1997) is implemented by the feedback loop between actions, perceptions, beliefs, goals, and updated actions. The timeliness of responses is out of scope of this work's model, since (i) it is domain dependent and (ii) timely responses to any change in the environment for any imaginable application domain, objective function, and set of goal is an assumption that is obviously both impossible to guarantee and to be proven. This is also acknowledged in Wooldridge's example for reacting in sufficiently short time, and in general by stating that this is rather a vision of agents than a hard requirement (Wooldridge 1997, p. 28). However, rationality implies that an agent will only perform actions if they are expected to increase the value of the delegated objective function. If actions are expected to be performed too late to have positive effects on OF_i, agent a_i will not consider these actions (act_i will not return them); if the actions have already been planned, the agent will not perform these actions, but react to the changed

expected impact and dismiss the actions.

2.1.1.1.6 Social ability Agents have *social ability*, i.e.,

- agents have the capability of interacting with other agents via agent communication languages (Genesereth & Ketchpel 1994, Wooldridge & Jennings 1995);
- agent interactions are conceptualized on the knowledge level (Newell 1982), i.e., they are conceived in semantic terms;
- agents have the ability to make run-time decisions about interactions (Jennings 2000); and
- agents are able to coordinate joint and conflicting goals and actions with other agents required for, e.g., cooperative problem solving or negotiation (Wooldridge 1997, Castelfranchi 1998).

According to Wooldridge & Jennings (1995), agents have 'social ability': the capability of interacting with other agents via agent communication languages (Genesereth & Ketchpel 1994). Although the term 'social' is common in agent-based computing to describe the ability of agents to communicate with others (e.g., (Wooldridge & Jennings 1995, Wooldridge 1997, Jennings 2000)), it can lead to misinterpretations of how 'social' agents are in comparison to human societies. In addition, the term is defined differently in related literature. For example, in the definition by Wooldridge & Jennings (1995), the capability to interact with other agents (i.e., to communicate) is sufficient for the social ability property. In contrast, Castelfranchi (1998) states that agents cannot be considered 'social' just because they communicate. Agents are social because they interfere with, depend on, and influence each other when they act in a common environment (Castelfranchi 1998, p. 159). This includes coordination of joint and conflicting goals and actions, i.e., cooperative pursuing of goals and resolution of conflicts. Similarly, Wooldridge (1997) states further that agents should have the ability to engage in 'social activities' with other agents in order to achieve their goals. That is, to perform interactions with other agents required for, e.g., cooperative problem solving or negotiation.

As Jennings (2000) points out, agent interactions are conceptualized on the knowledge level (Newell 1982); they are conceived in semantic terms (e.g., which goals, when, by whom) in contrast to method invocations and function calls which take place on a purely syntactic level. Further, since agents operate in environments that they can only partly control and observe, agents need the

ability to make run-time decisions about interactions in contrast to hard-wired interactions in, for example, object-oriented programming.

Therefore, in this work, this *capability of communicating with other agents* is denoted as the *ability to interact* with other agents in terms of asynchronous communication. Contrarily, the understanding of social ability of agents goes beyond communication. An agent is assumed to have social ability if and only if (i) it is capable to interact (i.e., communicate) with other agents via agent communication languages, (ii) interactions are conceptualized on the knowledge level, (iii) decisions about interactions are made at run time, and (iv) joint and conflicting goals and actions can be coordinated with other agents (Newell 1982, Wooldridge & Jennings 1995, Wooldridge 1997, Castelfranchi 1998, Jennings 2000).

2.1.1.1.7 Agent learning Agents can potentially improve their performance through *learning*, i.e.,

- learning agents can improve their expected performance for future actions in accordance to the delegated objective function OF_i by using percepts received as experience in their reasoning cycle from perceptions to actions (Mitchell 1997, Russell & Norvig 2003).

Learning of software systems, also referred to as *machine learning*, denotes improving the system's performance, indicated by a performance measure, with experience (Mitchell 1997). For software agents, learning denotes the capability of improving the agents' abilities to act in the future by experience; by using percepts received, learning agents can improve the *expected* performance in accordance to the delegated objective function OF_i. Thus, the reasoning cycle from perceptions over beliefs and goals to actions is adapted to maximize the expected performance. However, different forms of learning from the simple memorization of experience to the creation of scientific theories exist (Russell & Norvig 2003).

In this thesis, learning beyond memorization of perceptions by means of their interpretation to beliefs is not considered in detail. Nevertheless, strategic behavior of agents to gain individual performance is analyzed, based on the game-theoretic framework.

2.1.1.2 Multiagent system

In section 2.1.1.1.2, properties of agent environments have been discussed from a single-agent perspective. Properties only relevant for environments with multiple agents – multiagent environments – are discussed in this section. Such

an environment is the basis for a multiagent system (MAS). The autonomous agents situated in this environment constitute a multiagent system. Thus, a MAS is a collection of autonomous agents, interacting through a communication infrastructure, representing different interests of stakeholders (Wooldridge 2009, p. 5).

According to Huhns & Stephens (1999), these environments may be *open* or *closed* and may or may not contain more than one agent. However, the authors identify these environments to be typically open and to contain a number of autonomous distributed agents. Similarly, it is assumed in this research that multiagent environments are *open* and potentially contain *multiple agents*. Further, the multiagent environment provides communication means to enable interactions (Huhns & Stephens 1999, p. 81–82).

Openness of the multiagent environment is defined as having *multiple loci of control* (Jennings 2000, p. 280). In open MAS, agents can in principle join and leave at any time and are owned by various stakeholders with different objectives (Huynh, Jennings & Shadbolt 2006, p. 120). That is, openness is characterized by the absence of a single controlling organization and no central designer (Huhns & Stephens 1999, p. 82), by software representing interests of a diverse range of stakeholders, and by constant change (Gasser 1991, Hewitt 1991). Openness requires autonomy and delegated objectives for agents in MAS: To represent the interests of their owner, agents have to receive respective objectives. To follow those different interests, agents must be autonomous from other agents. Openness does not necessarily imply that agents cannot conclude binding agreements on behalf of their owners. For example, in open real-world electronic marketplaces, participants can freely join and leave the platform. Concluded agreements are nevertheless valid and binding beyond platform participation. This research assumes that agents can conclude binding agreements as in cooperative game theory (Rasmusen 1989, Binmore 1992). Agent mobility, however, is out of scope of this research.

Definition 2.1.3 (**multiagent system**). *A **multiagent system** (MAS) is an **open** system of **multiple agents** that are **situated** in a **shared environment** that **enables agent communication** (Huhns & Stephens 1999, Jennings 2000, Wooldridge 2009).*

The agents in a MAS may have compatible goals (i.e., there's a single global goal) or individual conflicting objectives. The agents are required to interact with one another to manage the dependencies that result from the shared environment. Thus, *the* key problem in MAS is the *coordination* of agents in a shared environment (Bond & Gasser 1988, Jennings 1993). Multiagent coordi-

nation is discussed in section 2.1.1.2.2.

The understanding of multiagent systems in this work is based on the following assumptions.

- MAS are assumed to be *open*, i.e., there are *multiple loci of control* (Jennings 2000, p. 280). Openness is characterized by concurrent, asynchronous software systems with decentralized control, representing interests of a diverse range of stakeholders (Bond & Gasser 1988, Gasser 1991, Hewitt 1991).
- The agents in a MAS can conclude *binding agreements* (Rosenschein 1985, p. 30), i.e., as in cooperative game theory (Rasmusen 1989, Binmore 1992). However, due to the agents' autonomy, they are able to break these agreements, though agents not fulfilling binding agreements may have to pay a penalty fee.
- MAS and the agents in MAS are assumed to be *persistent*, i.e., start and termination of MAS and agents are not considered. That is, the set of agents of the MAS remains constant over the time relevant for the investigations of this research. Agents joining and leaving the MAS at run time are not considered.
- The *mobility of agents* is explicitly not considered, i.e., agents moving from one execution environment (e.g., machine) to another.

2.1.1.2.1 Multiagent communication Intelligent agents often operate in MAS to be able to work together productively in interconnected computer networks. Agent communication enables the agents to coordinate their actions in order to reach goals of their own or their multiagent organization more effectively or efficiently. The multiagent environment has to provide the computational infrastructure to enable multiagent interactions. Further, protocols are required for communication and interactions of agents.

Communication protocols enable agents to communicate and to interpret the messages exchanged. Therefore, communication protocols define the methods of interconnections as well as the syntax and semantics of the messages transferred. Whereas the syntax refers to the actual message content, the semantics also depend on the message type. In general, messages can either be assertions or queries. However, most agent communication languages consider more specific message types, based on the speech act theory (Singh 1993). To understand the semantics of a message, i.e., what the symbols denote, a shared ontology is required.

Interaction protocols define a structured exchange of defined messages between agents, i.e., interaction protocols facilitate agent conversations (Huhns & Stephens 1999, Wooldridge 2009).

Definition 2.1.4 (**communication channel**). *A **communication channel** CC_{ij} is a directed link between two agents – $CC_{ij} = (a_i, a_j) \in CC$ – which allows messages to be send from agent a_i to a_j (Wooldridge, Jennings & Kinny 2000).*

Communication channels do not define when or what messages are sent between agents, but indicate that a message path exists, i.e., messages *can* be send. The set of all communication channels of an agent defines its acquaintances. That is, all other agents of a MAS an agent is able to send messages to (Wooldridge et al. 2000).

Definition 2.1.5 (**agent acquaintances**). *An **agent's acquaintances** ACQ_i consist of the set of agents to which the agent has **communication channels** to, i.e., $ACQ_i = \{a_j : (a_i, a_j) \in CC\}$ (Wooldridge et al. 2000).*

2.1.1.2.2 Multiagent coordination There is a large body of research concerning coordination, as the scope of coordination research is both fuzzy and wide. Consequently, an interdisciplinary study of coordination research leads to a generic definition of coordination: "Coordination is managing dependencies between activities" (Malone & Crowston 1994, p. 90).

In DAI, coordination is regarded as a system property which facilitates actions of multiple agents in a shared environment, e.g., by providing means for livelock and deadlock avoidance. Nevertheless, there is no generally accepted definition or taxonomy of coordination in DAI. Huhns & Stephens (1999, p. 83) state that among self-interested agents, coordination is performed by negotiation, while non-antagonistic agents are coordinated by cooperation.

However, the term 'cooperation' is also often used in a broader sense, referring to distributed systems which have to interact to carry out their assigned tasks, i.e., cooperation in its broader sense includes interactions of both benevolent and competitive agents (Wooldridge 2009, p. 151). In this research, cooperation is regarded as coordination of non-antagonistic (benevolent) agents, though coordination of antagonists does neither imply cooperation nor negotiation, as it can also be performed by, e.g., legal proceedings (Bond & Gasser 1988, p. 19).

Definition 2.1.6 (**multiagent coordination**). ***Multiagent coordination** is **managing dependencies** between **agent actions** in a **multiagent system** (Malone & Crowston 1994, Huhns & Stephens 1999).*

2.1.2 Multiagent resource allocation

Resource allocation is a well-established field in multiagent research (Chevaleyre et al. 2006). This section introduces basic concepts of multiagent research allocation (MARA), i.e., game-theoretic concepts utilized for analyzing and designing resource allocation mechanisms in MAS and the relationship of intelligent agents as entities (i.e., players) acting in these systems along with coordination concepts.

Decision theory (Raiffa 1968) and game theory (Rasmusen 1989, Binmore 1992) are closely related, and both have received adoption by multiagent researchers for studying allocation problems. Decision theory addresses the maximization of the expected utility of decision makers under uncertainty. Game theory concentrates on multiagent encounter, i.e., interactions of agents, utility maximizing strategies in multiagent interactions, as well as the design of protocols or mechanisms to affect the behavior of rational, interacting agents to lead to desired outcomes of agent interactions (Parsons & Wooldridge 2002). Decision theory is also regarded as the theory of games against nature. Both disciplines have strong relations to the work of von Neumann & Morgenstern (1944) and share elementary concepts, e.g., individual preferences described by utility.

Game theory is applied to multiagent systems in which multiple agents have to reach agreements by negotiation and bargaining (Rosenschein & Zlotkin 1994, Kraus 1997). In these systems, game theory provides means to analyze and develop protocols (i.e., the 'rules of encounter' (Rosenschein & Zlotkin 1994)) for the negotiation interactions. A protocol can be regarded as a function which defines valid actions by the participants based on the negotiation history. In this context, mechanism design – also referred to as 'reverse game theory' – denotes the design of protocols so that any negotiation outcome has desirable properties (Parsons & Wooldridge 2002).

Definition 2.1.7 (**multiagent resource allocation**). *Let $\mathcal{A}$ be a set of agents and $\mathcal{R}$ be a set of resources, where $\mathcal{R}_j \subseteq \mathcal{R}$ denotes the set of resources of agent $a_j \in \mathcal{R}$. The distribution of $r_j \in \mathcal{R}_j$ to $a_i \in \mathcal{A} \forall a_i, a_j \in \mathcal{A}$, i.e., the allocation of r_j to $a_i \Leftrightarrow x(r_{ij}) = 1$, that is influenced by the agents, is denoted as multiagent resource allocation (MARA) (Chevaleyre et al. 2006).*

To apply game theory to the analysis and design of MARA, the assumptions of the applied concepts and methods from MAS and game theory have to be compatible. In addition, particular assumptions are made, regarding the application of MARA in software service networks (e.g., regarding the type of

resources). Therefore, this work makes the following assumptions for MARA. Agents in MAS are (i) *rational* and any rational agent possesses a (explicit or implicit) *utility function* whose expected value the agent is trying to maximize (Russell & Norvig 2003, p. 51); (ii) agents in MAS can conclude *binding agreements* (Rosenschein 1985, p. 30), i.e., as in cooperative game theory (Rasmusen 1989, Binmore 1992); (iii) agents' resources required for the production of services are regarded as being discrete, indivisible, non-shareable, and static throughout the allocation process in single-unit and multi-unit settings, depending on the substitutionality of resources for service production (Chevaleyre et al. 2006, pp. 10–12); and (iv) the *utilitarian social welfare* is utilized as a metric for allocative efficiency, since it is well suited to assess the system performance in terms of the maximal average profit of negotiating agents in electronic commerce transactions (Chevaleyre et al. 2006, p. 16). The following sections discuss these assumptions in detail.

2.1.2.1 Agents, games, and strategies

In applications of game theory in multiagent research, the *agents* are the *players* in games. An agent's (i.e., a player's) *type* abstracts from goals, beliefs, objective function, and control thread and determines the preference structure of the agent. An agent's *strategy* denotes the plan of actions of this agent for the game (e.g., an agent's bid in an auction). A *strategy profile* describes the set of all agents' strategies. The agents are assumed to be *rational*, i.e., agents chose the strategy that maximizes their (expected) utility, using their available information.

Definition 2.1.8 (**agent type**). *Let $a_i \in \mathcal{A}$ be an agent, θ_i denotes a_i's type which determines a_i's preference structure.*

Definition 2.1.9 (**strategy**). *Let $a_i \in \mathcal{A}$ be an agent, $\sigma_i(\theta_i) \in \Sigma_i$ denotes a_i's strategy, i.e., a_i's plan of actions for the game, where $\Sigma_i \subseteq \Sigma$ denotes the set of all available strategies for agent a_i.*

Definition 2.1.10 (**strategy profile**). *Let $\mathcal{A} = \{a_1, \cdots, a_n\}$ be the set of agents. The vector of all agents' strategies $\sigma(\theta) = (\sigma_1(\theta_1), \ldots, \sigma_n(\theta_n))$ is denoted as the strategy profile of the agents.*

A game is denoted as being strategic if the strategy of at least one agent has an impact on the strategy of at least one other agent. In contrast, strategic independence is present if the agents' decisions are independent of one another. Game theory is applied to analyze strategic games (Rasmusen 1989, Binmore 1992).

2.1.2.2 Types of resource

The nature of the resources themselves is a central property of resource allocation problems. While some resource properties directly refer to the resources, others constitute characteristics of the allocation process. Resources can be either continuous (e.g., liquids) or discrete (e.g., bottles). While continuous resources are typically infinitely divisible as a matter of principle, the allocation process may restrict the allocation to certain quantities, i.e., the continuous resource is discretized. Thus, the allocation mechanism greatly influences whether a resource is divisible or indivisible.

Besides resource characteristics, the allocation mechanism may also influence if a resource is shareable or non-shareable; shareable resources can be allocated to multiple agents at the same time. Further, resources can be consumable (e.g., fuel) and perishable (e.g., food). Non-consumable, non-perishable resources, which do not change during the allocation process, are denoted as static resources. Finally, single-unit or multi-unit resources can be distinguished. In a multi-unit setting, a set of resources exists for which a common identifier exists, i.e., single items of this set cannot be distinguished (e.g., multiple bottles with the same content). In a single-unit setting, each of the resources to be allocated can be identified using a unique identifier. Thus, any multi-unit allocation problem can be transformed into a single-unit allocation problem (Chevaleyre et al. 2006).

In this research, resources required for the production of services (i.e., hardware and software) are regarded as being discrete and indivisible, since it is assumed that computational resources are not practically infinitely divisible, but are discretized for their allocation in SNs. Resources are non-shareable in this setting, since the services produced using the allocated resources require these resources as a whole. That is, while a CPU or memory can be shared by multiple services in principle, the computing units allocated already constitute arbitrary small portions of the physical resources and are thus assumed not to be used simultaneously for the production of multiple services. The resources in this research are assumed to be static, since their characteristics do not change during the allocation process. Finally, this work considers multi-unit resources if the resources are subsitutable between service providers, since the single discretized computational units are non-distinguishable. In contrast, if the resources are non-substitutable between service providers, the allocation procedure is to be considered single-unit.

2.1.2.3 Utility and preference representation

Agents in MAS are *rational* and any rational agent possesses a (explicit or implicit) *utility function* whose expected value the agent is trying to maximize (Russell & Norvig 2003, p. 51).

The individual satisfaction of different alternatives is expressed by agents' preferences. In the context of resource allocation, these preferences refer to the satisfaction of alternative allocations, i.e., the sets of possible allocations of resources to an agent. An agent's preferences over a set of alternatives is denoted as the agent's preference structure.

Preference structures can be represented by means of different mathematical models; four types of preference structures can be distinguished: (i) cardinal, (ii) ordinal, (iii) binary, and (iv) fuzzy preference structures. Cardinal preference structures can be quantitative (i.e., numerical values are assigned to each alternative) or qualitative (i.e., elements from an ordered scale are assigned to each alternative). The assignment of values or elements to an alternative is done by means of an agent's *utility function* (Wooldridge 2009, pp. 223–226).

According to Russell & Norvig (2003), any rational agent possesses a (explicit or implicit) utility function whose expected value the agent is trying to maximize (Russell & Norvig 2003, p. 51), i.e., utility functions are a prerequisite for rationality of agents.

Since quantitative cardinal preference structure is most relevant for multiagent resource allocation, this work focuses on this structure. Quantitative preferences allow for both an inter-agent comparison of utility as well as for an expression of preference intensity. Thus, differences between two utility degrees can be calculated, e.g., for monetary compensations. Formally, utility can be defined as follows. Let

- $\mathcal{A} = \{a_1, \cdots, a_n\}$ be the set of agents,
- $\Omega = \{\omega_1, \cdots, \omega_m\}$ be the set of world states, and
- $\mathcal{U} = \{u_1, \cdots, u_n\}$ be the set of utility functions of the agents.

The world state denote any 'outcome' that the agents have preferences over. Then, agent $a_i \in \mathcal{A}$'s utility function can be defined as

$$u_i : \Omega \rightarrow CD_{u_i} \, \forall u_i \in \mathcal{U}.$$

The co-domain CD_{u_i} is typically a set of *quantitative* (numerical) values (e.g., $[0, 1]$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, etc.), though it can also be a set of totally ordered

qualitative values (e.g., literals such as "good" and "very good") (Chevaleyre et al. 2006, Wooldridge 2009).

Let ω, $\omega' \in \Omega$. A utility function defines a *preference ordering* $\succeq_i$ with

$$\omega \succeq_i \omega' \Leftrightarrow u_i(\omega) \geq u_i(\omega').$$

That is, agent a_i prefers state ω at least as much as state ω'. For *strict preference* this yields

$$\omega \succ_i \omega' \Leftrightarrow u_i(\omega) > u_i(\omega'),$$

i.e., agent a_i strictly prefers state ω over ω' (Wooldridge 2009).

Definition 2.1.11 (**utility function**). *Let $\mathcal{A}$ be the set of agents, Ω be the set of world states, and $\mathcal{U}$ be the set of utility functions of the agents. $u_i : \Omega \rightarrow CD_{u_i}\ \forall u_i \in \mathcal{U}$ denotes the utility function of the agents that defines the preference ordering of world states for every agent.*

In MARA, the set of world states Ω typically denotes the allocation of resources to agents. Let

$$\mathcal{R} = \{r_1, \cdots, r_m\}$$

be the set of resources. A resource allocation setting is represented by the triple $\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$. An agent's utility function associates a value to every resource subset (or bundle of resources), i.e.,

$$u_i : \mathscr{P}(\mathcal{R}) \rightarrow CD_{u_i}\ \forall u_i \in \mathcal{U},$$

where $\mathscr{P}(\mathcal{R})$ denotes the powerset of $\mathcal{R}$ and thus $\Omega = \mathscr{P}(\mathcal{R})$. Then, an allocation of resources is a mapping from $\mathcal{A}$ to $\mathscr{P}(\mathcal{R})$. This simple definition over resource bundles assumes that the values that agents assign to resources are *independent* of the allocation of other resources to agents. This is also denoted as being free of allocation externalities (Chevaleyre et al. 2006, p. 13). In this research, however, this assumption is too strict for the allocation of resources for service production, respectively services provided to agents in SNs. Thus, these assumptions will be relaxed in the remainder of the work.

2.1.2.3.1 Dominant strategies A *dominant strategy* is the best response of an agent to all available strategies of the other agents.

Definition 2.1.12 (**dominant strategy**). *Let $a_i \in \mathcal{A}$ be an agent and $\sigma_i(\theta_i)$*

a_i's strategy. $\sigma_i(\theta_i)$ is a dominant strategy if and only if

$$u_i(\sigma_1(\theta_1), \ldots, \sigma_i(\theta_i), \ldots, \sigma_n(\theta_n)) \geq u_i(\sigma_1(\theta_1), \ldots, \sigma_i(\theta_i'), \ldots, \sigma_n(\theta_n))$$

$$\forall \theta = (\theta_1, \cdots, \theta_n) \in \Theta.$$

That is, independent of the other agents' strategies, a dominant strategy will maximize the utility of the agent playing this strategy. In most games, however, there is no dominant strategy (Nisan, Roughgarden, Tardos & Vazirani 2007, Wooldridge 2009).

2.1.2.3.2 Social choice A social choice function aggregates the preferences of all agents into a social choice, i.e., a world state (or outcome).

Definition 2.1.13 (**social choice**). *A social choice function $f : \Sigma \to \Omega$ selects an outcome $\omega \in \Omega$ from the strategy profile $\sigma(\theta) = (\sigma_1(\theta_1), \ldots, \sigma_n(\theta_n))$ with $(\theta_1, \ldots, \theta_n) \in \Theta$.*

That is, the social choice function (also denoted as social choice rule) selects a feasible world state for each possible combination of preferences and other characteristics (Dasgupta, Hammond & Maskin 1979).

2.1.2.3.3 Pareto optimality A game's outcome, selected by the social choice function f, is *pareto optimal* (also referred to as pareto efficient) if there is no other outcome that would improve one agent's utility without decreasing the utility of at least one other agent.

Definition 2.1.14 (**pareto optimality**). *An outcome of the social choice function $f(\sigma(\theta)) = \omega$ is pareto optimal if and only if*

$$\forall \omega' \notin f(\sigma(\theta)), \theta \in \Theta, u_i(\omega') > u_i(\omega) \Rightarrow \exists a_j \in \mathcal{A} : u_j(\omega') < u_j(\omega).$$

That is, no agent can gain utility without another agent losing utility.

2.1.2.3.4 Nash equilibrium A strategy profile is a Nash equilibrium (NE) if no agent has an incentive to deviate from its strategy, given that the other participating agents do not deviate (Nash 1950, Rasmusen 1989, Binmore 1992).

Definition 2.1.15 (**Nash equilibrium**). *A strategy profile $\sigma(\theta) = (\sigma_1(\theta_1, \ldots, \sigma_n(\theta_n))$ with $(\theta_1, \ldots, \theta_n) \in \Theta$ is a NE if and only if*

$$u_i(\sigma_1(\theta_1), \ldots, \sigma_i(\theta_i), \ldots, \sigma_n(\theta_n)) \geq u_i(\sigma_1(\theta_1), \ldots, \sigma_i(\theta_i'), \ldots, \sigma_n(\theta_n))$$

$$\forall u_i \in \mathcal{U}, \forall \theta_i' \neq \theta_i.$$

That is, a set of strategies is a NE if and only if they are the best response to each other. This type of NE is a *pure strategy Nash equilibrium*. However, not every game has a pure strategy NE, and some games have more than one pure strategy NE (Wooldridge 2009, pp. 230–233).

2.1.2.4 Social welfare

The social welfare denotes a *preference aggregation* over all participating agents, i.e., a social welfare function produces the social preferences of the agent society, taking the individual agents' preferences (Wooldridge 2009, p. 254). It allows for comparison of different mechanisms by comparing their respective outcomes (Sandholm 1999, p. 202). Different notions of social welfare have been studied in social choice theory and welfare economics (Sen 1970, Moulin 1988, Arrow, Sen & Suzumura 2002).

For quantitative preference structures, the term *utilitarian social welfare* denotes the sum of all agents' utility for a given alternative, i.e., the utilitarian social welfare for world state $\omega \in \Omega$, $sw_u(\omega)$, is defined as

$$sw_u(\omega) = \sum_{a_i \in \mathcal{A}} u_i(\omega).$$

To give another example, the egalitarian social welfare sw_{eg} is given by the least utility, i.e., $sw_{eg}(\omega) = \min_{a_i \in \mathcal{A}} u_i(\omega)$. Thus, sw_{eg} is defined by the utility of the agent that is 'worst off'. It is appropriate when a minimum amount of a large number of agents has to be satisfied. In other words, it provides a certain level of fairness.

Further notions of social welfare exist in literature which are suitable for different objective functions. The Nash product is given by the product of the individual utilities ($sw_{np}(\omega) = \prod_{a_i \in \mathcal{A}} u_i(\omega)$). It constitutes a compromise between utilitarian and egalitarian social welfare as it favors both increases in the sum of all agents' utility and inequality-reducing redistributions. The elitist social welfare is given by the largest utility – $sw_{el}(\omega) = \max_{a_i \in \mathcal{A}} u_i(\omega)$. It can be useful in applications where a single agent is required to achieve its goals (Chevaleyre et al. 2006, pp. 17–19). Further details along with further notions of social welfare can be found in the research area of welfare economics (Sen 1970, Moulin 1988, Arrow et al. 2002). In this research, the *utilitarian social welfare is utilized* as it is well suited to assess the system performance in terms of the maximal average profit of negotiating agents in electronic commerce transactions. This definition of social welfare is most commonly used in the

multiagent systems research area (Chevaleyre et al. 2006, p. 16). Thus, this research sets $sw := sw_u$.

Definition 2.1.16 (**social welfare**). *Let $\mathcal{A}$ be the set of agents, Ω be the set of world states, and $\mathcal{U}$ be the set of utility functions of the agents. The social welfare $sw_u(\omega) = \sum_{a_i \in \mathcal{A}} u_i(\omega)$ denotes a preference aggregation over all participating agents.*

2.1.2.5 Allocation procedures

Allocation procedures can either be distributed or centralized. In centralized allocation mechanisms such as combinatorial auctions (Cramton, Shoham & Steinberg 2006), the allocation is determined by a central entity. In contrast, the allocation result of distributed mechanisms emerges from multiple, local allocation steps. In multiagent resource allocation, the following three issues are of major importance (Chevaleyre et al. 2006, p. 19):

- *Protocols* address ontological as well as communication issues. That is, protocols define which deals are possible and which messages agents have to exchange in order to establish an agreement.
- *Strategies* have to be designed to allow agents to maximize their (expected) utility for a given protocol. Strategies are also closely related to protocols; protocols should incentivize agents to adopt a certain behavior (mechanism design).
- *Algorithms* comprise both the computational problem of winner determination (e.g., in auctions) as well as computing the best response to certain messages in a protocol (i.e., proposal messages). These algorithms are also closely related to the respective protocols and strategies, as finding optimal allocations in service networks is often computationally infeasible (Bo & Lesser 2010).

For the design of an allocation mechanism, the protocol constitutes the first building block. For the protocol to be developed, the most fundamental design decision comprises the distribution of the allocation mechanism, i.e., whether a centralized or distributed design is adopted. The corresponding agent strategies are most commonly analyzed based on game theory. The convergence property of an allocation framework corresponds to the question if for mutually beneficially deals, the allocation converges to an optimal allocation; this optimality can constitute the maximal utilitarian social welfare or pareto optimality (Endriss, Maudet, Sadri & Toni 2003, Chevaleyre et al. 2006).

2.1.2.5.1 Allocation mechanism An allocation mechanism defines the strategies available to each agent (i.e., the strategy space) and the function to determine the outcome of the game, based on the strategy profile (Parkes 2001, Nisan et al. 2007).

Definition 2.1.17 (**allocation mechanism**). *A mechanism* $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_n, m(\cdot))$ *defines the sets of possible strategies* $\Sigma_i \subseteq \Sigma \forall\, a_i \in \mathcal{A}$ *and an outcome function* $m : \Sigma \rightarrow \Omega$ *that maps strategy profiles to outcomes (Parkes 2001, Nisan et al. 2007).*

It is impossible to design a mechanism that is allocatively efficient, budget balanced, and incentive compatible, at least in settings with quasi-linear preferences (Green & Laffont 1977, Walker 1980, Myerson & Satterthwaite 1983, Hurwicz & Walker 1990).

2.1.2.5.2 Allocative efficiency An allocation mechanism is denoted as being allocatively efficient if it maximizes the social welfare for all strategy profiles.

Definition 2.1.18 (**allocative efficiency**). *The allocation of mechanism* $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_n, m(\cdot))$ *with* $m(\sigma(\theta)) = \omega$ *is allocatively efficient if and only if*

$$sw(m(\sigma(\hat{\theta}))) = sw(\omega) \geq sw(\omega') \forall\, \sigma(\hat{\theta}) \in (\Sigma_1, \ldots, \Sigma_n), \omega' \in \Omega,$$

where $\hat{\theta}$ denotes the types *reported* by the agents to the mechanism. These types do not necessarily represent the *true types* (Parkes 2001, Nisan et al. 2007).

2.1.2.5.3 Incentive compatibility In an incentive compatible, also denoted as strategy-proof or truthful, mechanism it is rational for the agents to report their true types (preferences), i.e., it is not rational for the agents to 'lie' about their preferences.

Definition 2.1.19 (**incentive compatibility**). *Let* $\theta = (\theta_1, \ldots, \theta_n)$ *be the true agent types and let* $\hat{\theta}_i$ *be the type reported by agent* $a_i \in \mathcal{A}$. *A mechanism* $\mathcal{M}$ *is incentive compatible if and only if*

$$u_i(m(\theta_i, \theta_{-i})) \geq u_i(m(\hat{\theta}_i, \theta_{-i})) \forall\, \hat{\theta}_i \in \Theta_i, a_i \in \mathcal{A}.$$

2.1.2.5.4 Individual rationality A mechanism is (ex-post) individually rational if participating agents always receive non-negative utility, i.e., no agent is worse-off by participating.

Definition 2.1.20 (**individual rationality**). *Let* $u_i(m(\hat{\theta}_i, \hat{\theta}_{-i}))$ *denote* a_i*'s* ***utility*** *if agent* $a_i \in \mathcal{A}$ *participates,* $\bar{u}_i(m(\hat{\theta}_i, \hat{\theta}_{-i}))$ *the utility if* a_i *does not participate. A mechanism* $\mathcal{M}$ *is individually rational if and only if*

$$u_i(m(\hat{\theta}_i, \hat{\theta}_{-i})) \geq \bar{u}_i(m(\hat{\theta}_i, \hat{\theta}_{-i})) \forall\, a_i \in \mathcal{A}.$$

However, ex-post individual rationality always requires that *concrete* outcomes of participating agents produce non-negative utility. Therefore, the concept of ex-ante individual rationality is introduced that refers to expected utilities.

Definition 2.1.21 (**ex-ante individual rationality**). *Let* $E(u_i(m(\hat{\theta}_i, \hat{\theta}_{-i})))$ *denote* a_i*'s* ***expected utility*** *if agent* $a_i \in \mathcal{A}$ *participates,* $E(\bar{u}_i(m(\hat{\theta}_i, \hat{\theta}_{-i})))$ *denote the expected utility if* a_i *does not participate. A mechanism* $\mathcal{M}$ *is* ***ex-ante individually rational*** *if and only if*

$$E(u_i(m(\hat{\theta}_i, \hat{\theta}_{-i}))) \geq E(\bar{u}_i(m(\hat{\theta}_i, \hat{\theta}_{-i}))) \forall\, a_i \in \mathcal{A}.$$

It is important to note that for ex-ante individual rationality, distributional information about the other agents' preferences θ_{-i} has to be known (Parkes 2001, Nisan et al. 2007).

2.1.2.5.5 Budget balance For quasi-linear preferences of the agents, the outcome function $m(\cdot)$ of mechanism $\mathcal{M}$ can be decomposed into an allocation function $x(\hat{\theta}) \in \Omega$ and a payment function $p_i(\hat{\theta})$ that determines the payments made or received by the agents.

Definition 2.1.22 (**budget balance**). *Let* $\mathcal{M}$ *be a mechanism and* $m(\hat{\theta}) = (x(\hat{\theta}), p_1(\hat{\theta}), \ldots, p_n(\hat{\theta}))$. $\mathcal{M}$ *is* ***budget balanced*** *if and only if*

$$\sum_{a_i \in \mathcal{A}} p_i(\hat{\theta}) = 0.$$

That is, there are no net money transfers into or out of the allocation system (Parkes 2001, Nisan et al. 2007).

Definition 2.1.23 (**weak budget balance**). *Let* $\mathcal{M}$ *be a mechanism and* $m(\hat{\theta}) = (x(\hat{\theta}), p_1(\hat{\theta}), \ldots, p_n(\hat{\theta}))$. $\mathcal{M}$ *is* ***weakly budget balanced*** *if and only if*

$$\sum_{a_i \in \mathcal{A}} p_i(\hat{\theta}) \geq 0.$$

Thus, for *weak* budget balance, there can be net payments by the agents to the mechanism, but not vice versa (Parkes 2001, Nisan et al. 2007).

2.1.2.5.6 Allocation complexity The complexity of an allocation framework comprises (i) the computational complexity as well as (ii) the communication complexity. Computational complexity theory is concerned with the question how many computational resources are required to solve a given problem. Therefore, it provides a classification of problems of different complexity (Garey & Johnson 1979, Papadimitriou 1994).

Communication complexity is concerned with (i) the number of required agreements (or deals) to arrive at an optimal allocation outcome, (ii) the number of dialogue moves required to establish an agreement, (iii) the expressiveness of the communication language required, and (iv) the complexity of individual agent's reasoning tasks deciding on dialogue moves (Endriss & Maudet 2005, p. 96). The complexity of agents' reasoning for a given allocation mechanism strongly influences the computational complexity, since this reasoning is required to arrive at an allocation outcome.

2.1.3 Services & service networks

This section introduces fundamental concepts of services, especially electronic and software services, and respective service parameters. In addition, networks of electronic services in service-oriented architectures are described. Finally, service level agreements – formal statements of the obligations and guarantees in a business relationship regarding services – are discussed.

2.1.3.1 Service

The concepts of business services has been studied in research literature over the last decades, however, no commonly accepted definition exists (Hill 1977, Hill 1999, Gadrey 2000). This work is concerned with *electronic* services – services provided over electronic networks (Rust & Kannan 2002). Moreover, the 'service' concept of this research is limited to *software* services (Booth, Haas, McCabe, Newcomer, Champion, Ferris & Orchard 2004, Papazoglou et al. 2008).

Definition 2.1.24 (**service**). *A service $s_{ij} \in \mathcal{S}$ is a software activity performed by a service provider $a_j \in \mathcal{A}_{SP}$ over electronic networks for a customer $a_i \in \mathcal{A}$, based on an agreement $x(s_{ij})$, with well-defined service properties (Hill 1977, Hill 1999, Gadrey 2000, Rust & Kannan 2002, Booth et al. 2004, Papazoglou et al. 2008).*

The definition of services is based on the assumptions that (i) services are software activities by service providers for customers over electronic networks (Hill 1977, Hill 1999, Gadrey 2000, Christensen, Curbera, Meredith & Weerawarana 2001, Rust & Kannan 2002); (ii) services have well-defined inputs, outputs, preconditions, and effects (Booth et al. 2004, Martin, Burstein, Hobbs, Lassila, McDermott, McIlraith, Narayanan, Paolucci, Parsia, Payne, Sirin, Srinivasan & Sycara 2004); (iii) services provide reuse and interoperability of software, potentially under distributed ownership and control (Papazoglou 2003); and (iv) services are provided based on explicit formal statements of obligations and guarantees, regarding certain service properties (Verma 1999). The following sections discuss these assumptions in detail.

2.1.3.1.1 Goods, services, and tangibility Tangibility has traditionally defined the distinction between goods and services: goods are tangible (material) and services are intangible (immaterial). However, intangible goods such as information goods exist in practice and differ from services. Thus, a distinction of tangibles, intangibles, and services is evident. The differences of goods and services are especially notable in the production output. Good production outputs are entities of economic value and ownership of goods can be established as well as traded. Further, goods can be consumed independently of the time and location of their production. These characteristics of goods are also inherent in intangible goods (e.g., information and software) (Hill 1999). In contrast, services cannot be produced without the agreement and participation of the service customer and service production outputs do not constitute entities that exist independently of service provider and customer, i.e., services are not separable from the service customer (Hill 1977). A detailed analysis of the roles of customers in service production can be found in (Bitner, Faranda, Hubbert & Zeithaml 1997). However, the integration of the customer in value creation cannot be applied as a criterion to differentiate goods and services. Recent production strategies for goods consider an integration of the customer early in the production process to facilitate advanced flexibility in addressing individual customer requirements, i.e., mass customization (Piller, Moeslein & Stotko 2004).

There is also criticism regarding this characterization of service production outputs. Gadrey (2000) states that outputs of activities are material if the activities transform the state of reality and if this transformation is observable. Consequently, all service activities have observable material outputs (Gadrey 2000, p. 372). In addition, the identity of entities, which is inherent in Hill's analysis, is firmly rooted in social conventions. In other conventions it could

be considered that trading and altering (e.g., repairing) activities on an entity change its identity (Gadrey 2000, pp. 379–380).

Since no generally accepted definition of the service concept exists, especially across different research and application areas, this section characterizes the service concept based on commonly accepted attributes. Despite the differences, Hill (1977, 1999) and Gadrey (2000) determine common characteristics of the service concept:

A service is an *activity* that is *performed* by a service provider for a service consumer (customer) based on an *agreement*, *results* in a *change* of condition of an *entity*, and has an *output* that is *not separable* from the altered entity (Hill 1977, Hill 1999, Gadrey 2000).

The discussion shows that the service concept differs from the concepts of tangible and intangible goods in several attributes. For example, services are no entities and thus cannot be stocked (Hill 1999, p. 441). Consequently, the production of services significantly differs from the production of goods. In addition, electronic and software services can be further characterized, which is presented in the following sections.

2.1.3.1.2 Electronic, software, and Web services This research assumes services to be software activities by service providers for customers via electronic networks (Hill 1977, Hill 1999, Gadrey 2000, Christensen et al. 2001, Rust & Kannan 2002). These services have well-defined inputs, outputs, preconditions, and effects (Booth et al. 2004, Martin et al. 2004).

Information and communication technologies and the internet have significant impact on the service economy. There is an emerging paradigm shift from the electronic commerce with physical goods to *electronic services* (Rust & Kannan 2003). This paradigm defines electronic services as services provided over electronic networks (Rust & Kannan 2002). In contrast to the general service concept, electronic services show specific characteristics. Since these services are provided over networks, the consumption of the services is virtually independent of the location of their production and their activities cannot directly change conditions of physical entities. Although most findings apply to the general service concept, this work focuses on a subset of electronic services, namely software services, in the following. That is, the services provided by SPs over networks constitute remote execution of software, with well-defined inputs, outputs, preconditions, and effects (Martin et al. 2004) – Web services.

The concept of a Web service (WS) generally refers to functionality provided as a software service which is accessible over the Web in a standardized manner. The concept is independent of concrete realizations and provides a con-

cretization of objectives such as reuse and interoperability in distributed applications. It reduces heterogeneity by standardization, i.e., it provides platform-independent means for distributed interoperability.

There is a significant number of commonly accepted standards around WSs provided by the World Wide Web Consortium (W3C), commonly referred to as WS-*. The W3C defines WSs as software systems which support machine-to-machine interactions over networks. A W3C WS's interface is described in machine-readable form, specifically WSDL (Christensen et al. 2001). Interactions with WSs are performed using SOAP messages (Gudgin, Hadley, Mendelsohn, Moreau, Nielsen, Karmarkar & Lafon 2007), HTTP, XML serialization, and other WS-* standards (Booth et al. 2004).

2.1.3.1.3 Service-oriented computing and service-oriented architecture Services are assumed to provide reuse and interoperability of software, potentially under distributed ownership and control (Papazoglou 2003).

Service-oriented computing (SOC) is a paradigm that utilizes software services as building blocks for applications. It is regarded as a derivative of object and component orientation for distributed computing. The service-oriented architecture (SOA), which aims at reuse and interoperability of software services, is itself a building block of SOC. However, in existing literature, the term SOA is often used synonymously with SOC. This research regards SOA as the architectural basis of SOC (Papazoglou 2003).

SOA facilitates the utilization and organization of capabilities, potentially under distributed ownership and control; it has gained significant relevance for software development in the last decade. SOA-based technology provides means for SOC developments. The OASIS Reference Model for Service-Oriented Architecture (MacKenzie, Laskey, McCabe, Brown & Metz 2006) defines basic SOA principles and vocabulary for a common understanding of SOA. Key concepts of the SOA reference model include visability, realized by means of service descriptions, interaction, mediated by message exchange, and effects, denoting real-world changes as a result of capability utilizations. Further, SOA provides an abstraction for service utilization, i.e., realization details (e.g., implementation, deployment, etc.) are hidden from the service customer. The abstracted concepts are independent of concrete specifications and technologies used for their realization. Although SOA is most commonly implemented using Web services, it is technology-independent and can be implemented by other means and technologies (MacKenzie et al. 2006).

Grid computing has emerged in the mid 1990s from conventional distributed computing with a focus on large-scale resource sharing. Grid computing and re-

lated technologies have since then changed from distributed high performance computing (HPC) to a standards-base, flexible coordinated resource sharing over organizational boundaries (Foster, Kesselman & Tuecke 2001). The Open Grid Services Architecture (OGSA) (Foster, Kishimoto, Savva, Berry, Djaoui, Grimshaw, Horn, Maciel, Siebenlist, Subramaniam, Treadwell & Reich 2006) provides a high-level framework for service-oriented grid architecture, mostly based on Web service standards. The building blocks are means for homogeneous access to heterogeneous distributed resources in the grid fabric, as well as a SOA providing higher-level functionality. This functionality of the OGSA is outlined in the specification's capability descriptions. Execution Management Services (EMS) constitute the most relevant OGSA capability for resource allocation and scheduling. They include: (i) service containers which encapsulate running entities; (ii) Job Managers (JMs) which are high-level services, encapsulating execution of sets of jobs; and selection services which consist of (iii) execution planning services (EPS), (iv) candidate set generators (CSGs), and (v) reservation services. Information services make metadata of resources available. Data services provide management, access to and update capabilities for data resources. In addition, they are concerned with transfers of data between resources.

During the last years, Cloud computing has emerged, operationalizing many of Grid computing's promises while differing in detail. For example, access to existing Cloud computing services is mostly based on WS standards (i.e., WS-*), though the actual functionality commonly does not adhere to any standard. There is a large variety of definitions for the term Cloud computing and the current hype is turning it to a widely used term referring to almost any service related to outsourced hosting and computing resources (Vaquero, Rodero-Merino, Caceres & Lindner 2009, p. 50). Vaquero et al. (2009) analyze more than 20 definitions for Cloud computing to arrive at a comprehensive but general definition:

> "Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs." (Vaquero et al. 2009, p. 51)

Thus, scalability, pay-per-use utility model and virtualization constitute the

essential features to arrive at a minimal definition.

Besides the basic definition, there exist different levels of abstractions to group certain services in Cloud computing. The infrastructure as a service (IaaS) paradigm describes the resource-near provision of virtualized computing resources over a network, i.e., IaaS provides remote access to virtualized computing infrastructure. This infrastructure can be utilized by other SPs to provide their services in the Cloud ecosystem.

In contrast, platform as a service (PaaS) refers to a higher abstraction layer in which software platforms are provided to customers instead of virtualized hardware resources. The software as a service (SaaS) paradigm constitutes an approach where applications directly run on remote servers instead of being executed locally on the client computer (Vaquero et al. 2009).

2.1.3.1.4 Service properties Service properties are commonly categorized in functional and non-functional properties. Functional properties describe *what* a service does and non-functional properties describe *how* a services does it (O'Sullivan, Edmond & ter Hofstede 2002). However, the distinction of functional and non-functional service properties is often ambiguous and differs across application domains. This section provides an overview of relevant concepts and references with a focus on non-functional properties of electronic services.

Functional properties describe *what* a service does, i.e., capabilities of a service (Oaks, ter Hofstede & Edmond 2003). This includes (i) actions that a service performs, (ii) valid inputs of a service, (iii) conditions of a service's operations in term of preconditions and postconditions, as well as (iv) outputs that a service produces (also referred to as effects of a service). Functional service properties are defined in syntactic or semantic service specifications and are fundamental to discover and select appropriate services. However, a service description is only complete if non-functional properties are considered (O'Sullivan et al. 2002, p. 118). For example, a service may provide the required functionality while it cannot fulfill timely constraints of service provision, i.e., insufficient availability or execution time.

Non-functional service properties can be regarded as descriptions of limitations of the functionality of a service in terms of *how* a service provides a capability (O'Sullivan et al. 2002). Non-functional properties describe qualitative and financial characteristics of a service (Menasce 2004). Qualitative service characteristics, referred to as quality of service (QoS) properties, originate from the telecommunication and network domains, though extensive research work has been conducted in the service domain in the last decade (e.g.,

(O'Sullivan et al. 2002, Cardoso, Sheth, Miller, Arnold & Kochut 2004, Jaeger, Rojec-Goldmann & Mühl 2004, Jaeger & Ladner 2006)). The QoS profile of a service is described as a set of quantifiable, measurable service parameters that refer to qualitative characteristics. These parameters are commonly denoted as QoS parameters. In this context, the term service level refers to QoS parameter values. Financial characteristics include cost of usage as well as rewards and penalties if defined service levels are met or missed. Further details on service level, reward, and penalty definitions as part of an agreement between service provider and customer are given in the following.

2.1.3.1.5 Service level agreement Services are assumed to be provided based on explicit formal statements of obligations and guarantees, regarding certain service properties (Verma 1999).

An explicit formal statement of the obligations and guarantees regarding services in a business relationship is referred to as a service level agreement (SLA) (Verma 1999). Thus, a SLA provides the operational definition of a service as part of a contract between a service provider and a service customer in a SN. In real-world SNs, contractual agreements exist along the flows of services. For individual service requests, agreements have to be established. These agreements depend either directly or indirectly on other agreements along the SN (e.g., for procurement or outsourcing). SOA-related SLA approaches aim at providing an abstraction of the service while facilitating measurement and monitoring of service properties agreed upon (Czajkowski et al. 2004, pp. 264–265).

Current technical SLA approaches (e.g., WS-Agreement (Andrieux, Czajkowski, Dan, Keahey, Ludwig, Nakata, Pruyne, Rofrano, Tuecke & Xu 2007)) are limited to independent bipartite (1:1) relationships. Thus, current SLA management is not capable to represent the full complexity of SLAs existing in real-world service industries. The reason is that it originates from high performance computing (HPC), which concerns purely technical attributes of limited expressiveness which can be directly forwarded and broken-down to the lower level stages in a 1:1 manner. Multi-tier real-world business relationships can be mapped to sets of technical SLAs. The problem is, however, that current SLA approaches do not consider the dependencies between subsets of SLAs in SNs which are essential for individualization; i.e., there is a mismatch between the technical representation of SLAs in SOA and the required coverage of real-world business agreements, for which dependencies exist (e.g., hierarchical).

2.1.3.2 Service network

In this research, networks of software services in which composite services are offered by service providers (SPs), are denoted as *service networks* (SNs). The provision of these software services in SNs requires non-consumable *resources* (i.e., hardware and software) that are limited and possessed by SPs. The SPs' resources contribute to the production of service applications in SNs by means of service provision and composition. SNs are characterized by requiring multiple SPs' resources for the provision of composite services to customers. For the supply of software services to customers, software SPs have to allocate their scarce computational resources of a certain quality to customer requests, i.e., reserve respective resources for software services on the contracted service level. In return, SPs receive monetary compensations for providing services to customers.

The SPs' resources' contribution is carried out by production; i.e., the combination of production (input) factors and their transformation to services (output factors). Service flows are directed and primarily carried out from upstream agents down to the customer agent, which does itself not show primary value flows to other agents (Kirn 2008). The number of service providers can be different on each tier. Figure 2.3 shows a generic SN model: nodes represent agents, edges represent potential provision of services.

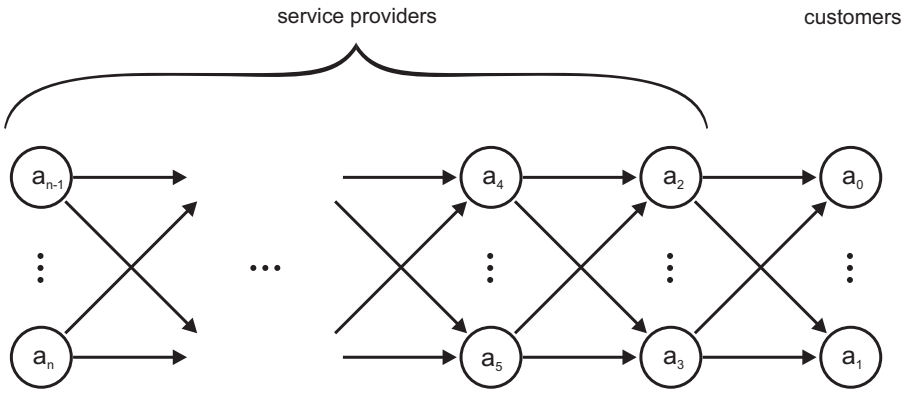

Figure 2.3: Generic service network model.

Definition 2.1.25 (**service network**). *A service network (SN) is a network of software services in which composite services $s_{ij} \in \mathcal{S}$ are offered by service providers $a_j \in \mathcal{A}_{SP}$ to customers $a_i \in \mathcal{A}_C$ based on agreements $x(s_{ij})$, where composite services s_{ij} aggregate multiple software services $\{s_{jk} : s_{jk} \in \mathcal{S}, a_k \in \mathcal{A}_{SP}\}$ into software application services (Papazoglou et al. 2008, Blau et al. 2009).*

The definition of SNs is based on the assumptions that (i) in SNs, composite software services aggregate multiple software services into software application services (service composition) based on workflows (McIlraith, Son & Zeng 2001, Papazoglou et al. 2008); (ii) service parameter aggregations for composite services depend on parameter types, as well as workflow patterns (van der Aalst, ter Hofstede, Kiepuszewski & Barros 2003, Jaeger et al. 2004, Jaeger & Ladner 2006, Karaenke et al. 2013); and (iii) supply and demand in SNs are allocated by service auctions (Smith 1989). The following sections discuss these assumptions in detail.

2.1.3.2.1 Composite service In SNs, composite software services aggregate multiple software services into software application services (service composition) based on workflows (McIlraith et al. 2001, Papazoglou et al. 2008).

Services are categorized in constituent (elementary, simple, atomic, primitive) and composite (complex) services (McIlraith et al. 2001, Papazoglou et al. 2008). Elementary services provide their functionality without requiring other services as input factors (e.g., database or sensor services). In contrast, composite services rely on multiple (elementary or composite) further services for their execution. Complex services often include functionality for interactions or conversions between composed services and the customer, enabling customers to make decisions (McIlraith et al. 2001). For example, a composite service can be a composition of elementary services for storage and computational resources.

The composition of multiple services to provide a composite service to a customer considers SNs as a solution space to fulfill the customer's demand. Composite services constitute collections of services that are governed under a shared workflow (directed acyclic graph, DAG). Figure 2.4 shows an example workflow, which contains six tasks ($tk_1 - tk_6$) and four gateways (represented by diamond shapes).

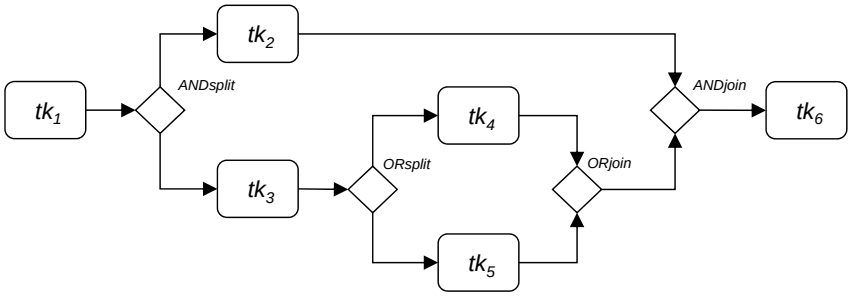

Figure 2.4: Example workflow.

To instantiate the workflow for different services, the tasks are separated from actual services by means of a binding, which is part of the succeeding definition. Let

- TK be a finite, non-empty set of tasks $tk \in TK$;
- GW be a finite, non-empty set of gateways $gw \in GW$;
- ARC be a set of arcs connecting tasks with tasks, tasks with gateways, and gateways with gateways, thus $ARC \subseteq (TK \times TK) \cup (TK \times GW) \cup (GW \times GW)$;
- $cwg \in CWG$ be a function which assigns a type to each gateway; $cwg : GW \rightarrow \{XORsplit, XORjoin, ANDsplit, ANDjoin, ORsplit, ORjoin, DISCjoin, Loop\}$;
- $bng \in BNG$ be a binding function, which assigns a service $s \in \mathcal{S}$ to each task, i.e., $bng : TK \rightarrow \mathcal{S}$.

Then a *workflow* can be defined as follows.

Definition 2.1.26 (**workflow**). *A **workflow** $wf \in WF$ is a directed acyclic graph with $WF = (TK, GW, ARC, CWG)$.*

Further, a *composite service* – the result of a service composition – can be defined as follows.

Definition 2.1.27 (**composite service**). *A **composite service** $cs \in CW$ is a tuple, $cs = (wf, bng) \in CW = (WF, BNG)$, of workflow $wf \in WF$ and binding function $bng \in BNG$.*

The model is restricted to well-formed (structured) workflows, where W has exactly one start and end node and is weakly connected. The correct usage of the different gateway types, thus how to connect nodes by arcs, is given as follows.

- Each task $tk \in TK$ has exactly one input arc and one output arc, i.e., $|\bullet\, tk| = |tk \,\bullet| = 1 \; \forall tk \in TK$.
- Each gateway $gw \in GW$ with $cwg(gw) \in \{XORsplit, ANDsplit, ORsplit\}$ has exactly one input arc and at least two output arcs, i.e., $|\bullet\, gw| = 1$ and $|gw \,\bullet| \geq 2$.
- Each gateway $gw \in GW$ with $cwg(gw) \in \{XORjoin, ANDjoin, ORjoin\}$ has at least two input arcs and exactly one output arc, i.e., $|\bullet\, gw| \geq 2$ and $|gw \,\bullet| = 1$.

- Each gateway $gw \in GW$ with $cwg(gw) = Loop$ has one input arc and two output arcs, i.e., $|\bullet\, gw| = 1$ and $|gw \bullet\,| = 2$. Let $arc_1 = (n_1, gw)$ be the input arc, then one output arc arc_2 connects back to node n_1 with $arc_2 = (gw, n_1)$, and one output arc arc_3 connects to another node with $arc_3 = (gw, n_2)$ and $n_1 \neq n2$.

This model is not bound to a particular workflow language such as the Business Process Model and Notation (BPMN) (OMG 2011*a*) or the Web Services Business Process Execution Language (WS-BPEL) (OASIS 2007), but conveys common constructs of workflow languages (van der Aalst et al. 2003, Jaeger et al. 2004, Jaeger & Ladner 2006, Karaenke et al. 2013).

2.1.3.2.2 Service parameter aggregation The service parameter aggregations for composite services depend on parameter types, as well as workflow patterns (van der Aalst et al. 2003, Jaeger et al. 2004, Jaeger & Ladner 2006, Karaenke et al. 2013).

Composite WSs play an important role in SOC. They allow meeting individual service customer demand that cannot be fulfilled by a single service. Business processes are realized by means of composing constituent services under a shared workflow (Dustdar 2004). Therefore, composite services must be configured so that they both meet the customer demand and preserve efficiency on the service provider side. In the past years, a significant number of composition approaches has emerged. An essential subtask of service composition is determining the QoS of composite services. This task is also called QoS aggregation (Jaeger et al. 2004) – aggregation of non-functional properties.

The QoS of a composite WS provided in a SN depends on two determinants: the QoS parameters and execution order of constituent services, defined in a workflow. QoS parameters commonly do not adhere to any standards regarding their number, naming, data types, and conceptualization. QoS aggregation has been subject of much research in SOC (O'Sullivan et al. 2002).

The basic approach for QoS aggregation is standardization on the syntactical level, i.e., providing standardized formats for representing parameters in service descriptions and SLAs. Harmonization on the semantic level addresses the problem of QoS parameter heterogeneity by defining a set of QoS parameters. In the Semantic Web services research field, QoS ontologies have been proposed to specify the conceptualization of QoS parameters formally (e.g., (Dobson, Lock & Sommerville 2005, Muñoz Frutos, Kotsiopoulos, Vaquero & Merino 2009)). These ontologies contain taxonomies of QoS parameters and define attributes such as application domains and units of measurement. How-

ever, beyond identifying similar parameters, these ontologies do not provide information that could guide the QoS aggregation.

For QoS aggregation, one class of existing work proposes to state aggregation functions explicitly (e.g. (ul Haq, Huqqani & Schikuta 2009)). The drawback of this approach is that service providers need to determine the function for all potential parameters. Another class of approaches abstracts from single parameters and identifies sets of several parameters for which aggregation functions are provided, assuming that the parameter sets are extensible without fundamentally altering the overall approach (e.g., (Zeng, Benatallah, Ngu, Dumas, Kalagnanam & Chang 2004, Cardoso et al. 2004)). It is an important insight to abstract from diverse and domain-specific QoS parameters and arrive at parameter types with regard to their aggregation. QoS aggregation based on graphs is proposed in (Cardoso et al. 2004) - the graph reduction works inversely to operations that designers use when crafting a workflow. Jaeger et al. (2004) ground QoS aggregation on workflow patterns (van der Aalst et al. 2003, Jaeger et al. 2004, Jaeger & Ladner 2006), deriving so-called composition patterns and respective aggregation functions. This research utilizes QoS parameter aggregation based on composition patterns (Karaenke et al. 2013).

The composite QoS depends on two determinants: QoS parameters and workflow. QoS parameters are diverse with regard to number, name, data type, and conceptualization and they rarely adhere to any standard. Instead of contributing to their harmonization, a classification with regard to their aggregation exclusively is proposed. This classification is built upon the following principle: If two parameters share the same aggregation function, then they belong to the same parameter type, regardless of other characteristics. Applying this principle results in five parameter types:

- Type 1: Parameters that are always summed up along all deterministic paths of the workflow (e.g., cost of service execution). For non-deterministic paths, the aggregation depends on whether the lower or upper bound is calculated.
- Type 2: Parameters for which the critical path is determined by the maximal values in parallel executions (e.g., duration of execution time).
- Type 3: Parameters that denote a capacity (e.g., throughput).
- Type 4: Parameters that denote a probability (e.g., uptime probability).
- Type 5: Parameters for which the critical path is determined by the

minimal values in parallel executions (e.g., key length of the services encryption algorithm).

The second determinant workflow is analyzed by means of workflow patterns (Jaeger et al. 2004); this analysis arrives at a set of composition patterns $CP = \{Sequence, Loop, XORXOR, ANDAND, ANDDISC, ORDISC, ORDISC\}$. For instance, the $ORDISC$ pattern describes an $ORsplit$ followed by $ORjoin$. $ANDDISC$ and $ORDISC$ describe $ANDsplit$ and $ORsplit$ followed by a m-out-of-n join (discriminator). The latter is used when an activity is triggered after m out of n branches have been completed (e.g., to improve response time, two databases are queried and the first result is processed, the second is ignored) (van der Aalst et al. 2003). Both determinants span a matrix of cases, with each cell giving the respective aggregation function. This research defines the set of all aggregation functions as:

$$AF = \{af_{cp,pt} : cp \in CP \wedge pt \in PT_s, af_{cp,pt} : \mathbb{R}^n \to \mathbb{R}\}$$

PT_s is the set of parameter types of service $s \in \mathcal{S}$. n denotes the number of parameters to be aggregated; the domain of $af_{cp,pt}$ consists of n-tuples with QoS parameters of the constituent services. Since the aggregation function depends on (cp, pt), there exist $5 \cdot 7 = 35$ aggregation functions.

Several pairs (cp, pt) share the same aggregation function. Thus, the number of functions can be reduced to only seven. This research defines generic aggregation functions $af \in AF$ as shown in table 2.1, with $pv_1, \ldots, pv_n$ denoting the parameter values of the services to be aggregated, and k for the number of iterations in a *Loop* pattern. To each pair (cp, pt), this work assigns the respective aggregation function. Following (Jaeger et al. 2004), upper and lower bound are distinguished (as shown in tables 2.2 and 2.3). This distinction is necessary to assess whether a particular parameter value meets the guarantee defined in the SLA. Calculating expected or average parameter values is only possible if the distribution of the non-deterministic control flows (i.e., $XORXOR$, $ANDDISC$, $ORDISC$, and $ORDISC$) is known (Jaeger et al. 2004); however, this information is not available (Karaenke et al. 2013).

Table 2.1: Generic aggregation functions.

aggregation function	**definition**
af_{sum}	$af_{sum}(pv_1, \ldots, pv_n) = \sum_{i=1}^{n} pv_i$
$af_{product}$	$af_{product}(pv_1, \ldots, pv_n) = \prod_{i=1}^{n} pv_i$
af_{max}	$af_{max}(pv_1, \ldots, pv_n) = \max(pv_1, \ldots, pv_n)$
af_{min}	$af_{min}(pv_1, \ldots, pv_n) = \min(pv_1, \ldots, pv_n)$
af_{power}	$af_{power}(pv_1, \ldots, pv_n) = (pv_1, \ldots, pv_n)^k$
af_{linear}	$af_{linear}(pv_1, \ldots, pv_n) = k \cdot (pv_1, \ldots, pv_n)$
$af_{identity}$	$af_{identity}(pv_1, \ldots, pv_n) = (pv_1, \ldots, pv_n)$

Table 2.2: Aggregation functions for *Sequence*, *Loop*, and *XORXOR* composition patterns.

type	**bound**	*Sequence*	*Loop*	*XORXOR*
1	upper	af_{sum}	af_{linear}	af_{max}
	lower	af_{sum}	af_{linear}	af_{min}
2	upper	af_{sum}	af_{linear}	af_{max}
	lower	af_{sum}	af_{linear}	af_{min}
3	upper	af_{min}	$af_{identity}$	af_{max}
	lower	af_{min}	$af_{identity}$	af_{min}
4	upper	$af_{product}$	af_{power}	af_{max}
	lower	$af_{product}$	af_{power}	af_{min}
5	upper	af_{min}	$af_{identity}$	af_{max}
	lower	af_{min}	$af_{identity}$	af_{min}

Table 2.3: Aggregation functions for *ANDAND*, *ANDDISC*, *OROR*, and *ORDISC* composition patterns.

type	**bound**	*ANDAND*	*ANDDISC*	*OROR*	*ORDISC*
1	upper	af_{sum}	af_{sum}	af_{sum}	af_{sum}
	lower	af_{sum}	af_{sum}	af_{min}	af_{min}
2	upper	af_{max}	af_{max}	af_{max}	af_{max}
	lower	af_{max}	af_{min}	af_{min}	af_{min}
3	upper	af_{min}	af_{max}	af_{max}	af_{max}
	lower	af_{min}	af_{min}	af_{min}	af_{min}
4	upper	$af_{product}$	af_{max}	af_{max}	af_{max}
	lower	$af_{product}$	$af_{product}$	$af_{product}$	$af_{product}$
5	upper	af_{min}	af_{max}	af_{max}	af_{max}
	lower	af_{min}	af_{min}	af_{min}	af_{min}

2.1.3.2.3 Service auction Different forms of auctions constitute specific allocation mechanisms (e.g., English and Dutch auctions). Auctions are well-suited when the costs of the traded products are private information to the producing agents, i.e., the SP agents' valuation for the services. In this thesis, the SP agents' valuations for services corresponds to the true cost of service productions. Auction mechanisms can be designed, such that SP agents have an incentive to reveal their true costs for services (Vickrey 1961). This is referred to as incentive compatibility of a mechanism (section 2.1.2.5.3). In addition, auction mechanisms are suitable for allocation and pricing if the traded product has special or unusual characteristics (Smith 1989). The characteristics of composite services in SNs are highly individual, depending on the specific preferences of the requesting customer agent. Therefore, this research assumes that supply and demand in SNs are allocated by service auctions, where customer agents initiate the allocation by submitting service requests. This section presents a formal, game-theoretic framework for service auctions in multi-tier SNs, based on the definitions in this chapter. It is utilized for an analysis of the problem addressed, as well as for the construction of the proposed protocol.

In this framework, SPs and customers are represented by autonomous software agents (section 2.1.1.1). The actors delegate their objectives to their software agents, which conduct the negotiations for service provision on their behalf (Huhns et al. 2005). Thus, the edges in SNs represent the allocations of services (i.e., provision of services) among SP and customer agents. SP agents on different tiers contribute to the supply of service applications by means of service provision and service composition. This contribution is carried out by production; i.e., the combination of production (input) factors and their transformation to services (output factors). The number of SPs can be different on each tier. Since this research represents each actor in a SN by a software agent, a SN is defined by service agreements between agents.

2.1.3.2.3.1 Agent From definition 2.1.1, the set of agents is defined as $\mathcal{A}$. This research further differentiates the agents in this set as follows. Let $\mathcal{A}_C$ denote the set of all customer agents, $\mathcal{A}_{SP}$ the set of all service provider agents, and $\mathcal{A} = \mathcal{A}_C \cup \mathcal{A}_{SP}$.

2.1.3.2.3.2 Service Let $\mathcal{S}$ be the set of all available services. $s_{ij} \in \mathcal{S}$ denotes the service potentially provided from SP agent a_j to agent a_i (definition 2.1.24).

2.1.3.2.3.3 Time This work considers an allocation process in discrete time periods $t \in \mathcal{T} = \{0, \ldots, T\}$.

2.1.3.2.3.4 Offer Since customer agents initiate the interactions, $\mathcal{O}^t \subseteq \mathcal{O}$ denotes the set of 'offers' to SP agents in t to provide a certain service (i.e., offers represent requests for quote/proposal), stating the maximal payment the customer agents are willing to accept for service provision; $o_{ij}^t \in \mathcal{O}$ denotes the offer from agent $a_i \in \mathcal{A}$ to agent $a_j \in \mathcal{A}_{SP}$ for service s_{ij} in time period $t \in \mathcal{T}$. These offers are revealed to SP agents and do not necessarily represent the true reserve prices of service customers.

2.1.3.2.3.5 Bid $\mathcal{B}^t \subseteq \mathcal{B}$ is the set of bids the SP agents submit to customer agents in t in response to offers they have received; $b_{ij}^t \in \mathcal{B}$ denotes the bid from agent $a_j \in \mathcal{A}_{SP}$ to agent $a_i \in \mathcal{A}$ for the provision of service s_{ij} in time period $t \in \mathcal{T}$.

2.1.3.2.3.6 Valuation Let $v_i : \mathcal{S} \to \mathbb{R}$ denote the valuation function of customer agent $a_i \in \mathcal{A}_C$ for the provision of services, i.e., $v_{ij} := v_i(s_{ij})$ denotes the valuation of customer agent $a_i \in \mathcal{A}_C$ for the provision of services $s_{ij} \in \mathcal{S}$ by service provider agent $a_j \in \mathcal{A}_{SP}$.

2.1.3.2.3.7 Cost Let $c_j : \mathcal{S} \to \mathbb{R}$ denote the cost function of service provider agent $a_j \in \mathcal{A}_{SP}$ for the provision of services, i.e., $c_{ij} := c_j(s_{ij})$ denotes the cost of SP agent $a_j \in \mathcal{A}_{SP}$ when using its own resources for providing s_{ij} to agent $a_i \in \mathcal{A}$ with $c_{ij} > 0$.

2.1.3.2.3.8 Capacity Let $w_j : \mathcal{S} \times \mathcal{R}_j \to \mathbb{R}$ denote the capacity function of service provider agent $a_j \in \mathcal{A}_{SP}$ for the provision of services, i.e., $w_{ij} := w_j(s_{ij})$ denotes the capacity requirements (i.e., resources from $\mathcal{R}_j$) for the provision of services $s_{ij} \in \mathcal{S}$ by service provider agent $a_j \in \mathcal{A}_{SP}$. Further, let W_j denote the total capacity of SP agent a_j.

2.1.3.2.3.9 Allocation The binary functions $x^t \in \mathcal{X}$ with $x^t : \mathcal{S} \to \{0, 1\}$ denote whether the service $s_{ij} \in \mathcal{S}$ is allocated to the SP agent a_j in period t ($x^t(s_{ij}) = 1$) or not ($x^t(s_{ij}) = 0$), i.e., whether a contract for service s_{ij} has been established between customer and SP agent or not. $\mathcal{S}^t \subseteq \mathcal{S}$ with $\mathcal{S}^t = \{s_{ij} : x^t(s_{ij}) = 1\}$ denotes the set of services contracted in period $t \in \mathcal{T}$.

2.1.3.2.3.10 Payment The function $\psi : \mathcal{O} \times \mathcal{B} \to \mathbb{R}$ determines the payment from a customer agent $a_i \in \mathcal{A}$ to a SP agent $a_j \in \mathcal{A}_{SP}$ with $x^t(s_{ij}) = 0 \Leftrightarrow \psi_{ij}^t := \psi(o_{ij}^t, b_{ij}^t) = 0$ and $x^t(s_{ij}) = 1 \Leftrightarrow \psi_{ij}^t := \psi(o_{ij}^t, b_{ij}^t) > 0$.

2.1.3.2.3.11 Service dependency $\varphi : \mathcal{S} \to \mathscr{P}(\mathcal{S})$ is a function that returns the set of services that are required to provide a service s_{ij} along the service network paths (including s_{ij} itself), where $\mathscr{P}(\mathcal{S})$ denotes the powerset of the set $\mathcal{S}$. That is, φ defines multi-tier service production dependencies. Furthermore, this work defines the binary function $z^t : \mathcal{S} \to \{0, 1\}$, which returns 1 if and only if contracts have been established for all services which are required to provide the service s_{ij} in period t, and 0 otherwise. Thus, z^t defines if multi-tier dependencies along the SN paths have been considered in the allocation in period t:

$$z^t(s_{ij}) = \begin{cases} 1 & \text{if } \forall s_{k\ell} \in \varphi(s_{ij}) : x^t(s_{k\ell}) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Obviously, $z^t(s_{ij}) \leq x^t(s_{ij})\ \forall s_{ij} \in \mathcal{S}$, as $s_{ij} \in \varphi(s_{ij})$. In order to simplify the following notion, this research sets $z_{ij}^t := z^t(s_{ij})$, and $x_{ij}^t := x^t(s_{ij})$.

2.1.3.2.3.12 Penalty The function $\rho_i^t : \mathcal{S} \to \mathbb{R}$ denotes the penalty payments received and paid by agent a_i in period t if a SP agent fails to provide a contracted service to a_i or a_i fails to provide a contracted service to a customer, respectively.

With penalty factor $\gamma \in \mathbb{R}^+$, $\rho_j^t(s_{ij}) = (1 - z_{ij}^t)\,\gamma\,\psi_{ij}$ denotes the outgoing penalty payment and $\rho_j^t(s_{jk}) = (1 - z_{jk}^t)\,\gamma\,\psi_{jk}$ the incoming penalty payment in case of overcommitment for the allocation in period $t \in \mathcal{T}$ for agent $a_j \in \mathcal{A}$. Since customer agents only receive but never pay penalties themselves, it yields $\rho_j^t(s_{ij}) = 0\,\forall a_j \in \mathcal{A}_C, t \in \mathcal{T}$.

2.1.3.2.3.13 Customer utility function The utility function of customer agent $a_i \in \mathcal{A}_C$ is determined by a_i's valuation (v_{ij}) for services $s_{ij} \in \mathcal{S}$ that are provided in $t \in \mathcal{T}$ (i.e., $z_{ij}^t = 1$), less the payments ψ_{ij}^t for service provision, plus penalty payments $\rho_i^t(s_{ij})$ for allocated services (i.e., $x_{ij}^t = 1$). Thus, the utility function of customer $a_i \in \mathcal{A}_C$ is given by:

$$U_i^t(\mathcal{S}^t) = \sum_{a_j \in \mathcal{A}_{SP}} z_{ij}^t(v_{ij} - \psi_{ij}^t) + x_{ij}^t \rho_i^t(s_{ij}). \tag{2.2}$$

Customer agents do not receive any penalty payments from their SP agents

if these provide the service in accordance to the established contracts (i.e., $z_{ij}^t = 1 \Leftrightarrow \rho_i^t(s_{ij}) = 0$). Contrarily, if a SP agent fails to provide its contracted service (i.e., $x_{ij}^t = 1$ and $z_{ij}^t = 0$), the SP agent has to pay a penalty fee to the customer agent (i.e., $\rho_i^t(s_{ij}) > 0$). This state is denoted as overcommitment by agent a_j. Formally, $x_{ij}^t = 1 \wedge z_{ij}^t = 0 \Leftrightarrow \rho_i^t(s_{ij}) > 0$.

2.1.3.2.3.14 Service provider utility function The utility function of SP agent $a_j \in \mathcal{A}_{SP}$ is determined by the payments it receives for the provision of services $s_{ij} \in \mathcal{S}$ (ψ_{ij}^t with $z_{ij}^t = 1$), less the cost for allocated resources for the provision of s_{ij} (c_{ij} with $x_{ij}^t = 1$) and penalties to be payed ($\rho_j^t(s_{ij})$).

In addition, agent a_j can act as a customer in the next tier. Consequently, the utility function has to take into account payments (ψ_{jk}^t) to SP agents $a_k \in \mathcal{A}_{SP}$ in the next tier for provided services $s_{jk} \in \mathcal{S}$ and penalty payments received ($\rho_j^t(s_{jk})$).

Then the utility function of SP agents $a_j \in \mathcal{A}_{SP}$ is given by:

$$\begin{aligned} U_j^t(\mathcal{S}^t) = &\sum_{a_i \in \mathcal{A}} z_{ij}^t \psi_{ij}^t - x_{ij}^t (c_{ij} + \rho_j^t(s_{ij})) \\ &- \sum_{a_k \in \mathcal{A}_{SP}} \Big(z_{jk}^t \psi_{jk}^t - x_{jk}^t \rho_j^t(s_{jk}) \Big). \end{aligned} \tag{2.3}$$

2.1.3.2.3.15 Utilitarian social welfare The social welfare is given by the sum of (2.2) and (2.3) over all participating agents in $\mathcal{A}$. Thus,

$$\begin{aligned} sw(\mathcal{S}^t) &= \sum_{a_i \in \mathcal{A}_C} U_i^t(\mathcal{S}^t) + \sum_{a_j \in \mathcal{A}_{SP}} U_j^t(\mathcal{S}^t) \\ &= \sum_{a_i \in \mathcal{A}_C} \sum_{a_j \in \mathcal{A}_{SP}} z_{ij}^t (v_{ij} - \psi_{ij}^t) + x_{ij}^t (1 - z_{ij}^t)\, \gamma\, \psi_{ij}^t \\ &\quad + \sum_{a_j \in \mathcal{A}_{SP}} \sum_{a_\ell \in \mathcal{A}} z_{\ell j}^t \psi_{\ell j}^t - x_{\ell j}^t \Big(c_{\ell j} + (1 - z_{\ell j}^t)\, \gamma\, \psi_{\ell j}^t \Big) \\ &\quad - \sum_{a_j \in \mathcal{A}_{SP}} \sum_{a_k \in \mathcal{A}_{SP}} z_{jk}^t\, \psi_{jk}^t - x_{jk}^t (1 - z_{jk}^t)\, \gamma\, \psi_{jk}^t. \end{aligned} \tag{2.4}$$

Since $\mathcal{A} = \mathcal{A}_C \cup \mathcal{A}_{SP}$, the second inner sum in (2.4) can be split up into two sums running over $\mathcal{A}_C$ and $\mathcal{A}_{SP}$. Then indices can be set such that $\ell = i$ in the resulting sum over $\mathcal{A}_C$, as it iterates over the same set of agents. In the same way, let $j = k$ and $\ell = j$ in the resulting sum over $\mathcal{A}_{SP}$. Finally, the cost for providing services to agents are combined from $\mathcal{A}_C$ and $\mathcal{A}_{SP}$ respectively. Thus,

$$\begin{aligned} sw(\mathcal{S}^t) &= \sum_{a_i \in \mathcal{A}_C} \sum_{a_j \in \mathcal{A}_{SP}} z_{ij}^t v_{ij} - x_{ij}^t c_{ij} \\ &\quad - \sum_{a_j \in \mathcal{A}_{SP}} \sum_{a_k \in \mathcal{A}_{SP}} x_{jk}^t c_{jk} \\ &= \sum_{a_j \in \mathcal{A}_{SP}} \Big(\sum_{a_i \in \mathcal{A}_C} z_{ij}^t v_{ij} - \sum_{a_\ell \in \mathcal{A}} x_{\ell j}^t c_{\ell j} \Big). \end{aligned} \tag{2.5}$$

In social welfare, payments and penalties sum up to zero. This result is intuitive, since payments constitute redistribution of wealth between the agents, but do not generate a surplus from a welfare perspective.

2.2 Problem analysis

In this section, the addressed problem is formally described and certain properties of solutions to the problem are analyzed. It is proven that overcommitments cannot exist in optimal solutions and that finding optimal solutions is $\mathcal{NP}$-complete. Further, it is shown that the utilitarian social welfare optimal allocation in multi-tier service networks is supported by a strategy profile of a Nash equilibrium and that equilibria always exist in multi-tier service network allocation games. The requirements for the allocation protocol to solve the problem addressed are described by means of a structured informal document with all features of the protocol (Huget & Koning 2003, pp. 180–182). Finally, a definition of the requirements is provided.

2.2.1 Utilitarian social welfare maximization problem

Utilitarian social welfare (section 2.1.2.4) is most commonly used in the multiagent systems research area to assess the system performance in terms of the maximal average profit of negotiating agents in electronic commerce transactions (Chevaleyre et al. 2006, p. 16). Therefore, the allocation problem addressed in this thesis can be formulated as a maximization problem of this metric.

Let $\mathcal{S}^* = \bigcup_{t \in \mathcal{T}} \mathcal{S}^t$. Then the utilitarian social welfare maximization problem consists of determining the optimal set of services:

$$
\begin{aligned}
sw(\mathcal{S}^*) = & \max_{\mathcal{S}^* \in \mathcal{S}} \sum_{t \in \mathcal{T}} \sum_{a_j \in \mathcal{A}_{SP}} \Big(\sum_{a_i \in \mathcal{A}_C} z_{ij}^t v_{ij} - \sum_{a_\ell \in \mathcal{A}} x_{\ell j}^t c_{\ell j} \Big) \\
& \text{s.t.} \quad \forall a_j \in \mathcal{A}_{SP} : \sum_{a_i \in \mathcal{A}_C} w_{ij} x_{ij}^t \leq W_j.
\end{aligned} \tag{2.6}
$$

The set of services $\mathcal{S}^* = \bigcup_{t \in \mathcal{T}} \mathcal{S}^t$ maximizes the utilitarian social welfare, i.e., the sum of customer agents' valuations (v_{ij}) for services $s_{ij} \in \mathcal{S}$ that are provided in $t \in \mathcal{T}$ ($z_{ij}^t = 1$), less the SP agents' costs for allocated resources for the provision of s_{ij} (c_{ij} with $x_{ij}^t = 1$). The side condition ensures, that resources from SP agents are not allocated beyond their resource capacities. That is, the resources allocated for service provision (w_{ij} with $x_{ij}^t = 1$) must not exceed the total resource capacities $W_j \, \forall a_j \in \mathcal{A}_{SP}$.

Next, it is proven that optimal solutions to the social welfare maximization problem do not contain overcommitments (section 2.1.3.2.3) in theorem 1 as follows. If there is overcommitment in an utilitarian social welfare optimal allocation, there would be a service $s_{ij} \in S$ for which $z^t(s_{ij}) = 0$ and $x^t(s_{ij}) = 1$. It yields $c_j(s_{ij}) > 0$ for all SP agents and services. Therefore, the overcommitment would cause negative welfare, due to the cost of the allocated resources; it would not lead to positive welfare by the valuation for s_{ij}, since s_{ij} is not provided ($z^t(s_{ij}) = 0$). Hence, overcommitments cannot exist in utilitarian social welfare optimal solutions.

Theorem 1. *If $\mathcal{S}^* = \bigcup_{t \in \mathcal{T}} \mathcal{S}^t$ is the set of contracted services of an utilitarian social welfare optimal allocation, then $z^t(s_{ij}) = x^t(s_{ij}) \, \forall s_{ij} \in \mathcal{S}^t, \forall t \in \mathcal{T}$.*

Proof. By contradiction. Let $\mathcal{S}^*$ be the set of contracted services of an utilitarian social welfare optimal allocation such that $\exists t \in \mathcal{T}, s_{ij} \in \mathcal{S}^t \subseteq \mathcal{S}^* : z^t(s_{ij}) \neq x^t(s_{ij})$. As $z^t(s_{ij}) \leq x^t(s_{ij}) \; \forall s_{ij} \in \mathcal{S}^t$, it yields $z^t(s_{ij}) = 0$ and $x^t(s_{ij}) = 1$. Let $\mathcal{S}^{*'} := \mathcal{S}^* \backslash \{s_{ij}\}$. As $\mathcal{S}^*$ is an optimal allocation, $sw(\mathcal{S}^*) \geq sw(\mathcal{S}^{*'}) \Leftrightarrow sw(\mathcal{S}^{*'}) - x^t(s_{ij}) c_j(s_{ij}) \geq sw(\mathcal{S}^{*'})$. As $x^t(s_{ij}) = 1$ and $c_j(s_{ij}) > 0$, the contradiction follows. □

2.2.1.1 Computational problem complexity

The computational complexity (section 2.1.2.5.6) of the utilitarian social welfare optimization problem is analyzed in this section. To prove that the problem addressed is $\mathcal{NP}$-complete, it has to be shown that (i) the problem is in $\mathcal{NP}$ and (ii) any instance of a problem that is known to be $\mathcal{NP}$-complete can be transformed to the problem addressed in polynomial time. For $\mathcal{NP}$ membership, a given solution has to be verifiable in polynomial time (verifiability); i.e.,

a non-deterministic algorithm which 'guesses' solutions can 'solve' the problem in polynomial time (Garey & Johnson 1979).

A transformation (reduction) *from* the 0-1 knapsack problem *to* the problem addressed in this thesis is provided. The 0-1 knapsack problem consists of the determination of a subset of a given set of items, with defined valuation and weight each, to be put in a knapsack with limited weight capacity, such that the valuation of the items in the knapsack is maximized. The specificity of the *0-1* knapsack problem is that there is only one of each item. The transformation is defined as follows:

(i) Overcommitments are excluded, (ii) only one time period is considered, (iii) only one SP agent is considered, (iv) the function that determines which items are put in the knapsack is transformed to the allocation function x_{ij}^1, (v) the valuations of the items are transformed to cost less valuation of services, (vi) the weights of the items are transformed to the capacity requirements of the services, and (vii) the knapsack capacity is transformed to the resource capacity of the single SP agent.

Since this reduction can be done in polynomial time, the problem is shown to be $\mathcal{NP}$-hard. Together with $\mathcal{NP}$ membership, this proves that the problem addressed is $\mathcal{NP}$-complete.

Theorem 2. *The computation problem for the socially optimal allocation is $\mathcal{NP}$-complete.*

Proof. $\mathcal{NP}$ membership is trivial, since for any given solution to the allocation problem, the calculation of the resulting social welfare (2.5) and the verification of the constraints that the sum of the capacity requirements $w_{ij}x_{ij}$ does not exceed the given capacity W_j for each SP (2.6) can obviously be done in polynomial time. For hardness, this research defines a reduction from the 0-1 knapsack problem which is known to be $\mathcal{NP}$-complete (Karp 1972). The 0-1 knapsack problem is defined as follows. Let U be a finite set and let $u \in U$. Further, $g(u)$ denotes the weight and $b(u)$ the valuation of element u, and K is a positive integer. Now maximize $\sum_{u\in U} a(u)b(u)$ s.t. $\sum_{u\in U} a(u)g(u) \leq K$, where $a(u)$ denotes a 0-1 assignment to each $u \in U$ (Garey & Johnson 1979). Any instance of the 0-1 knapsack problem can be reduced to our optimization problem as follows. By theorem 1, $z_{ij}^t = x_{ij}^t$ can be assumed. In addition, let $|\mathcal{T}| = 1$, $|\mathcal{A}_{SP}| = 1$, $\mathcal{A}_C = U$, $x_{ij}^1 = a(u)$, $v_{ij} - c_{ij} = b(u)$, $w_{ij} = g(u)\,\forall u \in U$, and $W_j = K$ with $a_j \in \mathcal{A}_{SP}$. Obviously, this reduction can be done in polynomial time. Hence, each SP agent's optimization problem and therewith the utilitarian social welfare maximization problem is $\mathcal{NP}$-hard. □

2.2.1.2 Equilibria

This section applies the concept of Nash equilibrium (NE) to analyze the network allocation game (section 2.1.2.3.4). Informally, a strategy profile is a Nash equilibrium if no player has incentive to deviate from his strategy given that the other participating players do not deviate (Nash 1950, Rasmusen 1989).

It is shown that the empty allocation constitutes a NE if all bids exceed all customer valuations and the costomers' offers are 0 in proposition 1. If a SP agent deviates, it cannot gain positive utility, since the willingness to pay for all customers is 0. Thus, there is no incentive for SP agents to deviate from their strategy. For customer agents, deviating is not beneficial, since the bids of all SP agents exceed the valuations of all customer agents. Therefore, a NE always exists in multi-tier service network allocation games.

Proposition 1. *A Nash equilibrium always exists in multi-tier service network allocation games.*

Proof. Let $o_{ij}^t = 0$ and $b_{ij}^t = N \ \forall a_i, a_j \in \mathcal{A}, \ \forall t \in \mathcal{T}$ where $N > v_{ij}$ is a large number. As N exceeds the customer agents' valuation for all services without loss of generality, this results in an empty allocation, i.e., no service is contracted. Further, SP agents have cost ($c_{ij} > 0$) for the provision of contracted services. Thus, no agent can gain a positive utility by deviating from the characterized strategy profile. That is, the empty allocation constitutes a NE. □

In proposition 2, it is proven that utilitarian social welfare optimal allocations are supported by a strategy profile of a NE in multi-tier service networks as follows. If a service is not in the optimal set of services, customers' offers are 0 and all bids exceed all customer valuations for these services. If a service is part of the optimal set of services, bids are equal to offers and the valuations exceed the payments for these services, and SP agents bid their marginal cost. If a SP agent deviates, it cannot gain positive utility, since the willingness to pay for all customers equals the marginal cost for service production. Therefore, there is no incentive for SP agents to deviate from their strategy. For customer agents, deviating is not beneficial, since the bids of all SP agents exceed the valuations of all customer agents for services that are not in the set of optimal services. For services that are part of the optimal set of services, it is not beneficial for the customer agent to deviate, since for lower offers, it will not receive the allocation and cannot gain utility, due to the valuation for these services. Higher offers are useless, since customer agents would pay more than

necessary for the services requested. Therefore, the utilitarian social welfare optimal allocation in multi-tier service networks is a Nash equilibrium.

Proposition 2. *The utilitarian social welfare optimal allocation in multi-tier service networks is supported by a strategy profile of a Nash equilibrium.*

Proof. Let $\mathcal{S}^*$ be an utilitarian social welfare optimal allocation. The agents' strategy profiles are constructed as follow. If $s_{ij} \notin \mathcal{S}^* \Rightarrow o_{ij}^t = 0$ and $b_{ij}^t = N$, where N is a large number; $s_{ij} \in \mathcal{S}^* \Rightarrow \exists t \in \mathcal{T} : o_{ij}^t = b_{ij}^t$, $v_{ij} \geq \psi_{ij}^t$, and $\psi_{ij}^t = c_{ij} + \sum_{s_{jk} \in \varphi(s_{ij})} \psi_{jk}^t \ \forall a_i, a_j \in \mathcal{A}$. Thus, each SP agent has 0 utility and cannot gain a positive utility by deviating from this strategy, as their bids correspond to the marginal cost for service provision. The customer agents cannot increase their utility by choosing a different strategy, as N exceeds the customer agents' valuation for $s_{ij} \notin \mathcal{S}^*$ or $v_{ij} \geq \psi_{ij}^t$ otherwise. This strategy profile can always be constructed and constitutes a NE. □

2.2.2 Requirements description

For the production of the requested services in SNs it is possible for SPs to (i) utilize resources from the own inventory or (ii) buy services from a third party (subcontracting). The latter is done if (i) the own capacity is not sufficient, (ii) the own resources do not have the required functionality, or (iii) if the utilization of the own capacity is economically not favorable because of the cost function (e.g., step cost). The subcontracting can also comprise parts of the service. This means, that a service provided to the customer can be an arbitrary composition of own and procured services.

SPs have to consider economically relevant values to determine the concrete resources for service provision. For services in SNs, this implies that the individual requirements of the customer determine the requirements of SLAs that have to be established in upstream SN tiers. Therefore, the protocol must allow for subcontracting activities by SPs and for establishing all (sub-)agreements for a concrete customer request in a coordinated manner: the protocol has to avoid overcommitments. That is, it must prevent agreements with customers without establishing necessary sub-agreements with other agents.

For example, the service requested by the customer can be a data mining service which is provided in tier $\lambda = 0$. The customer request includes a definition of the service along with respective parameters (e.g., throughput). The SPs can subcontract parts of the service to, respectively procure appropriate subservices from, their suppliers (e.g., storage and analysis services) in tier $\lambda = 1$. The protocol has to ensure that agreements will either be established with the

customers *and* with respective suppliers or no agreements will be established at all.

As a next step towards a protocol specification, this research performs the analysis stage in accordance to the interaction protocol engineering approach. On the basis of the discussion and SN model, a structured informal document with all features of the protocol can be provided (Huget & Koning 2003, pp. 180–182). This informal description is shown in table 2.4 for the proposed protocol.

The formal problem analysis and informal description of the protocol are utilized to derive requirements for the protocol to be specified. Next, these requirements are discussed in detail. They comprise (i) distributed allocation, (ii) service dependencies, and allocation mechanism properties: (iii) allocative efficiency, (iv) incentive compatibility, (v) individual rationality, (vi) budget balance, and (vii) allocation complexity.

Distributed allocation refers to centralized vs. distributed allocation. There are, however, existing approaches which distribute parts of the allocation problem, but still require a central coordinating entity. Service dependencies are twofold – they can be allowed by an approach without guaranteeing their consideration in every allocation outcome. In contrast, the stronger interpretation of this criterion requires every valid allocation outcome to guarantee the consideration of service dependencies.

The mechanism properties of an allocation procedure are analyzed with regard to the properties defined in section 2.1.2.5.

Table 2.4: Informal protocol description.

Name	Multi-tier contract net protocol
Keywords	multi-tier resource allocation, service networks, swaps
Agents' role	customer, service provider (SP)
Initiator	customer
Prerequisite	The agents must know how to send messages to other agents.
Function	A customer requests to buy a service which has to be provided at a certain service level (quality of service). A SP offers a bid on the request. The SP may procure (sub-)services from other SPs (subcontracting, outsourcing) in multi-tier request processing if (i) the SP's capacity is insufficient, (ii) the SP's resources do not provide the required functionality, or (iii) if the utilization of the SP's capacities is not economically favorable. This can also comprise parts of the service; i.e., a composition of own and procured services. The protocol enables to establish all agreements along the SN in a coordinated manner.
Behavior	The protocol can be decomposed into three phases: (i) In the collect proposal phase, the customer's request is transferred to potential SPs in service network tier λ. The SPs process the request and may decide to establish sub-agreements for parts of the request in tier $\lambda + 1$ of the service network; i.e., they transfer own requests to other SPs in tier $\lambda + 1$. SPs' subcontracting activities can be recursively extended to an arbitrary number of tiers. Before a SP in tier λ answers the final customer's request with a binding bid, it evaluates the bids received from SPs in tier $\lambda + 1$. Thus, the SP can make a bid to the customer based on the results of multi-tier bids. (ii) In the acceptance notification phase, the SPs are informed about the acceptance of the bids they have provided. The SPs in tier λ subsequently inform the SPs in tier $\lambda + 1$ about the acceptance of dependent bids. The recursion is executed to the number of tiers on which bids were submitted. (iii) The execution phase comprises the actual service delivery. The result of the execution (e.g., service level) is reported to the customer. SPs in tier λ report the state of the provision of the services to their customers which may constitute SPs in tier $\lambda - 1$. The recursion is executed to the number of tiers on which agreements have been established.
Constraints	All agents must be authenticated.
Termination	(i) All SPs have provided the services as contracted, (ii) the actual delivery of one or more services has failed, (iii) no agreement has been established, or (iv) a timeout has occurred during resource allocation or execution.

2.2.2.1 Distributed allocation

It is required that the distributed nature of SNs is considered. Therefore, the protocol must not assume that central coordinating entities (such as central auctioneers) exist to select and enforce allocation outcomes.

Requirement 1 (distributed allocation)**.** *The protocol must allow for distributed resource allocation, i.e., it must not require central coordinating entities.*

The distribution of the allocation comprises the existence of a central agent that selects and enforces allocation outcomes. In a completely centralized auction-based setting, there is a single auctioneer that computes the allocation outcomes. This requires the existence of an appropriate agent that can assume this role. In addition, this agent has to be able to communicate with every agent relevant for the allocation.

In distributed winner determination scenarios, the central agent does not calculate the allocation outcomes itself, but distributes the computational task to other agents. This type of distribution is concerned with distribution of the allocation computational load only. Thus, it neither reduces the complexity of winner determination, nor does it make a central auctioneer agent obsolete. In this thesis, an allocation procedure is denoted as distributed if and only if there is no single agent that selects and enforces allocation outcomes along the complete SN, since the assumption of a central entity contradicts the distributed nature of SNs.

2.2.2.2 Service dependencies

As a central focus of this research, the protocol must consider service dependencies over multiple tiers of SNs. It has been shown that overcommitments cannot occur in socially optimal allocations. Therefore, this research formulates a respective requirement for the protocol proposed.

Requirement 2 (service dependencies)**.** *The protocol must avoid overcommitments.*

An allocation procedure can guarantee consideration of service dependencies in SNs if and only if it addresses supply and demand side of service producing agents. This means that it is required to respect *both* input and output services of every SP agent *before* any binding allocation is made. That is, during the execution of the allocation procedure. If the service dependencies are not considered, the provision of services (i.e., fulfillment of agreements) may be

unaccomplishable due to missing services which are required for the provision (overcommitment).

Respecting service dependencies in the allocation procedure is not a binary criterion. For example, an approach can minimize the expected probability of overcommitments (i.e., existence of unconsidered service dependencies in the allocation outcome) by SP agents while it does not guarantee to avoid overcommitments. Thus, the service dependency requirement is divided into two sub-criteria to assess how pertinent allocation approaches deal with service dependencies.

The weaker form of this requirement is denoted as consideration of service dependencies. The stronger form, which guarantees to respect *all* service dependencies in the allocation outcome, is denoted as avoidance of overcommitments.

2.2.2.3 Allocative efficiency

Next, properties of the allocation mechanism are considered: allocative efficiency, incentive compatibility, individual rationality, budget balance, and allocation complexity. Since the allocation complexity in SNs is generally $\mathcal{NP}$-complete, the utilitarian social welfare optimization problem cannot be solved in polynomial time for any problem instance. In addition, it is impossible to design a mechanism that is allocatively efficient, budget balanced, and incentive compatible, at least in settings with quasi-linear preferences (Green & Laffont 1977, Walker 1980, Myerson & Satterthwaite 1983, Hurwicz & Walker 1990). Therefore, the proposed solution must approximate the utilitarian socially optimal allocation.

Requirement 3 (allocative efficiency)**.** *The allocation protocol must constitute a heuristic for the utilitarian social welfare maximization problem.*

The allocative efficiency of a mechanism refers to the mechanism's property of maximizing a preference aggregation (e.g. social welfare) of the agents. In this thesis, the notion of utilitarian social welfare is utilized. Thus, a mechanism is allocatively efficient if and only if it maximizes the utilitarian social welfare, i.e., the sum of utilities of the agents (sections 2.1.2.4 and 2.1.2.5.2).

2.2.2.4 Incentive compatibility

A mechanism is incentive compatible if it is rational for the agents to report their true preferences (types); the agents cannot gain utility by reporting untrue preferences (e.g., manipulating bids) (section 2.1.2.5.3).

Requirement 4 (incentive compatibility)**.** *The allocation protocol must be incentive compatible.*

2.2.2.5 Individual rationality

If no agent is worse-off by participating in the allocation, a mechanism is individually rational. That is, no agent can gain utility by not participating in the allocation game (section 2.1.2.5.4).

Requirement 5 (individual rationality)**.** *The allocation protocol must be individually rational.*

2.2.2.6 Budget balance

An allocation mechanism is budget balanced if there are no net payments by the mechanism to the participating agents, i.e., the single payments of the agents sum up to zero (section 2.1.2.5.5).

Requirement 6 (budget balance)**.** *The allocation protocol must be budget balanced.*

2.2.2.7 Allocation complexity

The complexity of an allocation procedure comprises computational and communication complexity (section 2.1.2.5.6). The allocation complexity in SNs is generally $\mathcal{NP}$-complete. The utilitarian social welfare optimization problem cannot be solved in polynomial time for any problem instance. Therefore, the proposed solution must approximate the utilitarian socially optimal allocation and must guarantee to terminate in polynomial time.

Computational complexity theory is concerned with the question of how many computational resources are required to solve a given problem. It provides a classification of problems of different complexity (Garey & Johnson 1979, Papadimitriou 1994). This thesis assigns a complexity class (e.g., $\mathcal{NP}$) of the problem to be solved, following the allocation procedures proposed in related research.

Requirement 7 (allocation complexity)**.** *The allocation protocol must guarantee to terminate in polynomial time for any problem instance.*

Communication complexity has four elements: (i) the number of required agreements (or deals) to arrive at an optimal allocation outcome, (ii) the number of dialogue moves required to establish an agreement, (iii) the expressiveness of the communication language required, and (iv) the complexity of individual

agents' reasoning tasks, deciding on dialogue moves (Endriss & Maudet 2005, p. 122).

In the following review of current solutions to the allocation problem, the latter two elements of communication complexity are neglected, due to the following reasons. First, the required communication language is widely application domain-dependent. This aspect of communication complexity is concerned with requirements of concrete *solution instances*, rather than with the communication complexity of *conceptual solutions*. Second, the individual agents' reasoning complexity is essentially a question of computational complexity.

2.3 Allocation approaches for multi-tier service networks

Resource allocation has been subject of comprehensive research in multiagent systems (see (Chevaleyre et al. 2006) for a survey); relevant approaches can be divided into two different groups.

The first group, which mostly does not deal with concrete negotiation mechanisms, focuses on principles of specific sub-topics. This includes approaches for distributed allocation, leveled commitments, and socially optimal allocation outcomes.

The second group consists of more concrete approaches which are found in the areas of interdependent negotiations, supply chain formation, and service network formation. Interdependencies of multiple negotiations for supply *and* demand constitute a class of relevant research. These approaches consider dependencies between inputs and outputs. In contrast, another class of related work is concerned with multiple supply (procurement) negotiations. It addresses the problem of procuring a desired product in the right amount and at the right price. That is, decision problems are addressed to avoid agreements for more products than intended (e.g., winning two auctions for one unit of a product each if just one unit is required). Among the approaches for supply chain formation, most consider coordination via a central entity, though there are also approaches which consider decentralized coordination. The most relevant class of service network formation research considers auctions as allocation mechanisms.

The requirements defined in section 2.2.2 must be fulfilled by any allocation approach to solve the research problem. Therefore, the following literature review assesses if and how pertinent allocation approaches meet one or more of these requirements.

2.3.1 Distributed allocation

Parkes & Shneidman (2004) present guidelines for the development of distributed allocation mechanism implementations based on the Vickrey-Clarke-Groves (Vickrey 1961) mechanism. The aim of the approach is to distribute as much computational load as possible onto network nodes and thus help to determine a suitable allocation result. However, the proposed allocation procedure requires a 'center' entity which communicates with network nodes through a trusted channel and hence selects and enforces allocation outcomes. Thus, this approach cannot be applied to SNs without the existence of central coordinating entities, i.e., it does not allow for distributed allocation in terms of the requirement. Since Parkes & Shneidman (2004) do not propose a concrete algorithm or negotiation framework, neither the service dependency nor the allocation mechanism properties requirements are applicable.

In the field of multiagent resource allocation, Bo and Lesser (Bo & Lesser 2010) investigate contract-based resource allocation across computational networks – a set of selfish agents route traffic for individual users. Before user agents can route traffic through node entities, contracts between user agents and the participating nodes need to be established. The negotiation approach proposed by Bo and Lesser is of distributed nature, i.e., agents act on behalf of themselves, and the corresponding resource allocation emerges from sequences of distributed negotiations.

In addition to mutual contracting, nodes are allowed to decommit from existing contracts at a penalty cost. The authors investigate the relationship between stability and optimality of the network resource allocation game and consider system dynamics during agent contracting. However, user agents are required to establish a separate contract with each node on the path. Thus, service dependencies, in which each node only negotiates contracts with nodes in adjacent tiers, are not considered. That is, the approach does not consider service dependencies.

The allocation problem addressed by Bo & Lesser (2010) is $\mathcal{NP}$-complete. However, the proposed distributed approach constitutes a polynomial time heuristic of the social welfare maximization problem. The agents exchange a finite set of messages, and network node agents decide if they are willing to accept user agents' bids. As network node agents act rational, they accept user agents' bids if and only if they do not exceed the network node agents' costs for routing the user agents' traffic. Thus, the network node agents' decision problem can be solved in polynomial time. Further, routing paths and node agents' marginal cost for routing traffic are known to the user agents, as

complete information is assumed. The user agents' decision processes consist of price comparisons for alternative paths and potential decommitments from obsolete node contracts. This reasoning can be done in polynomial time.

The proposed mechanism is not individually rational, since costs can occur without valuation for user agents. In addition, the approach does not incentivize truthful bidding, but is budget balanced. Regarding the communication cost, three messages are required to conclude a contract between a user and a node agent.

2.3.2 Leveled commitments

Sandholm & Lesser (2001) propose leveled commitment contracts which are not binding, but can be canceled by an agent by paying a pre-defined penalty to the other party (Sandholm & Lesser 1995, Sandholm & Lesser 2001). In (Sandholm & Lesser 1995), leveled commitments and an extension of the Contract net protocol (Smith 1980) are discussed. In (Sandholm & Lesser 2001), further details of leveled commitments are given. The approach addresses the traditional assumption of binding contracts between agents. In contrast, in leveled commitment contracts, the level of commitment is defined by breach (i.e., decommitment) penalties. An agent can dissolve an agreement (or contract) by paying a pre-defined penalty to the other agents involved in the agreement.

Sandholm & Lesser (2001) compare the proposed leveled commitment contracts to full commitment contracts in a setting where agents have probabilistic information about a contract's value. Thus, the expected utilities are analyzed for outside offers for contractor (the agent that pays the contract price) and contractee (the agent that performs the contracted action). The agents decide on decommitments when they know their outside offer, but the other agent involved in the agreement does not know any outside offer of other agents.

Four distinct decommitment mechanisms are analytically compared; they differ in the order of revealing the decommitment decision (sequentially or in parallel) and whether the penalties have to be paid if both agents decommit. In addition, games with future uncertainty for one and both agents are analyzed.

Leveled commitments can be applied to both centralized and distributed winner determination. The approach cannot consider service dependencies, but can mitigate the consequences of overcommitments, as the decommitment penalty would normally be smaller than the penalty for not being able to fulfill a binding agreement. However, leveled commitment contracts do not avoid overcommitments. They may even increase the probability of overcommitments, as upstream SPs can decommit from agreements which are required for the

fulfillment of interdependent agreements to customers.

Since the approach performs a game-theoretic analysis of decommitment games, but does not propose a concrete allocation protocol or mechanism, mechanism properties cannot be assessed. For example, the communication complexity greatly depends on concrete applications and cannot be assessed in general.

2.3.3 Socially optimal allocations of resources

Endriss, Maudet, Sadri & Toni (2006) analyze multilateral deals (i.e., agreements) to exchange bundles of indivisible resources between agents. Changes in resource distribution are analyzed for different collective utility functions (i.e., social welfare orderings). The proposed negotiation framework is analyzed for allocations with and without side payments with regard to guaranteed optimal social welfare and pareto optimality. Subsequently, both settings are applied to agent societies with different concepts of social welfare, i.e., interpretations of the concept of social welfare which are different from utilitarian social welfare – the concepts most commonly used in the multiagent systems research area.

This work constitutes a *distributed allocation* approach and focuses on the convergence to optimal allocations for different notions of social welfare. Since the negotiation framework considers multilateral agreements, a single agreement can comprise any number of agents and resources. The particular notion of dependency of agreements is introduced, though this notion of 'independently decomposable deals' refers to the agents involved, rather than to dependencies between allocations. That is, while it would be generally possible to consider service dependencies in multilateral deals, the proposed negotiation framework does not consider service dependencies in multi-tier SN.

The reason is that the framework would consider these dependencies in multilateral deals of all agents along the SN, since overcommitments would result in irrational agreements, though the framework assumes that *any* agent involved in the production of a service would participate in a single agreement. This assumption contradicts the multi-tier characteristic of the SN. The approach is individually rational and budget balanced by definition and guarantees allocative efficiency when side payments are possible, though it is not strategy-proof.

The communication complexity of the multilateral negotiation framework is analyzed for *one* of its four aspects in (Endriss & Maudet 2005), namely the required number of deals to arrive at an optimal allocation outcome. However, the presented upper and lower bounds for the needed number of deals refer to the abstract negotiation framework, rather than to a concrete negotiation pro-

tocol, i.e., these results are valid for any negotiation protocol enabling allocation under the provided assumptions.

2.3.4 Interdependent Supply Negotiations

In the area of interdependent procurements across multiple negotiations, Anthony & Jennings (2003) investigate the problem of bidding across multiple auctions to procure the best deal for the desired good. The approach addresses the decision process of agents bidding across auctions with different start and end times and with different auction protocols (English, Dutch, and Vickrey), i.e., agents' bidding decisions among concurrent heterogeneous auctions.

A genetic algorithm is applied to search for strategies in different environmental settings. These are experimentally evaluated in an electronic marketplace scenario with a group of random bidders; they simulate other bidders and bid on one single auction only. These random bidders' private valuations, starting bids, and bid increments are generated from a normal probability distribution.

Different bidding policies for the agent bidding on multiple auctions are analyzed. Here, the bidding agent does not submit further bids if it holds the highest bid in an English auction or has submitted a bid to a Dutch or Vickrey auction, to avoid purchasing of more than one product. The policy is defined by an aggregation function over different bidding functions for the remaining time left, the remaining number of auctions left, the desire to get a bargain, and the level of desperateness of a bidding agent. These four attributes are taken into consideration in distinct parametrized bidding functions, referred to as tactics. The functions' values are combined by the aggregation function with varying weights. In the experimental evaluation, different parameters for the tactics and different weights for the aggregation function are simulated.

Next, as the solution space for parameter values and weights is infinite, a genetic algorithm is applied to determine most successful bidding policies in predefined environments (e.g., short time, many auctions). The most successful bidding policies in terms of concrete parameter values and weights are then selected for different environments. Finally, these policies are experimentally evaluated against control bidding policies.

The approach does not propose an allocation protocol, but bidding policies in multiple procurement negotiations; it is addressing a single agent's decision problem when participating in multiple auctions. Thus, the requirements defined cannot be assessed. However, similar decision problems also exist in SNs and the approach can in principle be applied to service procurement in SNs.

The agents utilizing this bidding policy need to determine the environment type quickly and accurately. Further, probability distributions of the auctions are assumed to be static and known to the agents. With different strategies for the other auctions' participants, the bidding policy has to be evolved again, and more accurate probability predictions are necessary to select suitable target auctions.

Nguyen & Jennings (2005) analyze procurement activities for services in multiple bilateral negotiations, i.e., a customer agent is looking for a single SP agent from a number of multiple SP agents in its environment. Therefore, the customer agent concurrently negotiates bilaterally with the SP agents. The proposed negotiation framework is based on the concept of leveled commitment contracts (Sandholm & Lesser 2001).

The customer agents are assumed to know the probability distributions of successful negotiations for a set of strategies and the expected utility of successful negotiations. With this information, the customer agent calculates the probability that a SP agent is of a specific type. Then, it calculates the expected utility of applying different strategies to its proposal for this specific SP agent and selects the strategy which maximizes the expected utility. This is done until the customer agent has allocated strategies to all concurrent negotiations.

The negotiation framework considers 'intermediate deals' which can be decommited from at a dynamic penalty, starting at a percentage of the contract value and increasing towards the end of the negotiation. In an earlier version of the negotiation framework, customer agents were allowed to decommit at no penalty (Nguyen & Jennings 2004). Therefore, this version is heavily biased in favor of the customer agents. In (Nguyen & Jennings 2005), a customer agent decommits from a contract (and pays the penalty fee) if the expected utility of a new offer exceeds the utility of an established contract along with the penalty to be paid for the decommitment by a given threshold. The customer agent keeps bargaining this way until its private negotiation deadline is reached.

The authors also consider that a customer agent accepts more than one intermediate agreement to decommit from all, but the utility maximizing agreement at the private negotiation deadline. This avoids the risk of a single agreement that is revoked by the SP agent close to the negotiation deadline. However, if the SP agent does not decommit, the customer agent has to pay penalties for any additional intermediate agreement. The negotiation framework is empirically evaluated for three different types of SP agents.

Since the approach proposes bargaining between customer and SP agents, it constitutes a distributed allocation approach. It does not consider services

to be re-sold or to be utilized as input factors for value added services. Thus, the approach cannot consider service dependencies. However, the framework and findings can in principle be applied to negotiations on *distinct* tiers of SNs. The approach is budget balanced, but not individually rational or incentive compatible, and does not guarantee allocative efficiency. The computational complexity of the allocation is in $\mathcal{P}$, since the customer agent has to simply compare different offers by SPs. The communication complexity (in terms of messages to be sent) is potentially high, as multiple offers and counter-offers are sent and received by customer agents to *every* potential SP agent. In addition, decommitment messages are required to resolve unnecessary intermediate agreements.

Schillo, Kray & Fischer (2002) analyze resource allocation with the Contract net protocol (Smith 1980) and propose three strategies for the eager bidder problem. This problem results from a SP agent which submits bids to multiple reverse auctions for the utilization of its resources. On the one hand, the SP agents can submit single bids, potentially losing the auction without agreements for its resources. On the other hand, if SP agents submit bids for more than one auction, they may conclude more agreements than their resources can fulfill.

SP agents need to decide in which auctions they participate and how their commitments should be handled, with regard to limited resources. The authors compare three methods to address the eager bidder problem and compare them to the utilization of the conservative ad hoc solution, in which SP agents allocate resources just before sending a bid. The latter solution ensures resource availability. However, if the service provision requires only one SP agent and multiple customer and SP agents exist, this solution will lead to underutilization of SP agents' resources *and* not every customer agent will be provided with the service requested.

The first proposed solution to the problem addressed is based on the concept of leveled commitment contracts (Sandholm & Lesser 2001). This solution is only applicable for a limited number of concurrent customer agents, since penalty payments for decommitments must be taken into account. That is, no rational SP agent would establish contracts for which the decommitment penalties would exceed the expected utility of an outside offer.

The second solution approach is based on a protocol redesign. The authors propose the Contract net with confirmation protocol (CNCP) to postpone the commitment time as far as possible. Using the CNCP, the original bids of SP agents are non-binding. After selecting a potential SP agent from the bidders, the customer agents have to send an additional request to the bidder to con-

clude an agreement in accordance to the initial bid. SP agents make binding commitments only after the customer agents have explicitly requested them to do so. Since customer agents are expected to send these requests to the utility maximizing bidders only, this can significantly reduce the number of commitments made by SP agents.

As a third alternative to the conservative ad hoc solution, the authors propose a statistical risk analysis approach. Here, SP agents risk to enter commitments which they cannot fulfill as long as the estimated number of overcommitments is below a certain threshold. Therefore, the bidding agent needs to be able to determine the probability of getting a single bid accepted by a customer agent.

Schillo et al. (2002) characterize suitable environments for the different approaches proposed, since there is no generally superior solution to the eager bidder problem. The approach considers distributed auctioneers in terms of customer agents; it constitutes a distributed allocation approach. Although the addressed problem is relevant for the decision making of SP agents in SNs, the proposed solution does not include linking supply and demand side in interleaved interactions. Thus, it cannot consider service dependencies.

The approach does not propose a concrete mechanism. Therefore, the requirements for mechanism properties cannot be applied, e.g., the allocation complexity depends on the chosen solution alternative. While the computational complexity for winner determination is not altered using the proposed solution, the risk analysis requires additional computational resources. In contrast, leveled commitment contracts and the CNCP can increase the number of messages required to arrive at a final allocation.

2.3.5 Interdependent Supply and Demand Negotiations

Zhang, Lesser & Abdallah (2005) propose an approach using temporal ordering of negotiations, addressing supply chain formation problems. The authors investigate the problem of multi-linked negotiations, i.e., interconnected negotiations which influence each other. The relationships of related negotiations are classified into two categories. Two negotiations are directly linked if the characteristics of the subject of one negotiation (e.g., properties of resources, subtasks to be performed) directly affect the subject of the other negotiation.

In directly linked negotiations, the failure of one negotiation implies the infeasibility or unnecessity of the second. Indirectly linked negotiations compete for the utilization of common resources. That is, multi-linked negotiations denote multiple negotiations in which the negotiation over one issue influences

the negotiations over other issues.

The approach proposes a temporal ordering of multi-linked negotiations to be carried out either sequentially or in parallel. Thus, the first issue addressed is if these negotiations should be performed in parallel or in a sequence. If they are to be performed in sequence, the concrete order is to be determined to minimize conflicts.

Zhang et al. (2005) present a formal model for the problem of multi-linked negotiations along with a heuristic search algorithm to determine near-optimal solutions for the ordering and interdependencies. The relationships of directly-linked negotiations are described as a forest of rooted trees. A function defines for every non-leaf negotiation if the children are alternatives (OR) or complementary (AND).

The authors present a set of negotiation attributes which are domain dependent for the supply chain formation problem, such as start time and deadline. In addition, they identify a set of four attributes which are domain independent (duration, start time, deadline, and success probability). Zhang et al. (2005) argue that the problem cannot be reduced to project management or scheduling, as not only the negotiations are interdependent, but also the subjects and therewith attributes. Thus, two levels of interdependent entities exist in the problem of multi-linked negotiations.

The approach represent orderings in a directed acyclic graph that defines which negotiations can only start after others have been completed. Given a start time of the first negotiation, this graph constitutes the negotiation schedule. Finally, the authors evaluate the algorithm that constitute a heuristic for the ordering and attribute mapping problems in a set of multiagent simulation experiments.

The approach proposed by Zhang et al. (2005) is of distributed nature. It has a strong focus on the problem of directly-linked negotiations, which is essential for service dependencies. However, it does not avoid the establishment of unaccomplishable agreements and therewith penalty payments, as negotiations are considered as atomic blocks, and thus the interleaving of directly linked negotiations is not considered. That is, interdependencies in parallel negotiations are not considered – if they exist, the negotiations are to be performed in sequence; overcommitments cannot be avoided.

Since the approach does not present a concrete mechanism for allocation, but an ordering of interdependent allocations, the mechanism properties of an allocation utilizing this approach cannot be assessed.

Si, Edmond, ter Hofstede, Dumas & Chong (2005) propose a probabilis-

tic approach for composing interrelated negotiations, allowing compositions of alternative (one-or-the-other) and complementary (all-or-nothing) trading activities. The authors analyze a supply chain setting, where alternative trading activities denote alternatives for input procurements, while complementary trading activities can also comprise negotiations for inputs and respective outputs, i.e., dependencies over supply chain tiers.

However, the approach does not consider to interweave the complementary trading activities. The producer agents either acquire inputs first and produce outputs independently of customer agents' orders (buy-to-build), or they first negotiate with customers and subsequently acquire the inputs required to fulfill agreements with customer agents (build-to-order). The authors describe an agent using the buy-to-build high-level strategy. Before sending a bid to customer agents, the agent considers the global bidding history and calculate probabilities for closing prices. In addition, the average probability of an offer accepted by a customer agent is calculated. These probability distributions are then utilized to determine a producing agent's prices offered to customers. Thus, the approach allows the producing agents to maximize the expected utility based on the global negotiation history.

In (Si, Edmond, Dumas & Hofstede 2007), the research is generalized to any set of composite trading activities. The proposed model is capable of supporting the decision process of a single agent that (i) buys products in different auctions (i.e., English, Dutch, first-price sealed bid, and Vickrey auctions), (ii) sells products in reverse auctions, or (iii) buys and sells products by means of an alternating offers bargaining protocol. A special focus of this research are interdependencies, heterogeneous protocols, and temporal constraints of negotiations. Similar to their previous work, the authors calculate the expected utility of a set of trading activities based on the history of past negotiations.

Since the approach is applicable to various protocols, it can neither be considered centralized nor distributed. Similarly, the allocation mechanism cannot be assessed. Regarding service dependencies, the authors avoid invalid outcomes by assuming that the negotiations for the minimum number of required agreements minus one constitute 'secure' trading activities, i.e., that have received a binding offer by another agent, and are guaranteed to result in deals if executed. Any combination of successful and failed negotiations prior to the secure trading activity results in a set of agreements honoring the given constraints. However, it remains unclear how complementary trading activities can be combined such that the required number of 'secure' deals exists.

The nodes in the presented execution model represent composite or elemen-

tary negotiations. The authors present an analysis of trading activity states for different protocols, though the model is not capable of describing these single states, but a negotiation as a whole. As can be seen from the examples presented, the authors rely on information which is not represented in the model explicitly (i.e., the processing is done on negotiation state level while the model captures them as a whole only). This approach considers service dependencies with certain constraints on the dependencies, but cannot avoid overcommitments with the details provided.

2.3.6 Supply chain formation with auctions

Research on supply chain formation with combinatorial auctions assumes centralized winner determination along all tiers of the supply chain. A central entity, collecting all bids on input and output products in all tiers of the supply chain at a single point, is required.

Walsh, Wellman & Ygge (2000) propose a one-shot combinatorial auction protocol, and analyze bidding strategies for self-interested agents, i.e., the efficiency and producers' surplus are experimentally evaluated in five different networks of producer and consumer agents. The protocol's performance in terms of allocative efficiency is compared to a distributed, progressive auction protocol with non-strategic bidding.

The assumption of a central entity contradicts the distributed nature of supply chains and in particular SNs. Walsh et al. (2000) argue that, while acknowledging that it is infeasible to coordinate complete supply chains using a centralized approach in a single market, centralized coordination can be applied to sub-markets with strong dependencies between producer agents, since these sub-markets are only weakly dependent on the broader market of the complete network of producer and consumer agents (Walsh et al. 2000, p. 262).

The approach clearly utilizes centralized allocation. It explicitly considers interdependencies among inputs and outputs, i.e., it is suitable to avoid overcommitments. The allocation complexity of combinatorial auctions is generally $\mathcal{NP}$-complete, though there is a number of computationally manageable cases (i.e., solvable in polynomial time) (Rothkopf, Pekeč & Harstad 1998). The authors state, while not computationally manageable in general, many allocation problems can be solved using commercial mixed-integer-linear programming optimizers (Andersson, Tenhunen & Ygge 2000). The proposed protocol is budget balanced, though it is neither individually rational (producers can obtain inputs without the possibility to use it), nor incentive compatible.

Regarding communication complexity, one-shot combinatorial auctions es-

tablish optimal allocations with the minimal set of agreements. One dialogue move (message) is required by each agent to submit its combinatorial bid. The central auctioneer then sends a message to each agent, to inform it about the allocation outcome.

Walsh & Wellman (2003) present a decentralized protocol for supply chain formation based on auctions – SAMP-SB. Although this approach suggests different, decentralized auctions for every traded product, it requires mediators (auctioneers) that determine allocation outcomes. That is, a coordinating entity is required for each traded product, though the authors do not give details about this entity. Interdependencies between different auctions are not coordinated – producer agents place bids on output goods before they have received commitments on input goods (Walsh & Wellman 2003, p. 527). Thus, the SAMP-SB protocol cannot consider service dependencies.

To address this issue, and therewith individual rationality of the mechanism, the authors propose an extension to the SAMP-SB with decommitments – SAMP-SB-D. This protocol allows producer agents to decommit (at no penalty or fee) from agreements, for which production dependencies are not met. These decommitments are applied recursively to producer agents affected by other producer agents along the supply chain. Therefore, the SAMP-SB-D protocol makes auction bids non-binding; it avoids negative surplus for agents where production dependencies are not met, and is individually rational. There is the possibility for agents to manipulate bids and thus SAMP-SB-D is not incentive compatible. The protocol cannot enforce that only producer agents with unmet dependencies decommit from agreements (Walsh & Wellman 2003, p. 531).

The allocation complexity of the SAMP-SB(-D) protocols' auctions is in $\mathcal{P}$, since bids comprise non-combined (i.e., 'additive-OR') quantity-value pairs. These can be processed as if they where separate bids from different agents. Regarding communication complexity, the SAMP-SB protocol may need a number of bids that is exponential in network size (i.e., number of agents plus number of products) before it terminates. In addition to that, the SAMP-SB-D protocol requires additional messages for decommitments.

2.3.7 Service network formation with auctions

Vulkan & Jennings (2000) consider an allocation approach for the supply of *services*, i.e., service network formation. They present an English auction protocol, modified for the provision of services. A protocol for auctions arranged by the service providing agents, if the service seeking agents fail to do so, is proposed. Similar to this thesis, Vulkan & Jennings (2000) focus on two types

of agents, representing different organizations: Service-seeking agents (i.e., customer agents) and service-providing agents (i.e., SP agents). However, the authors neither assume that services can be re-sold, nor that services are utilized to produce value-added services to customers along service network tiers.

The proposed protocol constitutes an extension of the standard English auction to a multi-attribute auction (Bichler 2000). That is, besides the price, multiple additional service parameters are considered. The service-seeking customer agents weight valuations for service parameters (including the service price) differently. The service customer acts as an auctioneer and announces auction details to SP agents.

If the customer agent fails to initiate an auction, or if further details need to be negotiated after the auction, another protocol is initiated by the SP agents. Here, the SP agent that is approached by the customer agent and is not satisfied with a one-to-one negotiation (e.g., if communication is costly), initiates a 'pre-auction' protocol, in which SP agents compete for negotiation with the customer agents. The winning SP agent sends a take-it-or-leave-it offer for the service to the customer agent. The other SP agents do not interact with the customer agent, and receive payments from the winning agent for their cooperation.

The approach suggested by Vulkan & Jennings (2000) constitutes distributed allocation. Service dependencies are not considered, as a re-sale of services is not taken into account. The computational complexity of winner determination is in $\mathcal{P}$, since the auctioneer agent just has to calculate the utility of every offer received. The mechanism is budget balanced and individually rational, but not incentive compatible. The communication complexity depends on the increments of the English auctions, but is expected to be significantly lower as compared to multiple one-to-one negotiations.

Preist, Bartolini & Byde (2003) present an algorithm for decision making service composition agents. These agents buy component services in a set of English auctions, and sell composite services in reverse English auctions. The agents maintain a probabilistic model of the closing prices, based on past auctions. The authors present an algorithm to calculate the expected utility of bundled bids for any subset of auctions. To reduce the computational complexity, a simplified algorithm that does not assess any subset of auctions, is proposed.

Since they assume English auctions, the algorithms can be applied to agents participating in centralized, as well as distributed allocations. Service dependencies are considered and the probability of overcommitments can be minimized, though probabilistic approaches cannot guarantee the avoidance of over-

commitments. The approach does not include any alterations to a negotiation protocol; the allocation mechanism properties are those of standard English auctions.

2.3.8 Summary and research gap

Table 2.5: Requirement fulfillment of related approaches.
(● = fulfilled, ◐ = partly fulfilled, ○ = not fulfilled)

	distributed allocation	service dependencies	allocative efficiency	incentive compatibility	individual rationality	budget balance	allocation complexity
Bo & Lesser (2010)	●	○	○	○	●	●	$\mathcal{P}$
Endriss et al. (2006)	◐	◐	●	○	●	●	$\mathcal{NP}$
Nguyen & Jennings (2005)	●	○	○	○	○	●	$\mathcal{P}$
Walsh et al. (2000)	○	●	●	○	○	●	$\mathcal{NP}$
Walsh & Wellman (2003), SAMP-SB	◐	○	○	●	○	●	$\mathcal{P}$
Walsh & Wellman (2003), SAMP-SB-D	◐	●	○	○	●	●	$\mathcal{P}$
Vulkan & Jennings (2000)	●	○	○	○	●	●	$\mathcal{P}$
Preist et al. (2003)	●	◐	○	○	●	●	$\mathcal{P}$
This work	●	●	○	●	●	●	$\mathcal{P}$

Existing approaches for distributed allocation exist, but do not fulfill the identified requirements. The generic allocation approaches do not provide concrete mechanism details, and thus cannot be assessed regarding the requirements (e.g., (Sandholm & Lesser 2001, Parkes & Shneidman 2004)).

Table 2.5 shows the requirement fulfillment of related research that propose concrete mechanisms for resource allocation. All approaches are budget balanced, but only Walsh et al. (2000) and Walsh & Wellman (2003) can avoid overcommitments. However, the former approach is not distributed, and the latter is not individually rational. SAMP-SB-D (Walsh & Wellman 2003) can be applied to avoid the *consequences* of overcommitments, though the protocol makes auction bids non-binding, is not incentive compatible, and requires intermediaries for the allocation for each product.

The distributed approaches Bo & Lesser (2010), Nguyen & Jennings (2005), and Vulkan & Jennings (2000) cannot consider service dependencies. Endriss et al. (2006) assume that *any* agent involved in the production of a service would

participate in a single agreement, which contradicts the multi-tier characteristic of the SN. The approach of Preist et al. (2003) can minimize the probability of overcommitments, but cannot avoid them. In addition, these approaches are not incentive compatible.

The review shows that a research gap exists in the area of multiagent resource allocation which considers the distributed nature of SNs and avoids overcommitments. There is no approach that provides an appropriate allocation mechanism which constitutes an individually rational, budget balanced, incentive compatible polynomial time heuristic. This thesis proposes a multi-tier resource allocation protocol that avoids overcommitments by SPs, does not require central entities for coordination, is individually rational, budget balanced, incentive compatible for SPs, and has polynomial complexity.

Chapter 3

Design

This chapter provides the proposed protocol. First, the protocol is described, based on the game-theoretic model that was introduced in chapter 2. Second, the protocol is formally specified by using two techniques for protocol specification, UML sequence diagram (OMG 2011*b*) and PROMELA (Holzmann 1991). Third, an implementation of the protocol in a MAS is presented, which will be used for the simulation study in the succeeding chapter.

3.1 Game-theoretic protocol model

The protocol model is based on the formal framework presented in section 2.1.3.2.3. Supply and demand in SNs are allocated in sealed-bid second-price reverse service auctions in each tier: The bidding agents privately transfer their bids to the auctioneer, the lowest bidder wins and receives the payment of the second lowest bid (Vickrey 1961). This research analyzes three different bidding policies for SPs. Further, it assumes that the number of bidding agents is unknown to the bidders, and there is no collusion among bidding agents (Robinson 1985, Smith 1989).

The proposed protocol is defined as follows:

- Customer agents' offers o_{ij}^t are announced to SP agents which can *recursively* transfer own offers $o_{\ell m}^t$ to SP agents in the next tier.
- SP agents submit binding bids $b_{\ell m}^t$ to their customers, considering bids b_{mn}^t received from SP agents in the next tier.
- SP agents are informed about the acceptance of the bids they have pro-

vided and receive second-price payments (cf. Vickrey 1961), i.e.,

$$\psi^t_{\ell m} = \begin{cases} \min_{m \neq n} b^t_{\ell n} & \text{if } b^t_{\ell m} < \min_{m \neq n} b^t_{\ell n}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\min_{m \neq n} b^t_{\ell n}$ denotes the second highest bid.

For the composition of the SP agents' bids – which can be based on the utilization of either internal (own) or external (from other SPs) resources – this research investigates three different bidding policies: internal resources first (*IRF*), external resources first (*ERF*), and best price resources only (*BPRO*). Customer agents accept bids if and only if the provided service is produced to the full extend requested; i.e., customer agents do not accept the partial fulfillment of their requests.

Using the *IRF* bidding policy, SP agents try to establish contracts for their own resources first. Consequently, they request sub-services from providers in the next tiers only if the unallocated own resource capacity is insufficient to fulfill a customer agent's request. Thus, SP agents request sub-services exceeding the own resource capacity. If both own and external capacity are insufficient to fulfill a customer agent's request, SP agents submit bids on a partial fulfillment of the request, utilizing as many resources as possible. Therefore, downstream SP agents are able to combine partial bids to fulfill a customer agent's request even if none of the SP agents is able to fulfill the request using the own resources only. Formally, the SP agents' bidding function for the *IRF* bidding policy is defined as

$$b_{IRF}(s_{ij})^t = \begin{cases} c_j(s_{ij}) & \text{if } w_{ij} + \sum_{t'=0}^{t} \sum_{a_h \in \mathcal{A}_C} w_{hj} x^{t'}_{hj} \leq W_j, \\ c_j(s'_{ij}) + b^t_{jk} & \text{otherwise,} \end{cases}$$

with s'_{ij} such that $w'_{ij} = W_j - \sum_{t'=0}^{t} \sum_{a_h \in \mathcal{A}_C} w_{hj} x^{t'}_{hj}$ and $w_{jk} = w_{ij} - w'_{ij}$.

In contrast, using the *ERF* bidding policy, SP agents try to outsource as much of a customer agent's request as possible. Similarly to the *IRF* bidding policy, these SP agents combine partial bids and supplement received bids with utilization of own resources to arrive at the capacity required to fulfill the customer agent's request if possible. If both received bids and the SP agent's own capacity are insufficient to fulfill the customer agent's request, the SP agents submit partial proposals as for the *IRF* bidding policy. Formally, the SP agents' bidding function for the *ERF* bidding policy is defined as

$$b_{ERF}(s_{ij})^t = \begin{cases} b^t_{jk} & \text{if } w_{jk} = w_{ij}, \\ c_j(s'_{ij}) + b^t_{jk} & \text{otherwise,} \end{cases}$$

with s'_{ij} such that $w'_{ij} = \min(W_j - \sum_{t'=0}^{t} \sum_{a_h \in \mathcal{A}_C} w_{hj} x^{t'}_{hj}, w_{ij} - w_{jk})$.

The *BPRO* bidding policy avoids the utilization of non-best price (lowest marginal cost) resources. Thus, the SP agents compare the marginal cost which they have to pay to SP agents in the next tier (i.e., second prices) with the internal marginal cost of utilizing their own resources. Consequently, the SP agents using the *BPRO* bidding policy combine first-price bids to submit bids themselves if and only if the second price is below their internal cost. Obviously, the SP agents are more likely to submit partial bids using the *BPRO* bidding policy, as compared to the *IRF* and *ERF* bidding policies. However, the *BPRO* bidding policy allows to submit bids with the lowest possible prices. Formally, the SP agents' bidding function for the *BPRO* bidding policy is defined as

$$b_{BPRO}(s_{ij})^t = \begin{cases} b^t_{jk} & \text{if } \frac{b_{jk}}{w_{jk}} < \frac{c_{ij}}{w_{ij}}, \\ c_j(s'_{ij}) & \text{otherwise,} \end{cases}$$

with s'_{ij} such that $w'_{ij} = \min(W_j - \sum_{t'=0}^{t} \sum_{a_h \in \mathcal{A}_C} w_{hj} x^{t'}_{hj}, w_{ij})$.

3.2 Protocol specification

The game-theoretic protocol model defines the behavior of agents and their bidding policies. Implementing the protocol in a MAS requires a formal specification that defines message types and message sequences. For this purpose, *interaction protocol* specification techniques are available. The FIPA Contract net interaction protocol (CNP) (FIPA 2002*b*) – the FIPA interpretation of the original Contract net protocol proposed by Smith (1980) – provides a baseline protocol for resource allocation that can be enriched by recursion, distributed winner determination, and a specific allocation mechanism.

The proposed protocol allows subcontracting activities by the participants; i.e., a participant can evaluate if subcontracting is possible and feasible in advance to making binding proposals. The proposed interaction protocol is denoted as Multi-tier contract net protocol (MTCNP), since the protocol enables coordinated interactions over multiple service network tiers. This means that the protocol allows consideration of agreement dependencies over multiple tiers for subcontracting.

Adopting the existing CNP requires to extend its communicative acts (performatives) (FIPA 2002*a*). The *cfp* (call for proposals) can include customer requests which (i) explicitly require multiple services (e.g., storage and analysis services) or (ii) can be realized with a composite service (e.g., data mining service), potentially composed at run time. To calculate the service level of a composite service, service parameter aggregation definitions have to be defined for each pair of service parameter and composition pattern (section 2.1.3.2.2). The *inform-result* message type in the CNP does not relate to separate FIPA communicative acts but to the general *inform* act. Once an agent has completed one or more tasks, it sends a message to the initiator in the form of an *inform-done*; more information about the execution can be provided by an *inform-result*. In case of an *inform-result* message, the agent has to aggregate the results of the single services according to the parameter aggregation definitions. For example, the price of the data mining service can be calculated by summing up the prices of the single services (e.g., storage and analysis services) and the service provider's profit margin. In contrast, the throughput of the data mining service is calculated by the maximum throughput of the constituent services, since these are executed in sequence.

3.2.1 UML specification

Figure 3.1 shows the sequence diagram of the protocol's interactions. It consists of three encapsulated interaction sequences:

1. The *MTCNP-collect-proposals* interaction sequence includes collecting the proposals (bids) by the initiator (customer) from the participants (SPs). Collecting the proposals can also include subcontracting.

2. If and only if the participant has made a proposal, it is notified in the *MTCNP-acceptance-notification* interaction sequence about the allocation result.

3. If and only if the participant's proposal is accepted, the *MTCNP-execution* interaction sequence is performed. It includes the provision of information by the participants about the outcome of the execution of all allocated actions, which can also involve multiple tiers as for the allocation.

Details on the referenced interactions are given in the following.

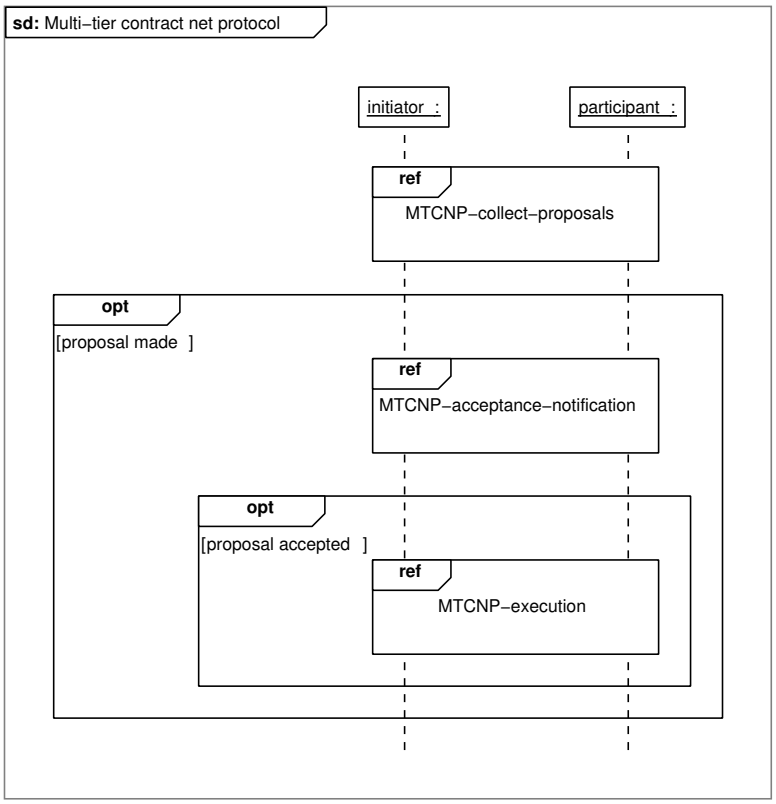

Figure 3.1: Multi-tier contract net protocol sequence diagram.

The MTCNP-collect-proposals interaction sequence is shown in figure 3.2. The initiator in tier λ sends a *cfp* message to the participants on the same tier. Optionally, if (i) participants exist in tier $\lambda + 1$ which are potentially capable of executing actions or sub-actions described in the *cfp* in tier λ and (ii) (one or more of) the tier λ participants prefer to subcontract actions or sub-actions in tier $\lambda + 1$ (acting as tier $\lambda + 1$ initiators), the MTCNP-collect-proposals interactions are recursively executed in tier λ+1. The recursion can be extended to an arbitrary number of tiers. Similarly to the CNP (FIPA 2002*b*), participants receiving the *cfp* generate n responses. The tier λ participants may decide that they refuse to propose, resulting in $i = n - j$ *refuse* act responses. Alternatively, j participants propose to perform the task, specified as *propose* acts.

@t is a time observation which defines t as the point in time when the *cfp* message is sent. d refers to the deadline duration which is also communicated in the *cfp* message. Thus, the constraints $\{t \ldots t + d\}$ specify that the *refuse* or *propose* messages should be received by the initiator in that interval (Haugen & Runde 2009).

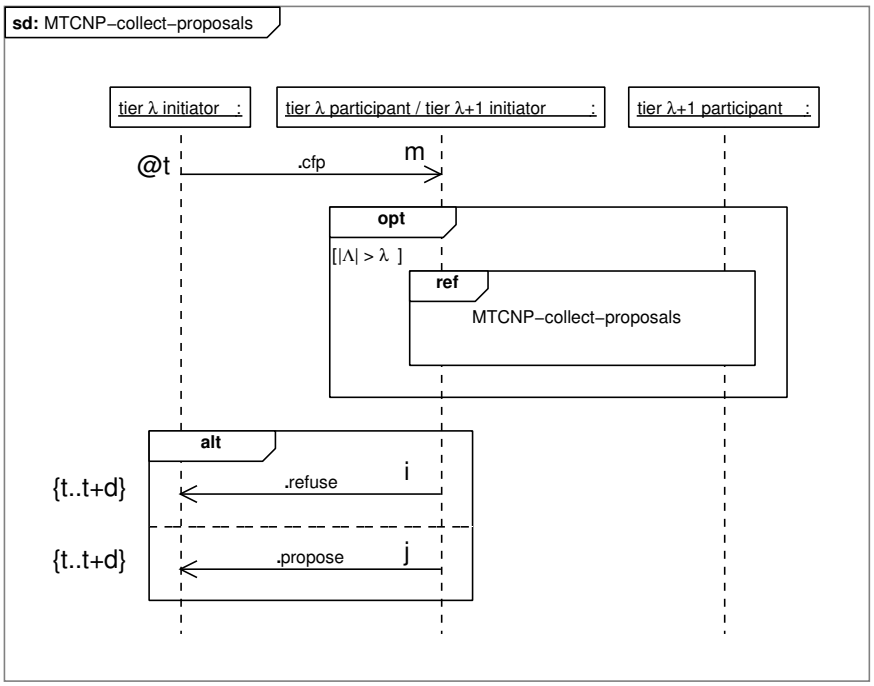

Figure 3.2: MTCNP-collect-proposals sequence diagram.

The initiator evaluates the j proposals received and selects participants to perform the tasks within the notification deadline duration nd; i.e., in the time interval $\{t \ldots t{+}nd\}$. nd is communicated to the participants in the *cfp* message. The MTCNP-acceptance-notification interaction sequence, shown in figure 3.3, covers informing the participants of the allocation result. The l participants of the selected proposals are notified with an *accept-proposal* message, defining the notification time nt in observation $@nt$. The remaining k participants receive a *reject-proposal* message. The recursion is executed to the number of tiers in which proposals were made (*propose* messages were sent).

The MTCNP-execution interaction sequence is shown in figure 3.4. Once the subcontracted participants in tier $\lambda + 1$ have completed the tasks, they send completion messages to the tier $\lambda + 1$ initiator in the form of an *inform-done* message or a more explanatory version in the form of an *inform-result* message. If a participant fails to complete one or more tasks, a *failure* message is sent. Similarly, the tier λ participants ($\lambda + 1$ initiators) report the state of the execution of tasks to the tier λ initiator. The result messages should be received by the initiators within the execution deadline duration ed; i.e., in the time interval $\{nt \ldots nt + ed\}$. The definition of ed is part of the agreement between the initiator and the participant. This definition can either be included in the *cfp* message or in the participants' proposals.

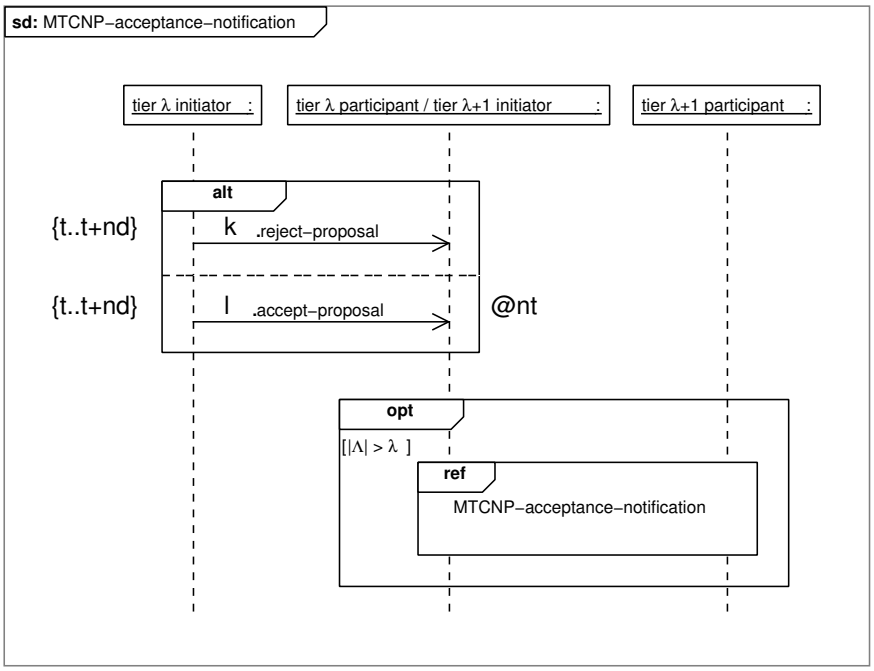

Figure 3.3: MTCNP-acceptance-notification sequence diagram.

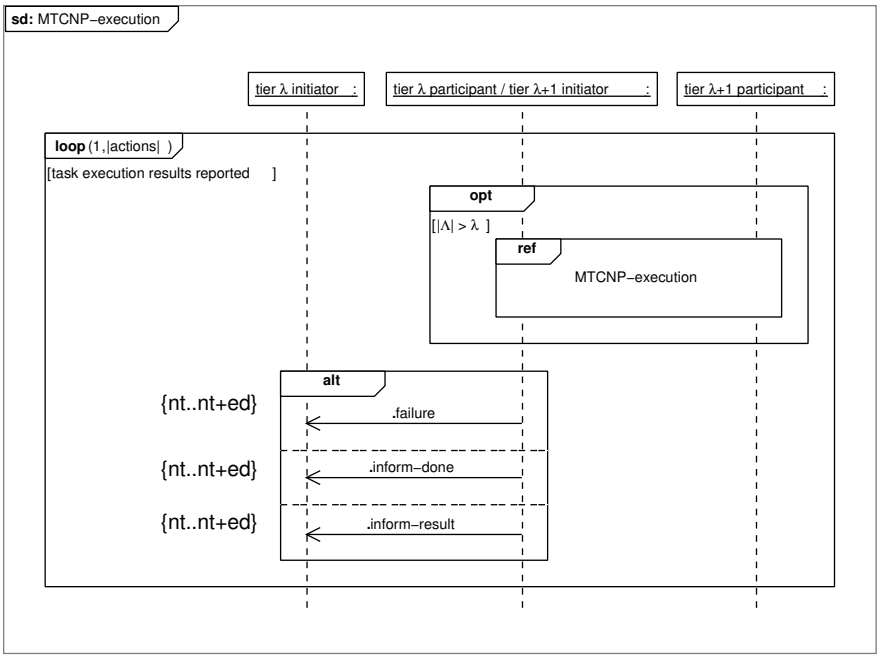

Figure 3.4: MTCNP-execution sequence diagram.

3.2.2 PROMELA model

PROMELA is a language for the specification and verification of protocol validation models – the complete and consistent sets of rules to govern interactions in distributed systems (Holzmann 1991). A protocol validation model can then be verified by a model checker (Clarke et al. 1999) such as SPIN (Holzmann 1997); SPIN accepts protocol specifications in PROMELA and correctness claims in linear temporal logic (LTL). PROMELA and SPIN have been successfully used in prior research on specifying and verifying interaction protocols (e.g., (Giordano et al. 2007)). To use these techniques, the behavior of the participants must be described in states for every (alternative) interaction. Therefore, this research constructs finite state machines (FSMs) for the participants, based on the UML sequence diagrams. This section first presents these FSMs, and then formulates the PROMELA model.

Figure 3.5 shows the FSM for the initiator, starting in state *s1*. On the initiator's side, state *s2* is reached after the *cfp* message has been sent. If the participant refuses to propose before the corresponding deadline, the end state is reached by the initiator. If a proposal is received by the initiator before the deadline, the initiator reaches state *s3*. If neither a *refuse* nor a *propose* message is received by the initiator during the deadline duration d, the protocol terminates with a timeout.

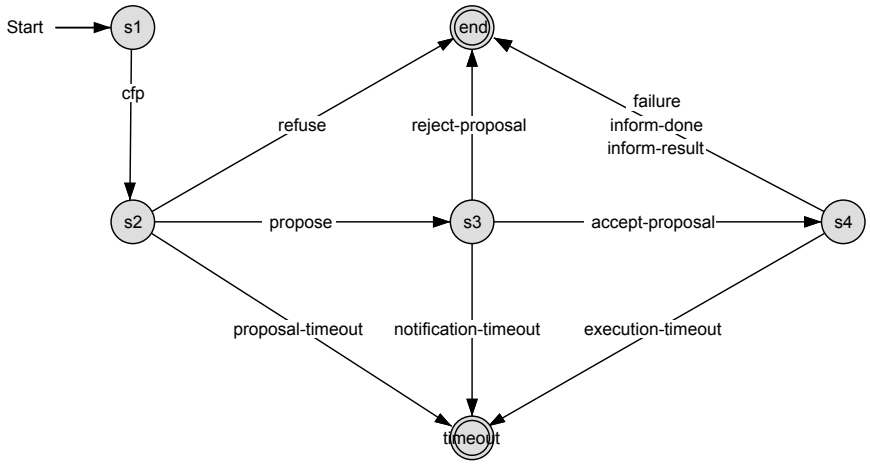

Figure 3.5: Initiator FSM.

If the proposal is accepted by the initiator within the notification deadline, state *s4* is reached. Similarly, the end state is reached if the proposal is rejected by the initiator within the notification deadline. Otherwise, a timeout occurs if neither a *accept-proposal* nor a *refuse-proposal* message is sent by the initiator

during the notification deadline duration *nd*.

From state *s4*, either the end state is reached by receiving the execution result or failure from the participant within the execution deadline duration *ed* or an execution timeout occurs.

Figure 3.6 shows the FSM for the participant. On the participant's side, receiving a *cfp* message leads to state *s2*, in which subcontracting with participants in the next tier is possible. Therefore, a participant in tier λ can act as initiator in tier $\lambda+1$ and send a *cfp* message to tier $\lambda+1$ participants, leading to state *s3* for the participant in tier λ. In state *s3*, the participant processes the response of the tier $\lambda+1$ participants to the *cfp* message and proceeds to state *s4* if the response to the *cfp* is received before the deadline duration *d*. Otherwise, a proposal timeout occurs. If the participant in tier λ does not send a *cfp* message in tier $\lambda+1$, state *s4* is reached.

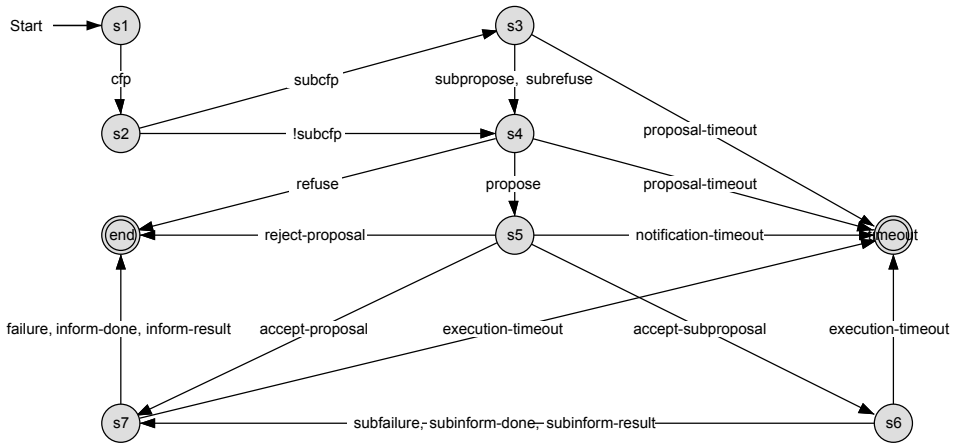

Figure 3.6: Participant FSM.

If the participant in tier λ refuses to propose – potentially after processing the result of subcontracting activities – to the initiator, the end state is reached. If the participants sends a proposal, state *s5* is entered. If the participant neither sends a *refuse* nor a *propose* message during the deadline duration *d*, the protocol terminates with a timeout.

Rejection of the proposal by the initiator leads to the end state for the participant. If the participant in tier λ did not collect proposals from tier $\lambda+1$ participants, the acceptance of the proposal by the initiator leads to state *s7*. Otherwise, the acceptance of the subproposal from tier $\lambda+1$ by the participant in tier λ leads to state *s6*. If the allocation notification is not communicated during the notification deadline duration *nd*, the protocol terminates with a timeout.

In state *s6*, the result of the execution in tier $\lambda + 1$ is processed. Either state *s7* is reached by receiving the execution result or failure by the initiator in tier $\lambda + 1$ (participant in tier λ) within the execution deadline duration *ed*, or an execution timeout occurs.

The result of the execution in tier λ is processed in state *s7*. In state *s7*, an execution timeout can also occur, leading to a protocol termination with a timeout. If the execution result or failure is reported on both tiers λ and $\lambda+1$, the end state is reached.

The interaction states of the protocol's participants – defined in the FSMs – are translated to PROMELA to be verified by model checking. In contrast to FSMs, PROMELA allows concurrency and recursion. In addition, incoming and outgoing messages can be differentiated. The PROMELA model of the MTCNP is shown in listing 3.1. An arbitrary recursion depth cannot be checked by SPIN. Thus, the model has been limited to seven tiers (listing 3.1, line 1) due to memory requirements for checking the proposed model.

Listing 3.1: MTCNP PROMELA model.

```
1  #define MAXDEPTH 7
2  mtype = {cfp, refuse, propose, reject_proposal,
       accept_proposal, failure, inform_done, inform_result}
3  int currentDepth = 0;
4  int nestedProposals = 0;
5  int slas[MAXDEPTH] = 0;
6  bool terminated=0;
7  init {
8   atomic{
9    chan toPinit = [1] of {mtype};
10   chan toIinit = [1] of {mtype};
11   chan toSubPinit = [1] of {mtype};
12   chan toSubIinit = [1] of {mtype};
13   run initiator(toPinit, toIinit); run participant(toPinit,
         toIinit,toSubPinit,toSubIinit, 0);
14  }
15 }
16 proctype initiator(chan toP, toI){
17  mtype ca;
18  s1: toP!cfp; goto s2;
19  s2: toI?ca;
20  if
21   ::(ca == refuse) -> goto end;
22   ::(ca == propose) -> goto s3;
23  fi;
```

```
24   s3:
25   if
26    ::1 -> toP!reject_proposal; goto end;
27    ::2 -> toP!accept_proposal; goto s4;
28   fi;
29   s4: toI?ca;
30   if
31    ::(ca == failure) -> goto end;
32    ::(ca == inform_done) -> goto end;
33    ::(ca == inform_result) -> goto end;
34   fi;
35   end: terminated=1; skip;
36  }
37  proctype participant(chan toP, toI, toSubP, toSubI; int tier){
38   mtype ca;
39   bool nestedProp=0;
40   s1: toP?ca;
41   if
42    ::(ca == cfp) -> goto s2;
43    ::else goto end;
44   fi;
45   s2: assert(nestedProp==0);
46   if
47    ::1 -> goto s4;
48    ::(2 && currentDepth < MAXDEPTH-1) ->
49     progress: atomic { chan toSubIi = [1] of {mtype};
50      chan toSubPi = [1] of {mtype};
51      run participant(toSubP, toSubI, toSubPi, toSubIi, tier+1);
52      toSubP!cfp; currentDepth++;
53      assert(currentDepth < MAXDEPTH);
54     } goto s3;
55   fi;
56   s3: toSubI?ca;
57   if
58    ::(ca == refuse) ->   goto s4;
59    ::(ca == propose) -> nestedProp=1; nestedProposals++; goto
        s4;
60   fi;
61   s4:
62   if
63    ::1 -> toI!refuse;
64     if
65      ::(nestedProp == 1) -> toSubP!reject_proposal; goto end;
```

```
66        ::else goto end;
67      fi;
68     ::2 -> if
69        ::((nestedProp == 1)||(tier==MAXDEPTH-1)) -> toI!propose;
              goto s5;
70        ::else toI!refuse; goto end;
71      fi;
72    fi;
73    s5: toP?ca;
74    if
75     ::(ca == reject_proposal) ->
76      if
77       ::(nestedProp == 1) -> toSubP!reject_proposal; goto end;
78       ::else goto end;
79      fi;
80     ::(ca == accept_proposal) -> slas[tier]++;
81      if
82       ::(nestedProp == 1) -> toSubP!accept_proposal; goto s6;
83       ::else goto s7;
84      fi;
85    fi;
86    s6: assert(nestedProp==1); toSubI?ca;
87    if
88     ::(ca == failure) -> goto s7;
89     ::(ca == inform_done) -> goto s7;
90     ::(ca == inform_result) -> goto s7;
91    fi;
92    s7:
93    if
94     ::1 -> toI!failure; goto end;
95     ::2 -> toI!inform_done; goto end;
96     ::3 -> toI!inform_result; goto end;
97    fi;
98    end: skip;
99  }
```

3.3 Implementation

This section presents the implementation of the proposed protocol and the simulation system. It elaborates on the used agent framework Jadex and the implementation of the protocol as a reusable library. Implementation details are given about the abstract types of the participating agents and the different

concrete agent types, i.e., customer agents and SP agents. Finally, the agent-based simulation system that will be used for the experimental evaluation, is described.

3.3.1 BDI agent system

This research adopts the belief-desire-intention (BDI) approach (Bratman, Israel & Pollack 1988) for implementing the agents using the protocol. First, the architectural foundations of BDI are described. Second, the Jadex BDI agent system is selected as the underlying framework of the implementation.

3.3.1.1 Belief-desire-intention architectures

The BDI model is the most common approach for deliberative (or hybrid) agents. These agents can deliberate over symbolic knowledge to reach given goals (Wooldridge 2000). The BDI architecture facilitates goal-driven system behavior. The model consists of the following concepts: beliefs capture informational attitudes realized as a data structure containing current facts about the world. Desires capture the motivational attitudes that form concrete goals if an agent has potentially the chance to fulfill the desire. Intentions capture the deliberative attitudes realized by reasoning to select appropriate actions to achieve given goals or to react to particular situations.

A BDI agent is equipped with sensors to assist it on its environmental awareness, and effectors to impact the environment by actions. A reasoning mechanism between the sensor input and the effector output deduces the necessary actions for achieving the agent's goals. The agent acquires new beliefs in response to changes in the environment and through the actions that it performs as it carries out its intentions (Bratman et al. 1988). Thus, the BDI agents allow reasoning regarding decisions to determine which – possibly conflicting – goals can be achieved and how the agent is going to achieve these goals.

Concrete architectures which follow the BDI approach include hybrid (e.g., procedural reasoning system, PRS (Ingrand, Georgeff & Rao 1992)) and deliberative agent architectures (e.g., intelligent resource-bounded machine architecture, IRMA (Bratman et al. 1988)) (Wooldridge & Jennings 1995).

3.3.1.2 Jadex BDI agent system

The Jadex BDI agent system (Pokahr et al. 2005) is based on the BDI approach. The computational reasoning model of Jadex follows the PRS archi-

tecture (Ingrand et al. 1992), a specific architecture for BDI systems. Jadex is implemented in the Java programming language.

All incoming messages and goal events constitute input to the reaction and deliberation mechanism in Jadex, which is common for PRS systems. This mechanism dispatches the events received to plans selected from the plan library. For the representation of beliefs, Jadex employs an object-oriented representation; arbitrary Java objects and sets of objects can be stored in named beliefs or belief sets. Jadex provides a set-oriented query language for operations on the beliefbase. Further, beliefs can be actively and conditionally monitored to trigger events or create and terminate goals.

As Jadex implements the BDI model, goals play a central role. In contrast to other BDI systems, goals in Jadex are explicitly represented in a goalbase. Thus, goals are accessible by the reasoning engine, as well as by plans which can actively monitor and alter goals. Goals are either adopted as top-level goals or sub-goals. If a goal is created from a plan, it is adopted as a sub-goal of the plan's root goal. The validity of goals can be expressed in terms of conditions on beliefs, i.e., goals can be specified to be valid only when certain conditions hold.

Jadex provides four different types of goals. Perform goals are directly associated with the execution of actions. These goals are considered to be reached if the specified actions are performed, independent of their outcomes. Achieve goals specify desired external effects to be reached; alternative plans can be provided how these goals can be reached. Query goals are similar to achieve goals, though the desired effects to be reached refer to internal effects in terms of availability of information. Maintain goals are utilized to sustain a desired state. That is, corresponding plans are executed to reestablish the desired state whenever the state has been changed to an undesired condition.

Jadex separates plans into a head and a body part. A plan's head specifies the conditions when a plan may be selected for execution. This includes events and goals handled by the plan, preconditions of its execution, as well as context conditions which must hold during plan execution. A plan's body provides a pre-defined sequence of actions, written in the Java programming language, which is executed when the plan is selected for execution.

Capabilities define a grouping of beliefs, goals, plans, and events which are required for a certain functionality (e.g., for interactions in accordance to a specific protocol): Jadex capabilities encapsulate functionalities in reusable modules which can be linked by import and export mechanisms (Pokahr et al. 2005).

3.3.2 Protocol implementation

The MTCNP has been implemented as a reusable capability for the Jadex BDI agent system version 0.96 on basis of the protocol capability included in the Jadex release. The MTCNP capability contains beliefs for the message and execution timeouts, a message filter to decide which messages are handled by the capability, as well as a data structure to store the relationships of proposals received and sent, i.e., the *proposal_compositions* belief is used to determine the proposals which have been used to create a proposal sent to a customer in a downstream tier.

3.3.2.1 Initiator

On the initiator's side without downstream customers, the *mtcnp_initiate* achieve goal constitutes the top-level goal of a MTCNP interaction. This goal is dispatched primary by the customer agents to initiate the interaction with SPs. Goal parameters include the *cfp* object as an input parameter, specifying the service requested, and a *result* object as an output parameter, giving details about the result of the service provision (e.g., service level). The *mtcnp_initiate* goal can be dispatched from an arbitrary plan providing the *cfp* object, as well as receiving SPs.

Once the customer has received proposals, the *mtcnp_evaluate_proposals* query goal – which determines the acceptable proposals – is instantiated. Thus, it takes *proposals* as an input parameter and provides the *acceptables* as an output parameter. This is realized by an evaluator plan implementation which is mapped to the *mtcnp_evaluate_proposals* query goal. The result of the service provision is not handled by a separate goal but by the *mtcnp_initiate* achieve goal as an output parameter.

3.3.2.2 Participant

On the participant's side, the *mtcnp_make_proposal* query goal constitutes the top-level goal of a MTCNP interaction. It is adopted as soon as a participating agent receives a *cfp* message. Hence, it includes the *cfp* as an input parameter and the *proposal* as an output parameter. Since proposals can be constructed utilizing received proposals from the next tier, the *mtcnp_make_proposal* query goal also includes a parameter to store the nested proposals received (i.e., the parameter *nested_proposal_messages*). For creating the actual *proposal* object, a concrete, domain-dependent implementation of at least one corresponding plan is required which is mapped to the *mtcnp_make_proposal* query goal. This

plan is also responsible for initiating the MTCNP interactions to the next tier.

The MTCNP capability provides the *mtcnp_collect_proposals* achieve goal for this purpose. This goal is similar to the *mtcnp_initiate* achieve goal, though it (i) constitutes a sub-goal of the *mtcnp_make_proposal* query goal and (ii) does not automatically result in the adoption of a *mtcnp_evaluate_proposals* query goal, as the final decision of acceptance is performed by the customer of the agent adopting the *mtcnp_collect_proposals* achieve goal. The plan implementations corresponding to this goal are thus not only responsible to initiate the collection of proposals from the next tier, but also to construct concrete proposals sent to customers from the proposals received. The *proposal_compositions* belief is used to store the relationship of proposals sent and received.

Once the customer has evaluated the (composed) proposals, the participating agents are informed about the allocation's result. This leads to adoption of the *mtcnp_execute_task* achieve goal on proposal acceptance or the *mtcnp_handle_rejected_proposal* achieve goal on proposal rejection. For both goals, the corresponding plan implementations have to handle the nested proposals received if existent. Handling nested proposals is trivial in case of proposal rejection: The implementation of the MTCNP capability includes functionality to reject all nested proposals.

In case of proposal acceptance, the nested proposals that have been utilized to construct the accepted proposal, have to be determined. The capability defines the *proposal_compositions* belief and the *nested_proposal_messages* goal parameter for this task. The latter is mapped from the *mtcnp_make_proposal* query goal to the *mtcnp_execute_task* achieve goal internally by the MTCNP capability without requiring any additional, domain-dependent implementation for specific use cases. First, the nested proposals corresponding to the accepted composed proposal are looked up, using the *proposal_compositions* belief data structure. Then, these proposals are used to determine the corresponding proposal messages from the *nested_proposal_messages* goal parameter. When the participating agent has determined the nested acceptables, it can dispatch the *mtcnp_acceptance_notification* achieve goal as a sub-goal of the *mtcnp_execute_task* achieve goal. The *mtcnp_acceptance_notification* achieve goal uses the nested proposal messages, as well as the nested proposals as input parameters.

The *mtcnp_acceptance_notification* achieve goal includes the results of the provision of the sub-services as output parameters. Finally, the (composed) result of all (nested) service provisions is forwarded to the *mtcnp_execute_task* achieve goal via the mapped plan implementation and sent to the initiating

agent by the capability.

3.3.3 Agent implementation

The agents have been implemented as Jadex agents that are independent of domain-specific particularities. Thus, this implementation can be specialized for domain-specific resource allocation in multi-tier SNs. A generic agent is provided for both customer and SP agents.

3.3.3.1 Customer agent

The customer agent in Jadex has beliefs about its demand for services, the valuation for these services, and the potential SPs. In addition, the customer agent has beliefsets of established agreements with SPs. Since the customer agent only interacts with SPs and does not offer services itself, this agent is not aware of the multi-tier nature of the SN.

Once a customer agent is started, it tries to establish contracts with SPs until its demand for services is fulfilled. Therefore, it adopts the *mtcnp_initiate* achieve goal, giving the service specifications in the *cfp* message.

The *mtcnp_evaluate_proposals* query goal evaluates the *proposal* messages received and determines the acceptable proposals received via one of its associated plan implementations, i.e., via evaluation plan instances. As the proposals received are binding for SP agents, the customer agent adds corresponding agreements to its beliefset upon determination of acceptables.

Since the costs for the service provisions are also determined via the established agreements, the customer agent is directly able to compute the resulting utility of established agreements. The capability then informs the SPs about the outcome of the allocation and includes the results of the service provisions in an output parameter of the *mtcnp_initiate* achieve goal.

3.3.3.2 Service provider agent

The SP Jadex agent has beliefs regarding its available capacity, the cost of capacity utilization, and the potential SPs in the next tier. The SP agent has a beliefset of established agreements with customers and other SP agents. Further, the SP agent's current bidding policy is represented in a belief. The allocation interaction on SP side is initiated by *cfp* messages received from customer agents.

The receipt of a *cfp* message – which includes the requested services' specifications – leads to an adoption of the *mtcnp_make_proposal* query goal by

the SP agent. Depending on a SP agent's bidding policy, it requests different sub-services from SP agents in the next tier.

Using the *IRF* bidding policy, the SP agent tries to provide the requested services, using as many of its own resources as possible. If it is able to fulfill the customer requests without external resources, the SP agent does not request additional sub-services from SPs in the next tier. In contrast, if the SP agent's own resources are either insufficient or incapable to provide the requested services, the SP agent submits *cfp* messages to the SPs in the next tier by adopting the *mtcnp_collect_proposals* achieve goal, requesting the sub-services which are essentially required to fulfill the customer's demand.

The *ERF* bidding policy differs from the *IRF* bidding policy in the priority of external and internal resources. Thus, a SP agent using the *ERF* bidding policy tries to fulfill the customer requests, using as many procured sub-services as possible by adopting the *mtcnp_collect_proposals* achieve goal. If it is able to provide the requested services using external resources from SPs in the next tier, the SP agent does not utilize its own resources. However, if the sub-services offered by SP agents in the next tier are insufficient or incapable to fulfill a customer agent's request to the full extend, the SP agent using the *ERF* bidding policy uses its own resources to create a proposal which fulfills the customer agent's request to the largest extend possible.

The *BPRO* bidding policy is different from the *IRF* and *ERF* bidding policies in terms of a combination of internal and external resources, as it avoids this combination. A SP agent using this bidding policy submits *cfp* messages to the SPs in the next tier, using the same service specification it receives from customers by adopting the *mtcnp_collect_proposals* achieve goal. It first requests the whole service as a sub-service, though this service can also be provided by multiple SPs in the next tier by a combination of sub-services. Second, the procurement costs are compared to the cost of utilizing the SP agent's own resources. Finally, the SP agent using the *BPRO* bidding policy submits *proposal* messages to its customer agents, which include the alternative with the lowest cost. On the one hand, this bidding policy ensures that only the cost-optimal resources are utilized. On the other hand, this can lead to partial proposals in terms of sub-services, which do not fulfill the customer agent's request to the full extend.

Independent of the bidding policy, once a proposal has been created, the SP agent provides this proposal to the *mtcnp_make_proposal* query goal as an output parameter. The *proposal_compositions* belief is used to store the relationship of proposals sent and received. The MTCNP capability then handles

the submission of the proposal to the customer agents.

The receipt of the acceptance notification by a SP agent leads to adoption of the *mtcnp_execute_task* achieve goal on proposal acceptance or the *mtcnp_handle_rejected_proposal* achieve goal on proposal rejection. If the proposal is rejected, all nested proposals are rejected by the MTCNP capability.

The *proposal_compositions* belief and the *nested_proposal_messages* goal parameter are utilized to determine the nested proposals that have been used to construct the accepted proposal. The SP agent determines nested acceptables and dispatches the *mtcnp_acceptance_notification* achieve goal to inform the SPs in the next tier about the allocation outcome.

The composed result of all direct and nested service provisions is forwarded to the *mtcnp_execute_task* achieve goal via the mapped plan implementation and send to the SP's customer agents by the capability.

3.3.4 Simulation system architecture

The simulation system consists of a simulation manager agent, customer agents, and SP agents based on the Jadex agent framework. The overall architecture of the simulation system is shown in figure 3.7. The simulation manager is capable of instantiating experiments with different parameters, reading the configuration from simulation setup files.

Customer and SP agents are created and destroyed by the simulation manager agent with help of the Jadex agent management system (AMS) agent, which is responsible for creating and destroying agents in the Jadex agent system. The simulation manager agent sends corresponding messages to the AMS agent. These messages include details of the agent creation in terms of, e.g., the number of customer and SP agents, valuation functions, and cost functions. The details can be passed to agents on creation as arguments to the agent configuration.

Once all agents have been started, the resource allocation is conducted in accordance to the MTCNP. The agents subsequently report the allocation outcome, as well as their utility to the simulation manager agent. Finally, the simulation manager produces output files which include details about the agents' utility, the utilitarian social welfare, established contracts, and allocation details for debugging purposes. To calculate the utility ratio of the distributed allocation in comparison to the socially optimal allocations' welfare, reference values are computed using the CPLEX optimization engine (IBM 2011). The socially optimal allocations' welfare values are output in the simulation result files.

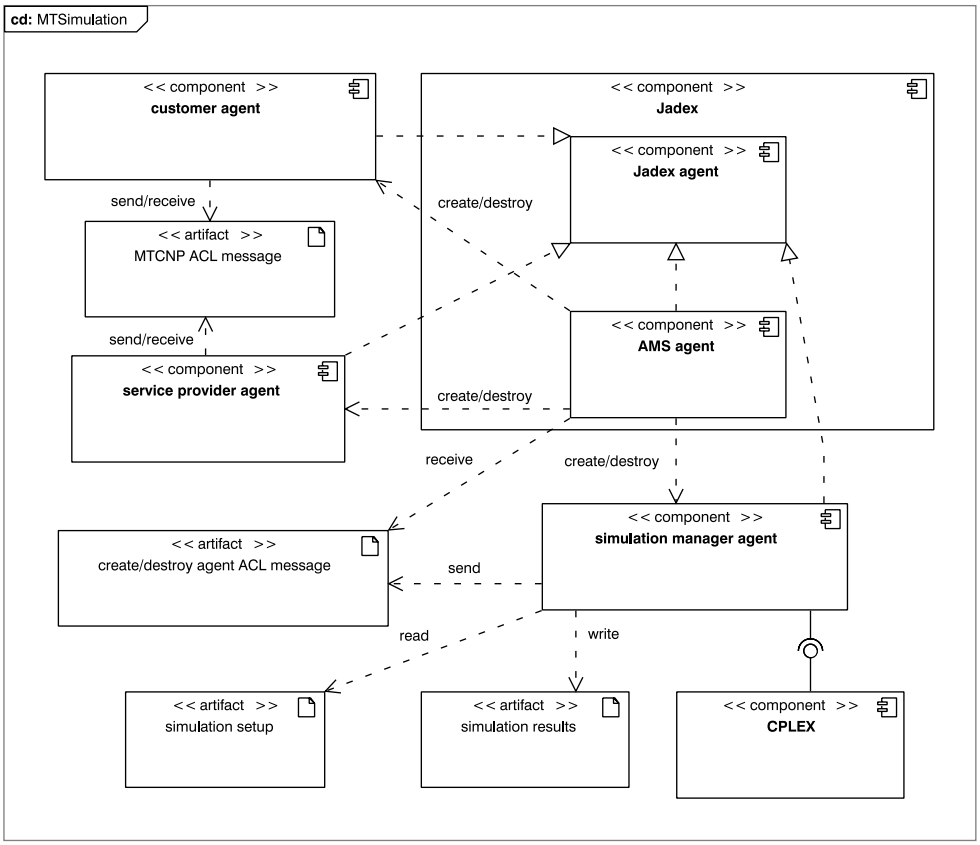

Figure 3.7: Simulation system architecture.

Chapter 4

Evaluation

This chapter states the evaluation of the proposed protocol. First, it validates the protocol specification based on the formal, game-theoretic model. Second, it verifies safety and liveness properties of the PROMELA model by means of the model checker SPIN. Third, it presents and discusses the results of the simulation study that evaluates the protocol implementation.

4.1 Formal protocol analysis

This section evaluates the formal protocol specification, based on the game-theoretic model, by means of proofs for (i) distributed allocation, (ii) service dependencies (avoidance of overcommitments), and the proposed mechanism's properties: (iii) allocative efficiency, (iv) incentive compatibility, (v) individual rationality, (vi) budget balance, and (vii) allocation complexity, in accordance to the requirements (section 2.2.2).

4.1.1 Distributed allocation

Theorem 3. *The proposed protocol allows for distributed resource allocation.*

Proof. Customer agents $a_i \in \mathcal{A}_C$ send their offers to $|\mathcal{A}_{SP,1}|$ SP agents, which respond with corresponding bids; i.e., in tier 1, there are at least $|\mathcal{A}_C|$ auctions, guided by the customer agents. In tier λ, $|\mathcal{A}_{SP,\lambda}|$ SP agents send their offers to $|\mathcal{A}_{SP,\lambda+1}|$ SP agents in the next tier, which respond with corresponding bids; i.e., in tier λ, there are at least $|\mathcal{A}_{SP,\lambda}|$ auctions, guided by the SP agents $a_j \in \mathcal{A}_{SP,\lambda}$. Therefore, the protocol allows for distributed resource allocation. □

4.1.2 Service dependencies

In theorem 4, it is shown that the protocol avoids overcommitments (i.e., no service is procured which cannot be used to produce another service for a customer) as follows. Assume that service s_{ij} is included in the allocation ($x_{ij}^t = 1$). Then there has been a binding bid for this service by agent a_j and the costs for providing s_{ij} – including lower tier bids for services utilized to produce s_{ij} – are less or equal to a_j's bid. However, according to the protocol, a_j submits a bid if it has received bids for all required services on lower SN tiers. By recursion, it can be concluded that there cannot be overcommitments in this case.

In the opposite case, s_{ij} is *not* included in the allocation ($x_{ij}^t = 0$), there is either no bid from agent a_j to a_i, or the bid is rejected. In the latter case, a_j can reject any corresponding bid on lower tiers, and thus no corresponding service is allocated.

Theorem 4. *The proposed protocol avoids overcommitments.*

Proof. Let $a_i \in \mathcal{A}$, $a_j \in \mathcal{A}_{SP}$. A necessary condition for $x_{ij}^t = 1$ is that a binding bid b_{ij}^t has been submitted. In addition, a_j expects a positive utility by submitting a binding bid b_{ij}^t if and only if it has received $b_{jk}^t \forall s_{jk} \in \varphi(s_{ij})$, such that $b_{ij}^t \geq c_{ij} + \sum_{s_{jk} \in \varphi(s_{ij})} b_{jk}^t$ with $b_{jk}^t < \min_{k \neq \ell} b_{j\ell}^t$. That is, a_j has received binding bids for all services required for the provision of s_{ij} and the total costs do not exceed the minimal payment from a_i. If b_{ij}^t is accepted by a_i ($x_{ij}^t = 1$), a_j accepts b_{jk}^t ($x_{jk}^t = 1$) with $b_{jk}^t < \min_{k \neq \ell} b_{j\ell}^t$. By recursion, it yields $x_{\ell m}^t = 1 \, \forall s_{\ell m} \in \varphi(s_{ij})$ and by (2.1) it follows that $z^t(s_{ij}) = x^t(s_{ij}) = 1$.

If a_j has either (i) not received $b_{jk}^t \forall s_{jk} \in \varphi(s_{ij})$ such that $b_{ij}^t \geq c_{ij} + \sum_{s_{jk} \in \varphi(s_{ij})} b_{jk}^t$ with $b_{jk}^t < \min_{k \neq \ell} b_{j\ell}^t$, or (ii) b_{ij}^t is rejected by a_i ($x_{ij}^t = 0$), a_j rejects all b_{jk}^t ($x_{jk}^t = 0$). By (2.1) it follows that $z^t(s_{ij}) = x^t(s_{ij}) = 0$. □

4.1.3 Allocative efficiency

In theorem 5, it is proven that for a single customer the proposed protocol results in a utilitarian socially optimal allocation for non-substitutable resources. Theorem 4 is used to eliminate overcommitments in this proof. Due to the second-price payments, it is individually rational for SP agents to bid their true marginal cost. Therefore, in each tier, the SP agent with the lowest bid, and therewith the lowest marginal cost, is allocated. Since the valuation is given and the costs for service provision are minimized, allocations are utilitarian socially optimal for single user agents and non-substitutable resources.

Theorem 5. *For a single customer request, the proposed protocol results in a utilitarian socially optimal allocation for non-substitutable resources.*

Proof. By theorem 4 it yields $z_{ij}^t = x_{ij}^t$. By assumption, collusion between SP agents is excluded (section 2.1.3.2). As SPs receive the second-price payment, it is individually rational for SPs to submit bids such that $b_{ij}^t = c_{ij} + \sum_{s_{jk} \in \varphi(s_{ij})} b_{jk}^t$ with $b_{jk}^t < \min_{k \neq \ell} b_{j\ell}^t$. It follows $b_{ij}^t = \sum_{s_{\ell m} \in \varphi(s_{ij})} c_{\ell m}$ with $c_{\ell m} < \min_{m \neq n} c_{\ell n}$. That is, the SPs with minimal costs are contracted along all service network paths. As v_{ij} is given and costs are minimized, the allocation is utilitarian socially optimal. □

4.1.4 Incentive compatibility

It is shown in theorem 6 that the proposed protocol is incentive compatible for SP agents; it is always rational for SP agents to bid their true marginal cost.

Theorem 6. *The proposed protocol is incentive compatible for SP agents.*

Proof. Let $a_j \in \mathcal{A}_{SP}$ be a service provider bidding on an offer from agent $a_i \in \mathcal{A}$ for service $s_{ij} \in \mathcal{S}$ and let c_{ij} be the total marginal cost for agent a_j for the provision of service s_{ij}. If $\exists b_{ik}^t \in \mathcal{B}^t : b_{ik}^t < c_{ij}$, agent a_j cannot gain positive utility by submitting a bid b_{ij}^t, such that $b_{ij}^t < b_{ik}^t \forall a_k \in \mathcal{A}_{SP}$, since it would win the reverse auction, but cost would exceed monetary compensation. If $\nexists b_{ik}^t \in \mathcal{B}^t : b_{ik}^t < c_{ij}$, there is also no incentive for agent a_j to submit a bid $b_{ij}^t < c_{ij}$, since it would win the auction and receive the second-price payment $\min_{k \neq j} b_{ik}^t$. □

4.1.5 Individual rationality

In theorem 7, it is shown that participation in the protocol is individually rational, i.e., no agent can obtain negative utility by participating. This results from the avoidance of overcommitments, proven in theorem 4, which results in zero penalty payments. Since SP agents bid their true marginal cost (theorem 6), and payments for service provision are never below the submitted bids, the protocol is individually rational for SP agents. Customer agents never accept bids higher than their valuation for the requested service and thus cannot receive negative utility from participating.

Theorem 7. *The proposed protocol is individually rational.*

Proof. No SP agent can receive negative utility, since by theorem 4, it yields $z_{ij}^t = x_{ij}^t \, \forall a_i, a_j \in \mathcal{A}, t \in \mathcal{T}$ and therewith no penalty payments, since $x_{ij}^t \rho_j^t(s_{ij}) = 0 \forall a_i, a_j \in \mathcal{A}, t \in \mathcal{T}$. No SP agent $a_j \in \mathcal{A}_{SP}$ overbids its marginal cost, as shown in theorem 6. Therefore, SP agents' bids are equal to the marginal cost for utilization of own resources and the procurement of lower tier services. Since

$\psi_{ij}^t \geq b_{ij}^t \forall a_i, a_j \in \mathcal{A}, t \in \mathcal{T}$, SP agents cannot receive negative utility from participating. Customer agents do not have costs besides payments and do not accept bids which exceed their valuation for a service. Thus, customer agents also cannot receive negative utility from participating. □

4.1.6 Budget balance

Theorem 8. *The proposed protocol is budget balanced.*

Proof. Since both ordinary payments ψ_{ij}^t, as well as penalty payments $\rho_j^t(s_{ij})$, distribute wealth between the participating agents only, the proposed protocol is budget balanced. □

4.1.7 Allocation complexity

Theorem 9. *The proposed protocol complexity is in $\mathcal{P}$.*

Proof. For winner and price determination in the single auctions, customer agents have to determine the lowest and second-lowest bids, i.e., sort the bids received. This can be done in polynomial time for each customer agent, and scales linearly with the number of customer agents. SP agents calculate their bids, respectively marginal cost for service provision, based on the bids received in higher tiers of the SN. They have to sort the bids received, and perform a simple comparison and/or addition to calculate the concrete bids. Hence, the proposed protocol is in $\mathcal{P}$. □

4.2 Model checking

Model checking is used to verify the logical consistency of a formal protocol specification, independent of the implementation, by an automated validation system. In addition to consistency, further correctness properties can be validated (Holzmann 1991).

To facilitate the verification of the protocol, this research assumes a deterministic disaggregation of customer requests to upstream SN tiers; i.e., the number of SLAs required in upstream SN tiers is known a priori. Thus, the required number of agreements in tier λ can be determined by multiplying the corresponding number of agreements in tier $\lambda+1$ with the disaggregation factor. If the number of required SLAs on each tier would be determined at run time, depending on the offers made on lower tiers (i.e., randomly), different experiments would not be comparable. Further, this work assumes that agreements are required in all tiers of the SN, since the decision whether subcontracting is

required or feasible is beyond the scope of the protocol specification. These assumptions do not mitigate the applicability of the protocol in non-deterministic environments with varying numbers of tiers – any resulting parameter setting can be verified ex post, though the set of possible parameter values has to be defined. As a result, the protocol enables multi-tier resource allocation if and only if the number of agreements – adjusted by the disaggregation factor from tier $\lambda = 0$ along the whole SN – is equal in all tiers after the protocol interactions for a single given customer requests have terminated; i.e.,

$$z(s_{ij}) = x(s_{ij})\ \forall s_{ij} \in \mathcal{S} \Rightarrow |\mathcal{S}_0| = |\mathcal{S}_n| \forall \mathcal{S}_n \subset S, n > 0, \tag{4.1}$$

where $\mathcal{S}_\lambda$ denotes the set of contracted services in tier λ. For a single customer request, establishing a SLA can either fail or a SLA is established in tier $\lambda = 0$ once the execution of the protocol terminates; i.e.,

$$z(s_{ij}) = x(s_{ij})\ \forall s_{ij} \in \mathcal{S} \Rightarrow |\mathcal{S}_0| = 0 \vee |\mathcal{S}_0| = 1. \tag{4.2}$$

The model checker SPIN can verify basic safety properties such as absence of deadlock, unreachable code, unspecified receptions, and invalid end states on the basis of basic PROMELA models (Holzmann 1997, pp. 287–288). In addition, SPIN checks the validity of user defined assertions. Therefore, SPIN can generate C-code for a model checking system specific for the given protocol model (`spin -a MTCNP.pmla`). The code is compiled for pure safety properties only (`gcc -o pan -DSAFETY pan.c`). The execution of the generated program shows that the model satisfies the safety properties mentioned above, considering about 1500 states/transitions, which underlines the need for an automated verification. Since zero unreachable states are reported and no assertion errors are raised, safety properties are proven for the PROMELA model.

The liveness properties formulae have been limited to seven tiers due to memory requirements for checking the proposed model (section 3.2.2). There is no evidence that a verification would not be possible for a larger number of tiers. However, due to the recursive model, the memory requirements for model checking increase exponentially with the number of tiers.

To assess the fulfillment of the requirements, this research needs to verify that the protocol enables multi-tier resource allocation (4.1) and proper termination of the protocol for a customer request (4.2). As the maximum number of tiers has been limited to seven tiers, (4.1) can be proven for $n \leq 6$ only. These properties have to be provided in LTL to be proven by SPIN:

$$\Diamond(\bigwedge_{n=1}^{6} |\mathcal{S}_0| = |\mathcal{S}_n| \wedge (|\mathcal{S}_0| = 0 \vee |\mathcal{S}_0| = 1)). \tag{4.3}$$

Therefore, the LTL formula has to be translated to PROMELA. First, symbol definitions have to be defined as shown in listing 4.1.

Listing 4.1: Symbol definitions.

```
1 #define t       terminated
2 #define c01     slas[0] == slas[1]
3 #define c12     slas[0] == slas[2]
4 #define c23     slas[0] == slas[3]
5 #define c34     slas[0] == slas[4]
6 #define c45     slas[0] == slas[5]
7 #define c56     slas[0] == slas[6]
8 #define c0_0    slas[0] == 0
9 #define c0_1    slas[0] == 1
```

Second, a LTL formula equivalent to (4.3) can be passed in command-line arguments to SPIN for translation into never claims. Never claims formalize behavior that should never happen – potential violations of correctness requirements. Thus, they are corresponding to the negated formulae, formalizing violations of the original:

```
spin -f '!(<> (t  && s01  && s12 && s23 && s34
&& s45 && s56 && (s0_0  || s0_1)))'
```

The resulting never claims can then be passed to SPIN in a separate file (`MTCNP.pmla.ltl`) in order to include them in the generated program (`spin -N MTCNP.pmla.ltl -a MTCNP.pmla`). Here, the code has to be compiled without the optimizations possible for pure safety properties, i.e., state compression, partial order reduction, and a transition coarsening strategy (Holzmann 1997, p. 288). The verifier shows that the resulting state space is three to four times larger than with safety properties optimizations; it reports unreachable states for the model for the generated never claims, which proves the absence of the negated formulae; i.e., (4.3) is proven for the PROMELA model.

4.3 Simulation

In the simulation study, the resource allocative efficiency of the protocol is compared to the centralized socially optimal allocation. This study considers different settings with regard to, e.g., network topology, number of customer and SP agents. Further, the experiments are executed with and without substitutable resources, i.e., SP agents can decide if they use own resources or procure sub-services from SP agents in the next tier in case of substitutable resources.

4.3.1 Experimental design

In the experiments, different scenarios from Cloud computing are considered. In Cloud computing, the assumption of substitutable resources holds in case of generic service requests (e.g., physical resource near IaaS services such as CPU or storage resources), while non-substitutable resources exist for specific services (e.g., SaaS services like specific format conversions or optimizations) (Armbrust et al. 2010). The experiments are conducted for different network topologies, as well as different agent parameters (e.g., cost). Details about the experimental design are given in the following.

Table 4.1 and 4.2 show the deterministic and random simulation experiment parameters. A set of experiment instances is generated for every deterministic parameter permutation; i.e., each of the deterministic parameter value vectors is supplemented by a set of vectors of the random parameters. The required capacity, SP cost, SP capacities, and SP agent progression intervals, are randomly determined, based on equal distributions for each experiment instance. The valuation function for customer agents is defined as $v_{ij} = 10w_{ij}$ to guarantee that at least one non-trivial solution exists for each allocation problem.

Table 4.1: Deterministic experiment parameters.

parameter	**range**
number of customer agents ($\|\mathcal{A}_C\|$)	$\{5, 10, 15, \ldots, 100\}$
number of SP agents in tier 1 ($\|\mathcal{A}_{SP,1}\|$)	$\{2, 4, 6, 8, 10\}$
number of SP agent tiers ($\|\Lambda\|$)	$\{1, 2, 3, 4, 5\}$

Table 4.2: Random experiment parameters.

parameter	**range** ($\subset \mathbb{N}$)
required capacity (w_{ij})	$[1, 5]$
SP agents' cost (c_{ij})	$[5, 10]$
SP agents' capacities (W_j)	$[5, 10]$
SP agent progression interval (δ_{SP})	$[2, 10]$

Let $\mathcal{A}_{SP,\lambda}$ denote the SP agents in tier $\lambda \in \Lambda$. The SN topologies are created as follows. For concurrent customer agents, SP agents in tier 1, $a_j \in \mathcal{A}_{SP,1}$ are generated randomly from an equal distribution in the progression interval δ_{SP}. Then, each SP agent is connected with at least one customer agent $a_i \in \mathcal{A}_C$. Further, additional connections are created relative to all possible additional connections $(|\mathcal{A}_{SP,1}| - 1) \cdot |\mathcal{A}_C|$. They are generated randomly from a normal distribution with a mean of $0.1 \cdot (|\mathcal{A}_{SP,1}| - 1) \cdot |\mathcal{A}_C|$ and a standard deviation of $0.1 \cdot (|\mathcal{A}_{SP,1}| - 1) \cdot |\mathcal{A}_C|$.

The number of SP agents in the next tier is again generated randomly from an equal distribution in the progression interval δ_{SP}. Next, each SP agent in tier $\lambda + 1$ is randomly connected to a SP agent in tier λ. Figure 4.1 illustrates an example of the SN creation process for this minimal set of edges.

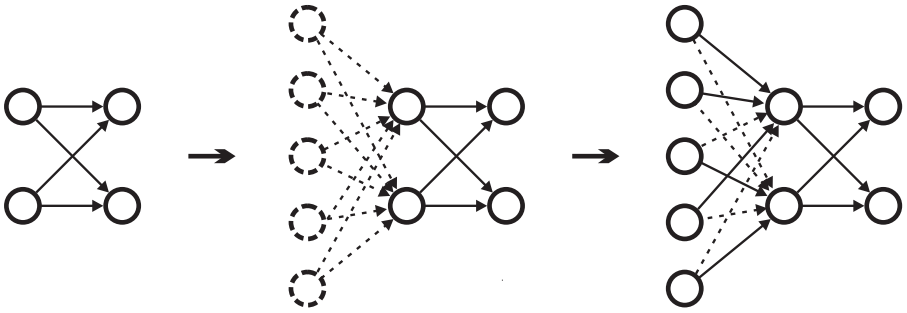

Figure 4.1: SN creation process example for minimal edges.

Additional *direct* connections are established, relative to all possible additional *direct* connections $(|\mathcal{A}_{SP,\lambda}| - 1) \cdot |\mathcal{A}_{SP,\lambda-1}|$. They are randomly taken from a normal distribution with a mean of $0.1 \cdot (|\mathcal{A}_{SP,\lambda}| - 1) \cdot |\mathcal{A}_{SP,\lambda-1}|$ and a standard deviation of $0.1 \cdot (|\mathcal{A}_{SP,\lambda}| - 1) \cdot |\mathcal{A}_{SP,\lambda-1}|$.

Next, *multi-tier* connections (in this example, from tier $\lambda = 2$ to the customer agents) are generated, relative to all possible additional *multi-tier* connections $(|\mathcal{A}_{SP,2}| - 1) \cdot |\mathcal{A}_C|$. They are created randomly from a normal distribution with a mean of $0.01 \cdot (|\mathcal{A}_{SP,2}| - 1) \cdot |\mathcal{A}_C|$ and a standard deviation of $0.05 \cdot (|\mathcal{A}_{SP,2}| - 1) \cdot |\mathcal{A}_C|$. In general, mean and standard deviation for multi-tier connection probabilities decrease with distance of the tiers involved as follows. Let $\lambda, \lambda' \in \Lambda$ be the tiers for which multi-tier connections are to be generated with $\lambda > \lambda' \Rightarrow \lambda \geq \lambda' + 2$, since these are concerning multi-tier connections. Then, the probability of multi-tier connections from λ to λ' is a normal distribution with a mean of $0.01 \cdot (|\mathcal{A}_{SP,\lambda}| - 1) \cdot |\mathcal{A}_{SP,\lambda'}|/2^{(\lambda-\lambda'-2)}$ and a standard deviation of $0.05 \cdot (|\mathcal{A}_{SP,\lambda}| - 1) \cdot |\mathcal{A}_{SP,\lambda'}|/(\lambda - \lambda' - 1)$. Figure 4.2 illustrates the example of the SN creation process for additional direct and multi-tier edges. If a SN is instantiated in which one or more customer agents are not connected to

any SP agent, an additional edge is randomly generated to connect the customer agent to the SN.

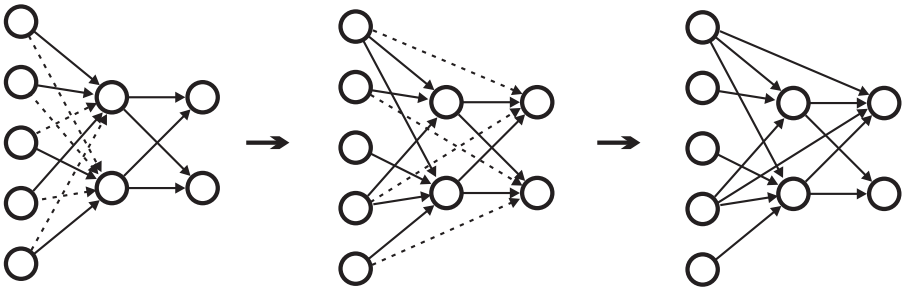

Figure 4.2: SN creation process example for additional and multi-tier edges.

This study includes fifty experiment setups with different random parameter values for each permutation of the deterministic parameter values, along with a corresponding SN topology. That is, fifty experiment setups are generated with these parameter values and SN topology with distinct random values for w_{ij}, c_{ij}, and $W_j \forall i, j$ for each experiment instance. The resulting 100 000 experiment instances are executed with fifty time periods (rounds) each, i.e., $\mathcal{T} = \{0, \ldots, 49\}$. Thus, customer agents are able to re-request the provision of services from SP agents if the demand is not fulfilled. SP agents can conclude contracts with the largest expected utility first, while they can propose to provide services to other customer agents in a subsequent time period of the experiment instance if sufficient capacity remains unallocated. The SP agents do not bid concurrently to several customer agents in one time period of an experiment instance to avoid unaccomplishable contracts. Instead, SP agents propose to provide the requested services with the highest required resource utilization, i.e., the highest expected SP utility. This research has conducted all experiments with the three different SP agent bidding policies described. After each experiment, the social welfare of the simulation's final allocation is calculated.

4.3.2 Results

4.3.2.1 Over all experiments

Table 4.3 shows mean, median, and standard deviation (sd) as a function of the number of customer agents over all experiments. In addition, it shows the difference (Δ) from the previous experiment (line) for median, mean, and standard deviation.

In general, the utility ratio decreases with an increasing number of customer agents. However, the decrease flattens for an increased number of customer

agents and the median is still above 0.82 for the experiments with 100 customers. This means that more than 50% of the experiments arrive at or above an efficiency of 0.82. The standard deviation lies between 0.117 and 0.148 over all experiments.

Table 4.3: Utility ratio as function of number of customer agents.

$\lvert\mathcal{A}_C\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
5	0.8947	0.8550	0.1471	-	-	-
10	0.8505	0.8261	0.1349	0.0443	0.0289	0.0122
15	0.8377	0.8183	0.1256	0.0128	0.0078	0.0093
20	0.8308	0.8130	0.1207	0.0069	0.0054	0.0049
25	0.8235	0.8035	0.1202	0.0073	0.0094	0.0005
30	0.8271	0.8064	0.1201	0.0036	0.0029	0.0001
35	0.8276	0.8071	0.1176	0.0005	0.0007	0.0026
40	0.8225	0.8015	0.1199	0.0051	0.0056	0.0023
45	0.8242	0.8021	0.1244	0.0017	0.0007	0.0045
50	0.8307	0.8052	0.1232	0.0065	0.0031	0.0012
55	0.8232	0.8025	0.1234	0.0076	0.0027	0.0002
60	0.8271	0.8007	0.1314	0.0039	0.0018	0.0080
65	0.8291	0.8038	0.1280	0.0021	0.0031	0.0034
70	0.8276	0.8065	0.1249	0.0015	0.0027	0.0031
75	0.8361	0.8095	0.1280	0.0085	0.0030	0.0031
80	0.8333	0.8109	0.1270	0.0028	0.0014	0.0010
85	0.8323	0.8115	0.1242	0.0010	0.0006	0.0028
90	0.8313	0.8088	0.1289	0.0010	0.0027	0.0047
95	0.8333	0.8121	0.1234	0.0020	0.0033	0.0055
100	0.8274	0.8087	0.1231	0.0059	0.0034	0.0003
$[5, 100]$	0.8326	0.8107	0.1265	-	-	-

Figure 4.3 shows the utility ratio as a function of the number of customer agents over all experiments. In the box plots, the box includes 50% of the data points, i.e., it limits the upper and lower quartiles. This means that 25% of the data points lie above the box, and 25% lie below it, where the line inside the box represents the median. For figure 4.3, this means that over all experiments and all number of customers, more than 75% of the experiments arrive at an efficiency above 0.7. In all experiments with more than five customers, more than 75% of the experiments arrive at an efficiency below 0.95.

The 'whiskers' (vertical lines) above and below the box are at most 1.5 times the inter-quartile range (distance between the first and third quartiles). Outliers are marked as single dots below the whiskers. Although the extreme outliers arrive at an efficiency below 0.2, their number is too small to have a distinct impact on the mean values.

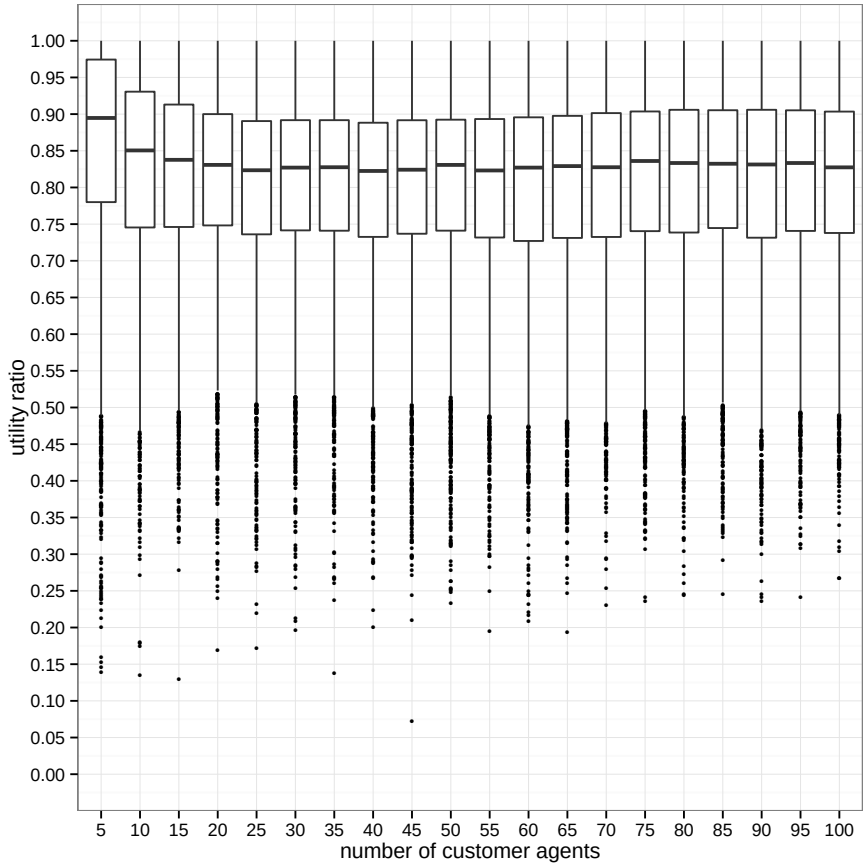

Figure 4.3: Utility ratio as function of number of customer agents.

For SNs with more than ten customers, the protocol implementation's efficiency is quasi-constant. Therefore, it can be concluded that the protocol implementation is scalable regarding the number of customers.

Table 4.4 shows mean, median, and standard deviation as a function of the number of SP agent tiers over all experiments. Additionally, it shows the difference (Δ) from the previous experiment (line) for median, mean, and standard deviation. The median efficiency of the protocol is decreasing for an increased number of tiers. The median utility ratio values lie in the interval $[0.8, 0.851]$ over all experiments. In contrast to the number of customers, the number of SP agent tiers has a quasi-linear effect on the protocol implementation's efficiency, i.e., efficiency is decreasing with an increasing number of SP agent tiers. However, the median of the protocol implementation's efficiency is still 0.8 for the experiments with five SP agent tiers. That is, more than 50% of the experiments with five SP agent tiers yield an efficiency at or above 0.8. The standard deviation lies between 0.108 and 0.138 over all experiments.

Table 4.4: Utility ratio as function of number of SP agent tiers.

$\|\Lambda\|$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
1	0.8471	0.8379	0.1088	-	-	-
2	0.8506	0.8298	0.1150	0.0036	0.0081	0.0062
3	0.8351	0.8119	0.1259	0.0155	0.0179	0.0109
4	0.8186	0.7950	0.1335	0.0165	0.0169	0.0076
5	0.8012	0.7787	0.1374	0.0174	0.0163	0.0039
$[1, 5]$	0.8326	0.8107	0.1265	-	-	-

Figure 4.4 shows the utility ratio as a function of the number of SP agent tiers over all experiments. Over all experiments and one to four SP agent tiers, more than 75% of the experiments arrive at an efficiency above 0.7, while for five SP agent tiers, the third quartile ends slightly below 0.7. In all experiments, more than 75% of the experiments arrive at an efficiency below 0.95.

In the experiments with one SP agent tier, there are very few outliers beyond the whisker, but the worst efficiency lies around 0.5 for these experiments. Since in this case there is only competition between SPs on one single tier, inefficient allocations cannot get worse by additional inefficient allocation in the next tier. The median is decreasing with the number of SP agent tiers and also, the third quartile (lower part of the box) is wider for an increased number of tiers, i.e., the standard deviation, especially with regard to lower efficiency values, increases with the number of SP agent tiers.

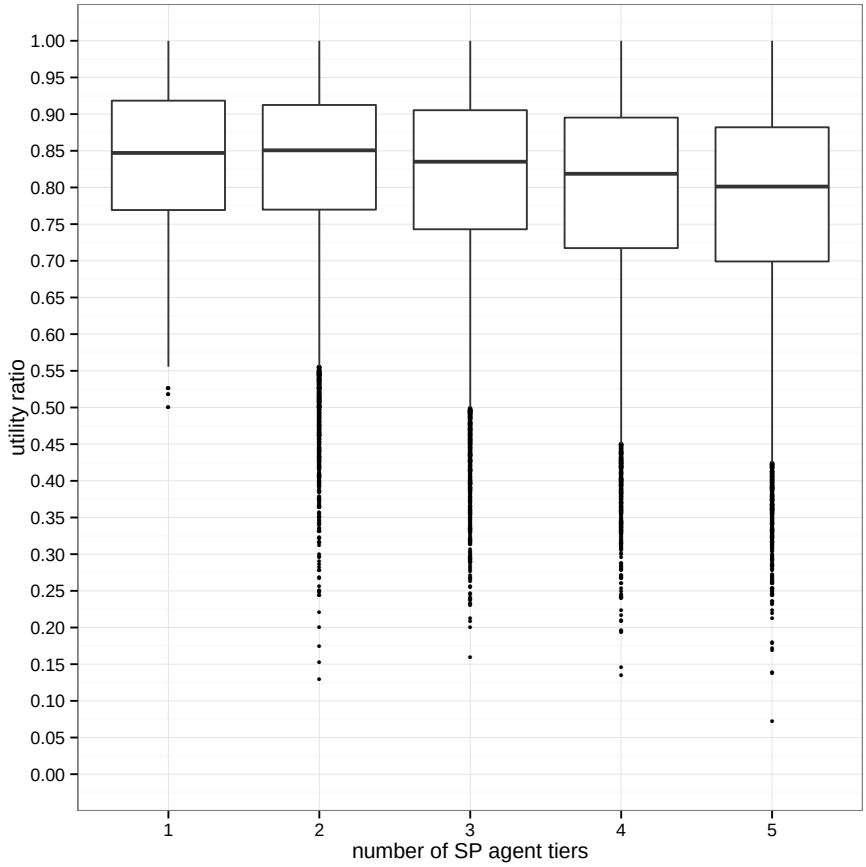

Figure 4.4: Utility ratio as function of number of SP agent tiers.

Table 4.5 shows mean, median, and standard deviation as a function of the number of SPs in tier 1. In addition, it shows the difference (Δ) from the previous experiment (line) for median, mean, and standard deviation. The median efficiency of the protocol is increasing for an increased number of SP agents in tier 1. The median utility ratio values lie in the interval $[0.795, 0.849]$ over all experiments. Inverse to the number of SP agent tiers, the number of SP agent in tier 1 has a quasi-linear positive effect on the protocol implementation's efficiency, i.e., efficiency is monotonically increasing with an increasing number of SP agents in tier 1. The median of the protocol implementation's efficiency is above 0.82 for the experiments with more than two SP agents in the first tier. That is, more than 50% of the experiments with more than two SP agents on the first tier yield an efficiency above 0.82.

The standard deviation lies between 0.097 and 0.164 over all experiments, and is considerably higher for two SP agents in the first tier. This results from a low SP-customer-ratio for these experiments and therefore a worse utilization

of the resources, due to the myoptic allocation strategy of the SP agents to allocate resources for agents with higher demand first.

Table 4.5: Utility ratio as function of the number of SPs in tier 1.

$\lvert\mathcal{A}_{SP,1}\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
2	0.7959	0.7724	0.1636	-	-	-
4	0.8217	0.8007	0.1309	0.0258	0.0283	0.0327
6	0.8333	0.8169	0.1144	0.0117	0.0162	0.0165
8	0.8426	0.8278	0.1048	0.0093	0.0108	0.0096
10	0.8483	0.8354	0.0976	0.0057	0.0077	0.0072
$[2, 10]$	0.8326	0.8107	0.1265	-	-	-

Figure 4.5 shows the utility ratio as a function of the number of SPs in tier 1. Over all experiments and four to ten SP agents in tier 1, more than 75% of the experiments arrive at an efficiency above 0.7, while for two SP agents in the first tier, the third quartile ends above 0.65. In all experiments, 25% of the experiments arrive at an efficiency above 0.9.

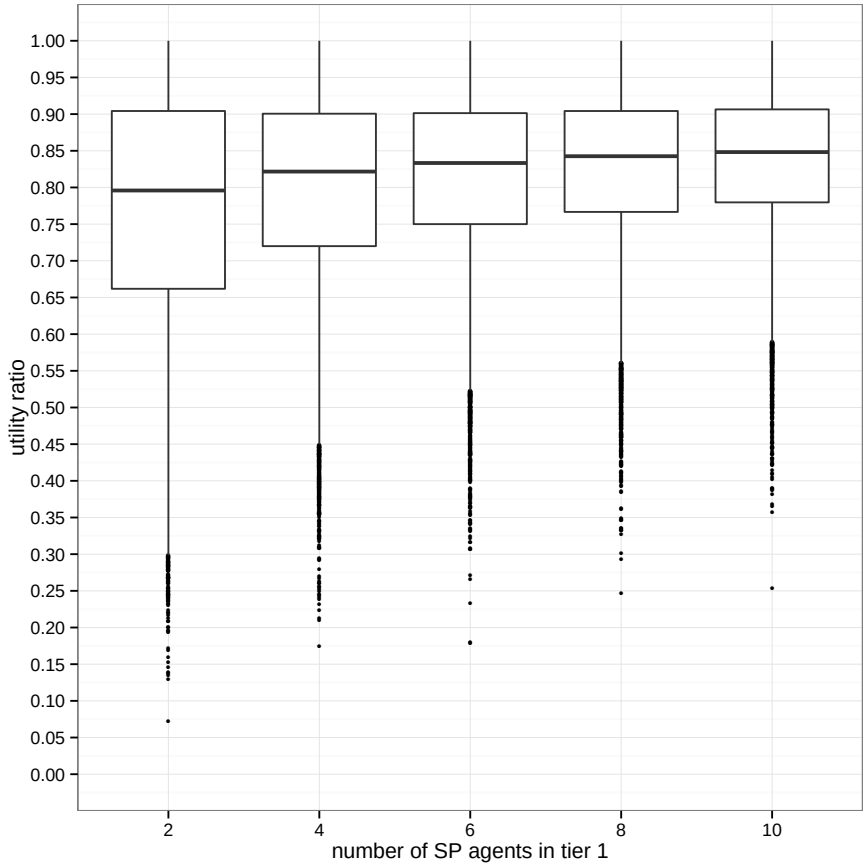

Figure 4.5: Utility ratio as function of number of SP agents in tier 1.

Table 4.6 shows the results on a linear regression analysis over all experiments. It contains the utility ratio as the dependent variable, and shows the influence of the number of customer agents ($|\mathcal{A}_C|$), the number of SP agent tiers ($|\Lambda|$), the number of SP agents in tier 1 ($|\mathcal{A}_{SP,1}|$), the number of SP agents ($|\mathcal{A}_{SP}|$), the number of total edges of the SN, and the demand-supply-ratio of the SN on the utility ratio.

All mentioned variables have a significant influence on the utility ratio. The number of customers has a negative effect on the utility ratio, and therewith the protocol implementation's efficiency; since the search space is increased, the probability of allocating resources suboptimally is also increased. Similarly, the number of SP agent tiers and the total number of SP agents have negative effects on the efficiency. In contrast, the number of SP agents in the first tier positively influences the utility ratio. The number of total edges also has a positive effect on the utility ratio, while the demand-supply-ratio has a negative effect.

Table 4.6: Linear regression analysis results.

	Dependent variable:
	UTILITY_RATIO
$\lvert\mathcal{A}_C\rvert$	-0.001^{***}
	(0.00003)
$\lvert\Lambda\rvert$	-0.011^{***}
	(0.001)
$\lvert\mathcal{A}_{SP,1}\rvert$	0.006^{***}
	(0.0002)
$\lvert\mathcal{A}_{SP}\rvert$	-0.003^{***}
	(0.0001)
TOTAL_EDGES	0.001^{***}
	(0.00002)
DEMAND_SUPPLY_RATIO	-0.003^{***}
	(0.0002)
Constant	0.852^{***}
	(0.002)
Observations	$100,000$
R^2	0.079
Adjusted R^2	0.079
Residual Std. Error	$0.121(df = 99993)$
F statistic	$1,432.620^{***}(df = 6; 99993)$
Note:	*p<0.1; **p<0.05; ***p<0.01

4.3.2.2 Non-substitutable resources

Table 4.7 shows the mean, median, and standard deviation values for the experiments with non-substitutable resources (NSR) as a function of the number of customer agents. It also shows the differences between the single experimental settings for these values. More than 50% of the experiments achieve an efficiency of above 0.81; the median is still above 0.81 for 100 customers. The standard deviation is 0.094 to 0.145. For an increasing number of customer agents, the utility ratio decreases for NSR until approximately 50 customer agents. With more customer agents, the utility ratio is quasi-constant.

Table 4.7: Utility ratio as function of number of customer agents for non-substitutable resources.

$\lvert\mathcal{A}_C\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
5	0.9347	0.9096	0.0944	-	-	-
10	0.9039	0.8843	0.0974	0.0308	0.0253	0.0029
15	0.8916	0.8705	0.1066	0.0123	0.0138	0.0093
20	0.8778	0.8554	0.1112	0.0138	0.0152	0.0046
25	0.8644	0.8388	0.1243	0.0134	0.0166	0.0131
30	0.8671	0.8294	0.1299	0.0027	0.0093	0.0056
35	0.8571	0.8287	0.1215	0.0100	0.0007	0.0084
40	0.8492	0.8193	0.1229	0.0079	0.0094	0.0014
45	0.8419	0.8082	0.1373	0.0073	0.0111	0.0144
50	0.8290	0.7986	0.1380	0.0129	0.0096	0.0007
55	0.8276	0.7984	0.1316	0.0014	0.0002	0.0064
60	0.8218	0.7900	0.1448	0.0057	0.0084	0.0132
65	0.8215	0.7891	0.1416	0.0004	0.0009	0.0033
70	0.8128	0.7880	0.1350	0.0087	0.0011	0.0066
75	0.8187	0.7889	0.1418	0.0060	0.0009	0.0068
80	0.8158	0.7881	0.1367	0.0029	0.0008	0.0051
85	0.8114	0.7872	0.1356	0.0044	0.0009	0.0011
90	0.8141	0.7883	0.1397	0.0027	0.0011	0.0041
95	0.8143	0.7859	0.1360	0.0002	0.0024	0.0037
100	0.8120	0.7911	0.1309	0.0023	0.0051	0.0051
$[5, 100]$	0.8464	0.8169	0.1336	-	-	-

Figure 4.6 shows the utility ratio as a function of the number of customer agents over all experiments with NSR. In all experiments with more than ten customers, more than 75% of the experiments yield an efficiency below 0.95. For five and ten customer agents, the efficiency is considerably higher: For five customers, almost 25% of the experiments yield the optimal result, while for ten customers, more than 25% still arrive at an efficiency above 0.95.

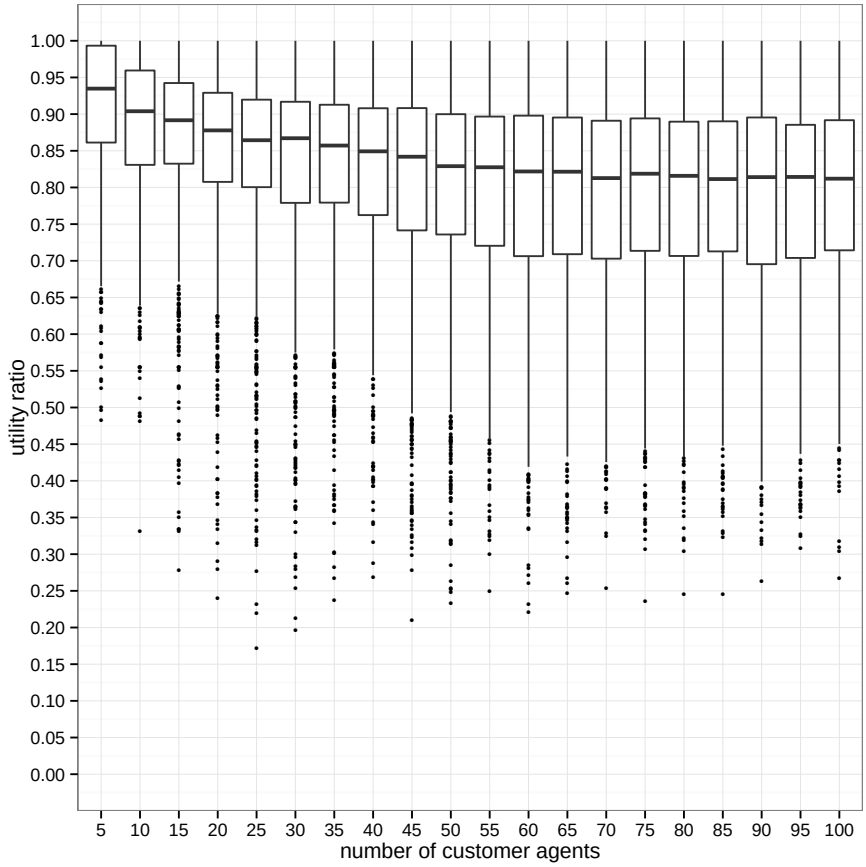

Figure 4.6: Utility ratio as function of number of customer agents for non-substitutable resources.

Table 4.8 shows the numerical results over all experiments with NSR as a function of the number of SP agent tiers. It shows mean, median, and standard deviation, as well as the differences from the previous experiment for these values. The median efficiency of the protocol is quasi-constant for an increased number of tiers and NSR; it lies in the interval $[0.838, 0.858]$, the standard deviation values are between 0.108 and 0.151. Since the resources are non-substitutable across tiers, the number of tiers does not have a considerable influence on the utility ratio.

The results for the number of SP agent tiers over all experiments with NSR are shown in figure 4.7. The protocol's efficiency in these experiments is quasi-constant, independent of the number of SP agent tiers. The experiments yield a median utility ratio between 0.83 and 0.86. The standard deviation increases with the number of SP agent tiers.

Table 4.8: Utility ratio as function of number of SP agent tiers for non-substitutable resources.

$\|\Lambda\|$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
1	0.8482	0.8380	0.1088	-	-	-
2	0.8574	0.8312	0.1224	0.0092	0.0068	0.0136
3	0.8440	0.8132	0.1346	0.0134	0.0180	0.0122
4	0.8426	0.8056	0.1426	0.0014	0.0076	0.0080
5	0.8381	0.7964	0.1510	0.0045	0.0092	0.0084
$[1, 5]$	0.8464	0.8169	0.1336	-	-	-

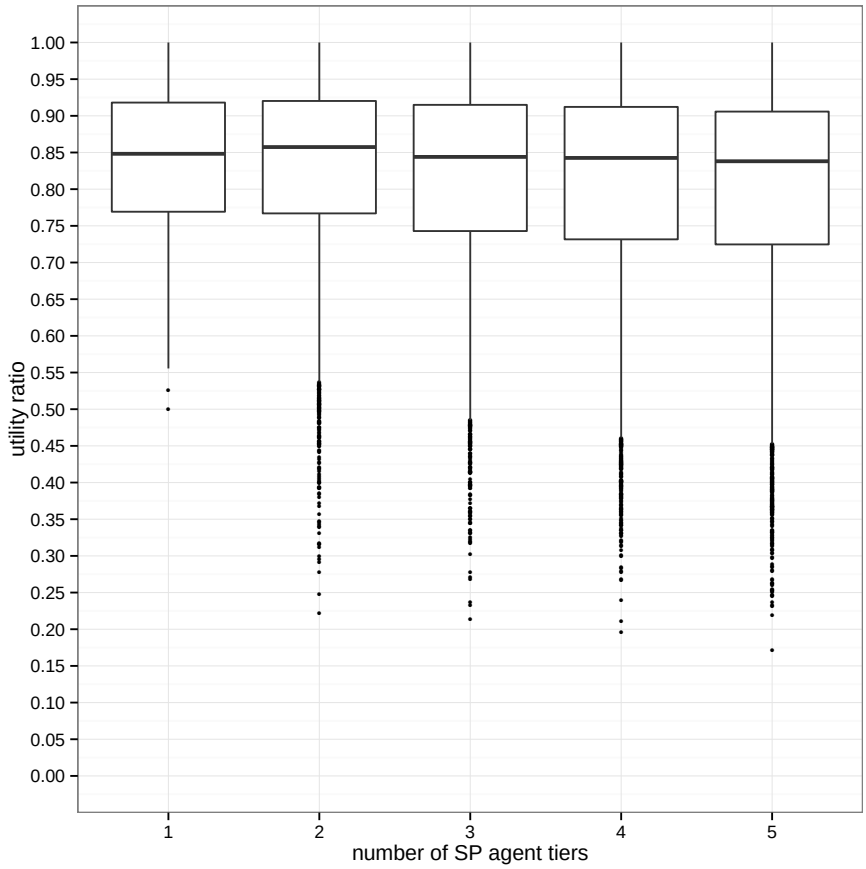

Figure 4.7: Utility ratio as function of number of SP agent tiers for non-substitutable resources.

Table 4.9 shows mean, median, and standard deviation as a function of the number of SPs in tier 1 for NSR. It also shows the differences from the previous experiment for these values. The median utility ratio values lie in the interval $[0.761, 0.873]$ for the experiments with NSR; they are increasing with the number of SP agents in tier 1. The protocol implementation's efficiency is monotonically increasing with an increasing number of SP agents in tier 1: The number of SP agents in tier 1 has a quasi-linear effect. More than 50% of the experiments with more than two SP agent tiers yield an efficiency above 0.82, i.e., the median of the protocol implementation's efficiency is above 0.82 for the experiments with more than two SP agents in the first tier.

The standard deviation is considerably higher for two SP agents in the first tier; it lies between 0.094 and 0.173 over all experiments with NSR. Due to the myoptic allocation strategy of the SP agents to allocate resources for agents with higher demand first and, in addition, a low SP-customer-ratio for two SP agents in the first tier, this results in a worse utilization of the resources.

Table 4.9: Utility ratio as function of the number of SPs in tier 1 for non-substitutable resources.

$\lvert\mathcal{A}_{SP,1}\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
2	0.7610	0.7468	0.1724	-	-	-
4	0.8289	0.8037	0.1370	0.0679	0.0569	0.0354
6	0.8519	0.8295	0.1176	0.0230	0.0258	0.0194
8	0.8667	0.8482	0.1015	0.0148	0.0187	0.0161
10	0.8726	0.8563	0.0943	0.0059	0.0081	0.0072
$[2, 10]$	0.8464	0.8169	0.1336	-	-	-

The utility ratio is shown as a function of the number of SPs in tier 1 for NSR in figure 4.8. Over all experiments with NSR and four to ten SP agents in tier 1, more than 75% of the experiments arrive at an efficiency above 0.7, while for two SP agents in the first tier, the third quartile ends in the interval $[0.6, 0.65]$. In all experiments, almost 25% of the experiments arrive at an efficiency above 0.9.

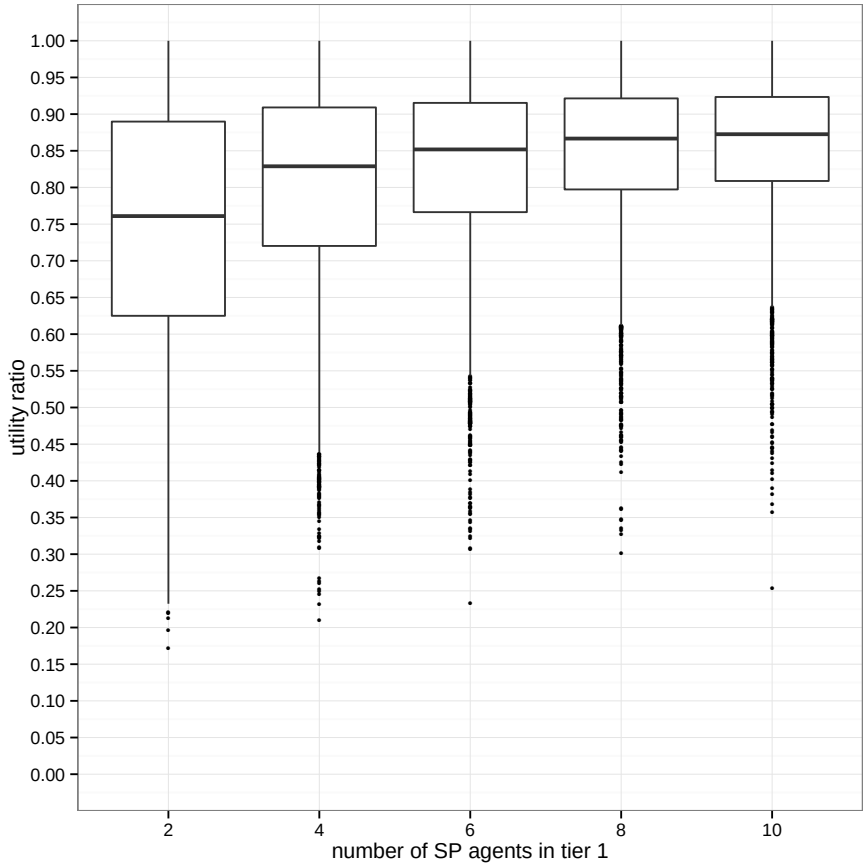

Figure 4.8: Utility ratio as function of number of SP agents in tier 1 for non-substitutable resources.

Table 4.10 shows the results on a linear regression analysis over all experiments for non-substitutable resources. In contains the utility ratio as the dependent variable, and shows the influence of the number of customer agents ($|\mathcal{A}_C|$), the number of SP agent tiers ($|\Lambda|$), the number of SP agents in tier 1 ($|\mathcal{A}_{SP,1}|$), the number of SP agents ($|\mathcal{A}_{SP}|$), the number of total edges of the SN, and the demand-supply-ratio of the SN on the utility ratio.

The number of SP agent tiers does not have a significant influence on the utility ratio, since resources are non-substitutable between SN tiers. All other variables have a significant influence on the utility ratio. The number of customers has a negative effect on the utility ratio, and therewith the protocol implementation's efficiency, since the search space is increased and the probability of allocating resources suboptimally is also increased. Similarly, the total number of SP agents has a negative effect on the efficiency. In contrast, the number of SP agents in the first tier positively influences the utility ratio. The

number of total edges and the demand-supply-ratio also have a positive effect on the utility ratio.

Table 4.10: Linear regression analysis results for non-substitutable resources.

	Dependent variable:
	UTILITY_RATIO
$\lvert\mathcal{A}_C\rvert$	-0.002^{***}
	(0.0001)
$\lvert\Lambda\rvert$	-0.002
	(0.001)
$\lvert\mathcal{A}_{SP,1}\rvert$	0.013^{***}
	(0.0004)
$\lvert\mathcal{A}_{SP}\rvert$	-0.002^{***}
	(0.0002)
TOTAL_EDGES	0.001^{***}
	(0.00003)
DEMAND_SUPPLY_RATIO	0.004^{***}
	(0.0004)
Constant	0.829^{***}
	(0.004)
Observations	25,000
R^2	0.164
Adjusted R^2	0.164
Residual Std. Error	$0.122(df = 24993)$
F statistic	$818.715^{***}(df = 6; 24993)$
Note:	*p<0.1; **p<0.05; ***p<0.01

4.3.2.3 Substitutable resources

In Cloud computing, equivalent generic services (e.g., IaaS) can be produced by different SPs. That is, the resources to provide these services to customers are substitutable between SPs. For the composition of the SP agents' bids for these services, this research investigates three different bidding policies: best price resources only (*BPRO*), external resources first (*ERF*), and internal resources first (*IRF*). The results of the simulation study for substitutable resources are presented in this section.

4.3.2.3.1 *BPRO* bidding policy Table 4.11 shows mean, median, and standard deviation as a function of the number of customer agents over all ex-

periments for substitutable resources for the *BPRO* bidding policy. Additionally, it shows the differences from the previous experiment for median, mean, and standard deviation.

In general, the utility ratio decreases with an increasing number of customer agents. However, the decrease flattens for an increased number of customer agents and the median is still above 0.75 for the experiments with 100 customers. That is, more than 50% of the experiments arrive at or above an efficiency of 0.75. The standard deviation lies between 0.105 and 0.138 over all experiments for substitutable resources and the *BPRO* bidding policy.

Table 4.11: Utility ratio as function of number of customer agents for substitutable resources and *BPRO* bidding policy.

$\lvert\mathcal{A}_C\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
5	0.9254	0.9013	0.1055	-	-	-
10	0.8710	0.8508	0.1173	0.0544	0.0505	0.0118
15	0.8452	0.8275	0.1203	0.0258	0.0233	0.0031
20	0.8324	0.8155	0.1187	0.0128	0.0120	0.0017
25	0.8065	0.7865	0.1243	0.0259	0.0289	0.0056
30	0.8060	0.7857	0.1244	0.0005	0.0008	0.0001
35	0.7975	0.7778	0.1241	0.0084	0.0079	0.0003
40	0.7823	0.7632	0.1319	0.0152	0.0146	0.0078
45	0.7767	0.7578	0.1354	0.0056	0.0054	0.0035
50	0.7861	0.7624	0.1320	0.0094	0.0047	0.0034
55	0.7691	0.7517	0.1323	0.0170	0.0107	0.0004
60	0.7659	0.7476	0.1378	0.0032	0.0041	0.0054
65	0.7735	0.7558	0.1300	0.0076	0.0083	0.0078
70	0.7696	0.7557	0.1281	0.0039	0.0001	0.0019
75	0.7738	0.7566	0.1340	0.0042	0.0008	0.0059
80	0.7818	0.7645	0.1323	0.0080	0.0079	0.0018
85	0.7792	0.7646	0.1267	0.0026	0.0001	0.0056
90	0.7694	0.7566	0.1321	0.0098	0.0080	0.0055
95	0.7783	0.7675	0.1244	0.0089	0.0109	0.0077
100	0.7815	0.7667	0.1250	0.0033	0.0008	0.0005
$[5, 100]$	0.7983	0.7808	0.1326	-	-	-

The utility ratio is shown as a function of the number of customer agents over all experiments for substitutable resources for the *BPRO* bidding policy in figure 4.9. More than 75% of the experiments arrive at an efficiency above 0.65. In all experiments with more than five customers, more than 75% of the experiments arrive at an efficiency below 0.95. For five customers, 25% of the experiments yield the optimal utility.

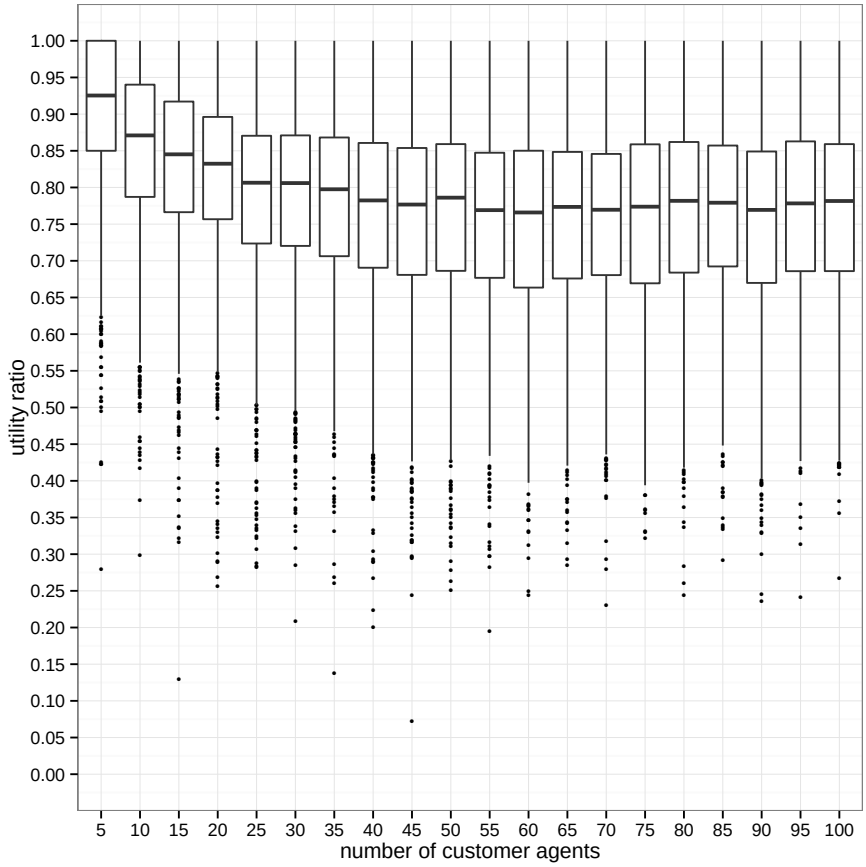

Figure 4.9: Utility ratio as function of number of customer agents for substitutable resources and *BPRO* bidding policy.

Table 4.12 shows mean, median, and standard deviation as a function of the number of SP agent tiers over all experiments for substitutable resources and the *BPRO* bidding policy, as well as the differences from the previous experiment for the mentioned values. The standard deviation is in the interval $[0.109, 0.137]$. The median efficiency of the protocol lies in the interval $[0.766, 0.847]$ and is decreasing for an increased number of tiers.

The utility ratio as a function of the number of SP agent tiers over all experiments for substitutable resources and the *BPRO* bidding policy is shown in figure 4.10. For one and two SP agent tiers, more than 75% of the experiments arrive at an efficiency above 0.7, while for four and five SP agent tiers, the third quartile ends slightly above 0.65. In all experiments, more than 75% of the experiments arrive at an efficiency below 0.95.

Table 4.12: Utility ratio as function of number of SP agent tiers for substitutable resources and *BPRO* bidding policy.

$\|\Lambda\|$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
1	0.8466	0.8378	0.1090	-	-	-
2	0.8108	0.7946	0.1266	0.0358	0.0432	0.0176
3	0.7876	0.7702	0.1323	0.0232	0.0245	0.0058
4	0.7716	0.7541	0.1358	0.0160	0.0161	0.0035
5	0.7661	0.7473	0.1369	0.0055	0.0069	0.0010
$[1, 5]$	0.7983	0.7808	0.1326	-	-	-

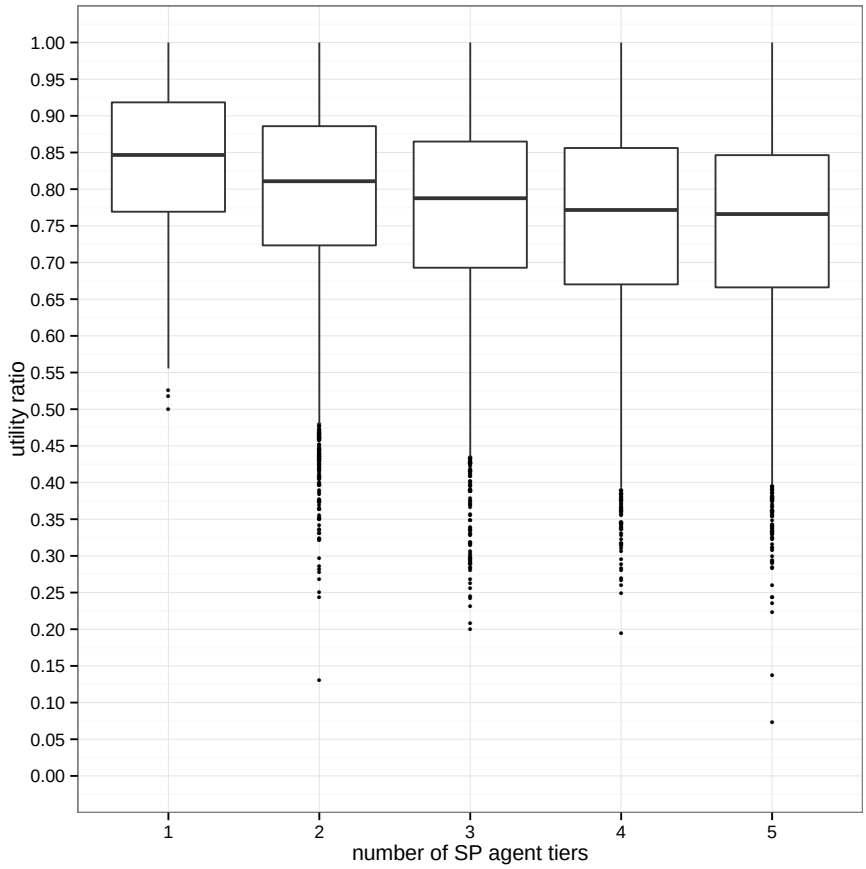

Figure 4.10: Utility ratio as function of number of SP agent tiers for substitutable resources and *BPRO* bidding policy.

Table 4.13 shows the results of the experiments for substitutable resources and the *BPRO* bidding policy. It shows mean, median, and standard deviation as a function of the number of SPs in tier 1, along with the differences from the previous experiment. The median efficiency of the protocol is increasing with the number of SP agents in tier 1 and lies in the interval $[0.743, 0.826]$. The median of the protocol implementation's efficiency is above 0.779 for the experiments with more than two SP agents in the first tier. That is, more than half of the experiments with more than two SP agent tiers yield an efficiency above 0.779. The number of SP agents in tier 1 has a quasi-linear positive effect on the protocol implementation's efficiency; it is monotonically increasing with an increasing number of SP agents in tier 1, inverse to the number of SP agent tiers.

The standard deviation is considerably higher for two SP agents in the first tier. It lies between 0.096 and 0.172 for these experiments. This results from a low ratio of SP to customer agents for these experiments, and therefore a worse utilization of the resources, due to the allocation strategy of the SP agents to allocate resources to agents with higher demand first.

Table 4.13: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *BPRO* bidding policy.

$\lvert\mathcal{A}_{SP,1}\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
2	0.7437	0.7347	0.1715	-	-	-
4	0.7790	0.7636	0.1385	0.0353	0.0289	0.0330
6	0.7985	0.7882	0.1195	0.0195	0.0246	0.0189
8	0.8119	0.8015	0.1078	0.0134	0.0133	0.0117
10	0.8256	0.8160	0.0968	0.0137	0.0145	0.0110
$[2, 10]$	0.7983	0.7808	0.1326	-	-	-

Figure 4.11 shows the utility ratio as a function of the number of SPs in tier 1 for substitutable resources for the *BPRO* bidding policy. Over all experiments and six to ten SP agents in tier 1, more than 75% of the experiments arrive at an efficiency above 0.7, while for two to four SP agents in the first tier, the third quartile ends below 0.7. In all experiments, more than 25% of the experiments arrive at an efficiency above 0.85.

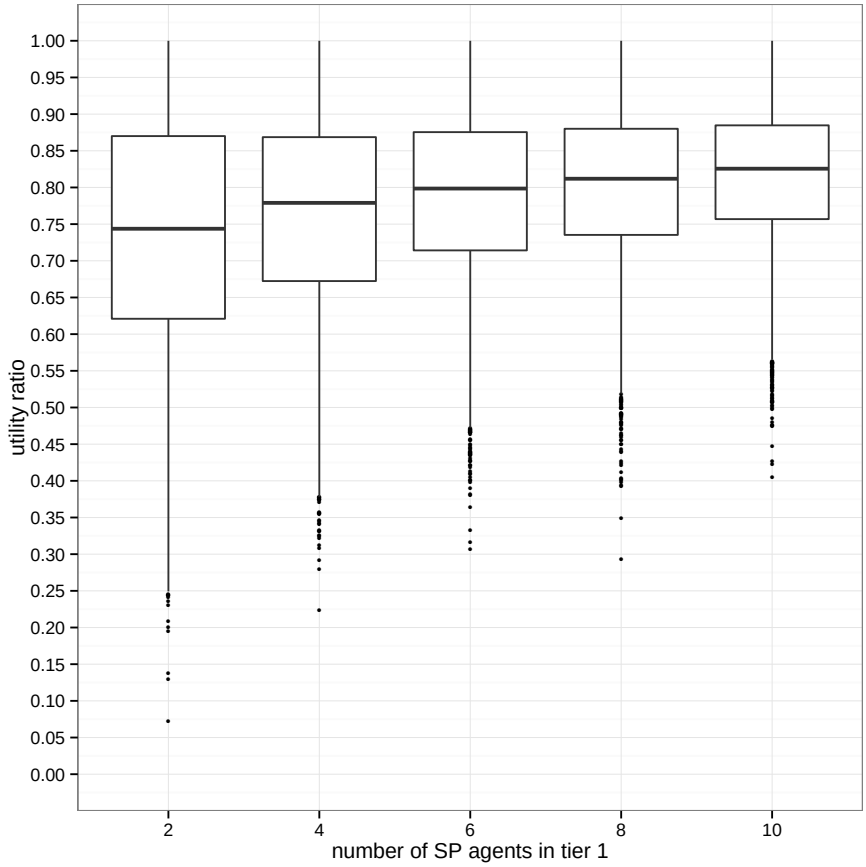

Figure 4.11: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *BPRO* bidding policy.

The results of a linear regression analysis for substitutable resources and the *BPRO* bidding policy are shown in table 4.14. In contains the utility ratio as the dependent variable. The influence of the number of customer agents ($|\mathcal{A}_C|$), the number of SP agent tiers ($|\Lambda|$), the number of SP agents in tier 1 ($|\mathcal{A}_{SP,1}|$), the number of SP agents ($|\mathcal{A}_{SP}|$), the number of total edges of the SN, and the demand-supply-ratio of the SN on the utility ratio is analyzed.

All mentioned variables significantly influence the efficiency. The number of customers has a negative effect on the utility ratio. This results from the increased search space and the increased probability of suboptimal allocation of resources. The number of SP agent tiers and the total number of SP agents also have negative effects on the efficiency. Contrarily, the number of SP agents in the first tier positively influences the utility ratio. The number of total edges and the demand-supply-ratio also have a positive effect on the utility ratio.

Table 4.14: Linear regression analysis results for for substitutable resources and *BPRO* bidding policy.

	Dependent variable:
	UTILITY_RATIO
$\lvert\mathcal{A}_C\rvert$	-0.002^{***}
	(0.0001)
$\lvert\Lambda\rvert$	-0.019^{***}
	(0.001)
$\lvert\mathcal{A}_{SP,1}\rvert$	0.009^{***}
	(0.0004)
$\lvert\mathcal{A}_{SP}\rvert$	-0.002^{***}
	(0.0002)
TOTAL_EDGES	0.001^{***}
	(0.00003)
DEMAND_SUPPLY_RATIO	0.004^{***}
	(0.0004)
Constant	0.847^{***}
	(0.004)
Observations	25,000
R^2	0.164
Adjusted R^2	0.164
Residual Std. Error	$0.121 (df = 24993)$
F statistic	$818.559^{***} (df = 6; 24993)$
Note:	*p<0.1; **p<0.05; ***p<0.01

4.3.2.3.2 *ERF* bidding policy Table 4.15 shows the experiment results as a function of the number of customer agents for substitutable resources and the *ERF* bidding policy. These numerical results include mean, median, and standard deviation, along with the differences from the previous experiment.

For more than five customer agents, the utility ratio increases with the number of customer agents. The median is still above 0.83 for the experiments with 10 customers; i.e., more than 50% of the results are at or above an efficiency of 0.83 for these experiments. The increase flattens for more than 50 customer agents and is quasi-constant for more than 75 customer agents. The standard deviation is in the interval $[0.083, 0.132]$.

Figure 4.12 shows the utility ratio as a function of the number of customer agents over all experiments for substitutable resources and the *ERF* bidding policy. More than 75% of the experiments achieve an efficiency above 0.7.

Table 4.15: Utility ratio as function of number of customer agents for substitutable resources and *ERF* bidding policy.

$\lvert\mathcal{A}_C\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
5	0.8704	0.8441	0.1316	-	-	-
10	0.8309	0.8233	0.1152	0.0394	0.0208	0.0163
15	0.8316	0.8248	0.1004	0.0007	0.0015	0.0148
20	0.8305	0.8244	0.0953	0.0011	0.0003	0.0051
25	0.8348	0.8304	0.0878	0.0043	0.0060	0.0075
30	0.8445	0.8408	0.0850	0.0097	0.0104	0.0028
35	0.8502	0.8425	0.0876	0.0057	0.0017	0.0026
40	0.8546	0.8446	0.0856	0.0044	0.0021	0.0020
45	0.8614	0.8553	0.0831	0.0068	0.0107	0.0026
50	0.8738	0.8623	0.0832	0.0124	0.0070	0.0001
55	0.8705	0.8592	0.0903	0.0033	0.0031	0.0072
60	0.8780	0.8635	0.0944	0.0075	0.0044	0.0040
65	0.8858	0.8685	0.0944	0.0078	0.0050	0.0000
70	0.8915	0.8722	0.0929	0.0058	0.0037	0.0015
75	0.8931	0.8760	0.0928	0.0015	0.0038	0.0002
80	0.8971	0.8753	0.1001	0.0040	0.0007	0.0073
85	0.8992	0.8784	0.0959	0.0022	0.0032	0.0041
90	0.8983	0.8769	0.0997	0.0010	0.0016	0.0037
95	0.8989	0.8759	0.0994	0.0007	0.0010	0.0003
100	0.8969	0.8694	0.1029	0.0020	0.0065	0.0036
$[5, 100]$	0.8697	0.8554	0.0983	-	-	-

Except for the experiments with ten customer agents, which show the worst efficiency for the *ERF* bidding policy, the third quartile ends above 0.75. The median increases up to 75 customers and is quasi-constant for more costumer agents. For these experiments, almost 25% yield an efficiency at or above 0.95. In contrast, for 15–30 customers, more than 75% of the experiments arrive at a utility ratio below 0.9.

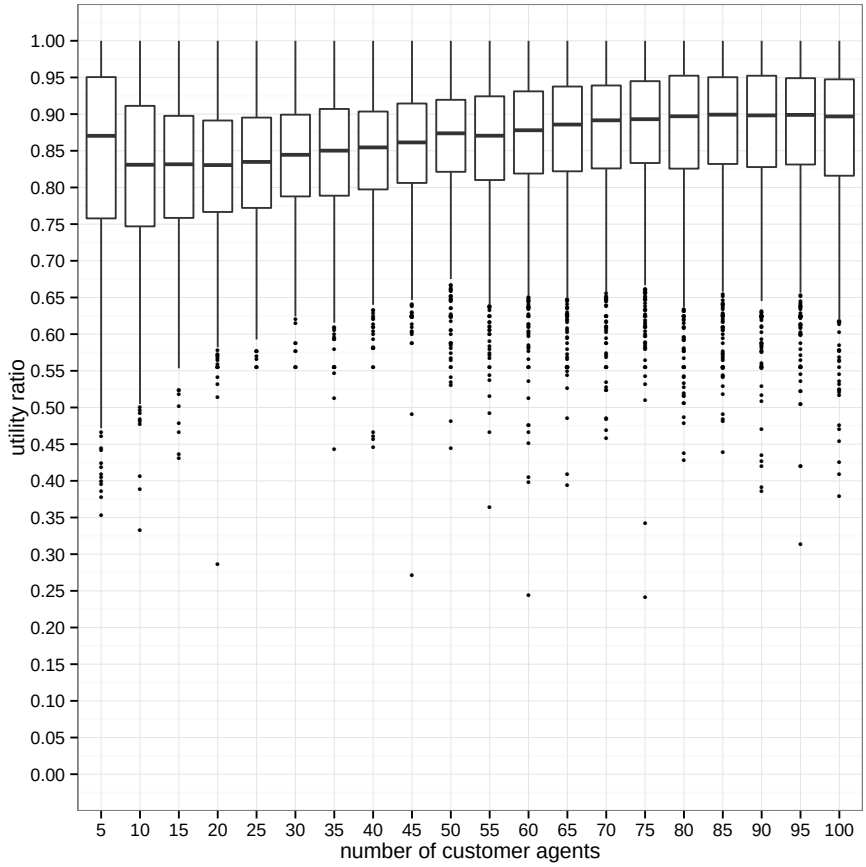

Figure 4.12: Utility ratio as function of number of customer agents for substitutable resources and *ERF* bidding policy.

The numerical results of the experiments for substitutable resources and the *ERF* bidding policy are shown in table 4.16 as a function of the number of SP agent tiers. The table shows mean, median, and standard deviation, as well as the differences between the experiment settings. The standard deviation lies between 0.091 and 0.109. The median efficiency of the protocol is decreasing for an increased number of tiers and the values arrive in the interval $[0.846, 0.886]$ over all experiments.

Figure 4.13 shows the utility ratio as a function of the number of SP agent tiers. In these experiments, more than 75% of the experiments arrive at an efficiency above 0.75, but below 0.95. It is notably higher for two and three SP agent tiers. There is a trade-off between more SP agents in total, due to more tiers, and forwarded requests along the complete SN to the last tier, due to the *ERF* bidding policy.

Table 4.16: Utility ratio as function of number of SP agent tiers for substitutable resources and *ERF* bidding policy.

$\|\Lambda\|$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
1	0.8462	0.8375	0.1085	-	-	-
2	0.8857	0.8690	0.0912	0.0396	0.0315	0.0173
3	0.8851	0.8681	0.0933	0.0006	0.0009	0.0021
4	0.8707	0.8579	0.0969	0.0143	0.0102	0.0036
5	0.8556	0.8446	0.0968	0.0152	0.0133	0.0001
$[1, 5]$	0.8697	0.8554	0.0983	-	-	-

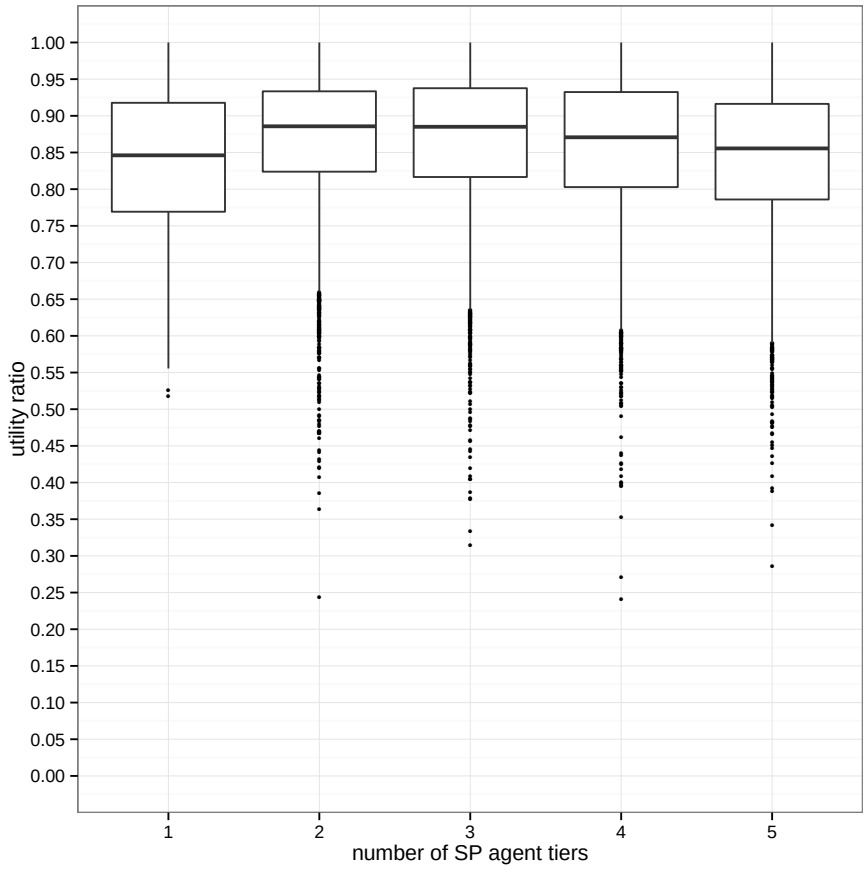

Figure 4.13: Utility ratio as function of number of SP agent tiers for substitutable resources and *ERF* bidding policy.

Table 4.17 shows mean, median, and standard deviation as a function of the number of SPs in tier 1 for substitutable resources for the *ERF* bidding policy. In addition, it shows the difference (Δ) from the previous experiment (line) for median, mean, and standard deviation. The median efficiency of the protocol is quasi-constant for six to ten SP agents in tier 1; the values lie in the interval $[0.861, 0.892]$ over all experiments and in $[0.861, 0.866]$ for six to ten SP agents in the first tier. The standard deviation lies between 0.085 and 0.124 and is considerably higher for two SP agents in the first tier. This results from a worse utilization of the resources, due to SP agents allocating resources for customer agents with higher demand first. In addition, the SP-customer-ratio is higher for an increased number of SP agents in tier 1.

Table 4.17: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *ERF* bidding policy.

$\lvert\mathcal{A}_{SP,1}\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
2	0.8916	0.8590	0.1239	-	-	-
4	0.8720	0.8548	0.1003	0.0197	0.0042	0.0236
6	0.8652	0.8552	0.0895	0.0068	0.0005	0.0108
8	0.8649	0.8541	0.0877	0.0003	0.0012	0.0018
10	0.8615	0.8539	0.0850	0.0034	0.0002	0.0027
$[2, 10]$	0.8697	0.8554	0.0983	-	-	-

The utility ratio is shown as a function of the number of SPs in tier 1 for substitutable resources and the *ERF* bidding policy in figure 4.14. Almost 75% of the experiments arrive at an efficiency above 0.8, and more than 25% of the experiments even arrive at an efficiency above 0.9. The standard deviation is considerably smaller for an increased number of SP agents in tier 1.

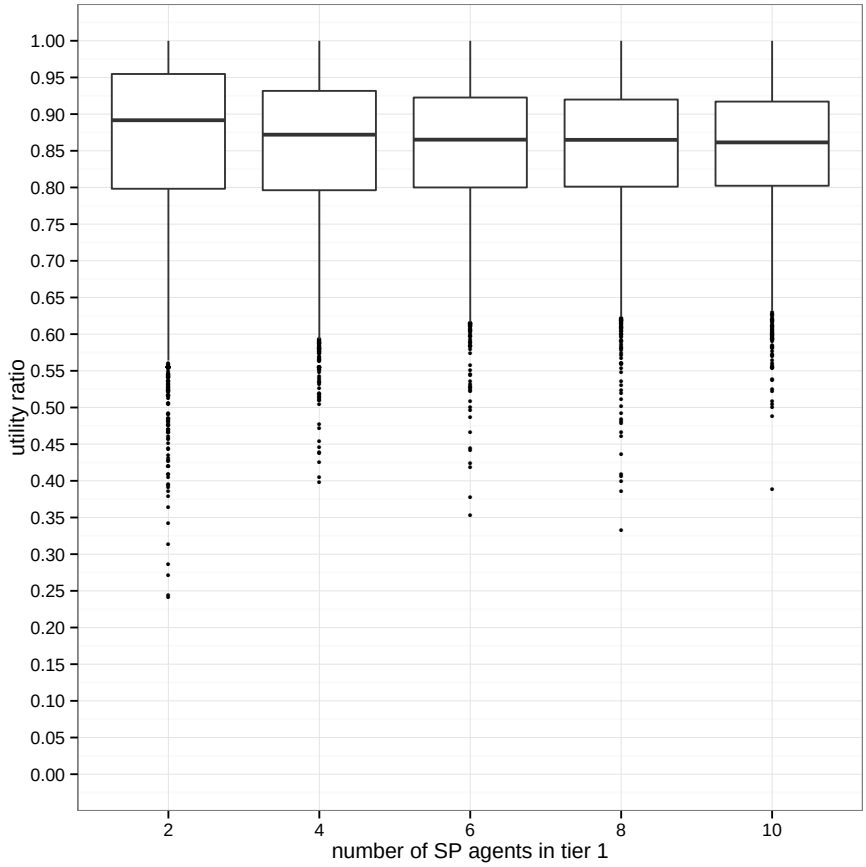

Figure 4.14: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *ERF* bidding policy.

Table 4.18 shows the results on a linear regression analysis over all experiments for substitutable resources for the *ERF* bidding policy. It shows the results of the analysis of the influence of the number of customer agents ($|\mathcal{A}_C|$), the number of SP agent tiers ($|\Lambda|$), the number of SP agents in tier 1 ($|\mathcal{A}_{SP,1}|$), the number of SP agents ($|\mathcal{A}_{SP}|$), the number of total edges of the SN, and the demand-supply-ratio of the SN on the utility ratio as the dependent variable.

All variables have a significant influence on the utility ratio. The number of customer agents has a positive effect on the efficiency. This results from the *ERF* bidding policy, since all intermediate SPs forward their requests to the next tier and thus all customer demand is cumulated at the last SP tier. Therefore, the search space for allocations is reduced to a single tier.

The number of total edges has a positive, the demand-supply-ratio a negative effect on the utility ratio. The number of SP agent tiers and the number of SP agents in the first tier also negatively influence the utility ratio, since

concurrency of forwarded requests occurs.

Table 4.18: Linear regression analysis results for for substitutable resources and *ERF* bidding policy.

	Dependent variable:
	UTILITY_RATIO
$\lvert\mathcal{A}_C\rvert$	0.0001^{***}
	(0.00004)
$\lvert\Lambda\rvert$	-0.005^{***}
	(0.001)
$\lvert\mathcal{A}_{SP,1}\rvert$	-0.005^{***}
	(0.0003)
$\lvert\mathcal{A}_{SP}\rvert$	-0.002^{***}
	(0.0002)
TOTAL_EDGES	0.001^{***}
	(0.00002)
DEMAND_SUPPLY_RATIO	-0.012^{***}
	(0.0003)
Constant	0.898^{***}
	(0.003)
Observations	25,000
R^2	0.112
Adjusted R^2	0.112
Residual Std. Error	$0.093(df = 24993)$
F statistic	$526.290^{***}(df = 6; 24993)$
Note:	*p<0.1; **p<0.05; ***p<0.01

4.3.2.3.3 *IRF* bidding policy Table 4.19 shows the experiment results for substitutable resources and the *IRF* bidding policy: mean, median, and standard deviation as a function of the number of customer agents, as well as the differences from the previous experiment for the mentioned values.

The standard deviation is in the interval $[0.101, 0.19]$. For more than five up to approximately 75, the utility ratio increases with the number of customer agents. For more customer agents, it is quasi-constant.

Table 4.19: Utility ratio as function of number of customer agents for substitutable resources and *IRF* bidding policy.

$\lvert\mathcal{A}_C\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
5	0.7936	0.7651	0.1894	-	-	-
10	0.7500	0.7460	0.1608	0.0436	0.0191	0.0286
15	0.7555	0.7505	0.1405	0.0055	0.0046	0.0203
20	0.7613	0.7566	0.1329	0.0058	0.0061	0.0076
25	0.7578	0.7585	0.1223	0.0035	0.0019	0.0106
30	0.7765	0.7697	0.1212	0.0187	0.0112	0.0011
35	0.7897	0.7794	0.1189	0.0132	0.0097	0.0024
40	0.7902	0.7787	0.1164	0.0005	0.0007	0.0025
45	0.7983	0.7872	0.1132	0.0081	0.0084	0.0032
50	0.8152	0.7975	0.1102	0.0169	0.0103	0.0030
55	0.8161	0.8008	0.1101	0.0008	0.0033	0.0001
60	0.8175	0.8017	0.1153	0.0015	0.0009	0.0052
65	0.8233	0.8016	0.1140	0.0058	0.0000	0.0013
70	0.8258	0.8100	0.1090	0.0025	0.0084	0.0050
75	0.8333	0.8164	0.1059	0.0075	0.0065	0.0031
80	0.8310	0.8157	0.1071	0.0023	0.0008	0.0012
85	0.8295	0.8157	0.1037	0.0015	0.0000	0.0034
90	0.8289	0.8133	0.1085	0.0006	0.0024	0.0048
95	0.8338	0.8190	0.1016	0.0049	0.0057	0.0070
100	0.8219	0.8077	0.1075	0.0119	0.0113	0.0059
$[5, 100]$	0.8072	0.7895	0.1245	-	-	-

Figure 4.15 shows the results as a function of the number of customer agents. More than 75% of the experiments arrive at an efficiency above 0.65, except for five and ten customer agents. For these experiments, the third quartile ends in the interval $[0.6, 0.65]$.

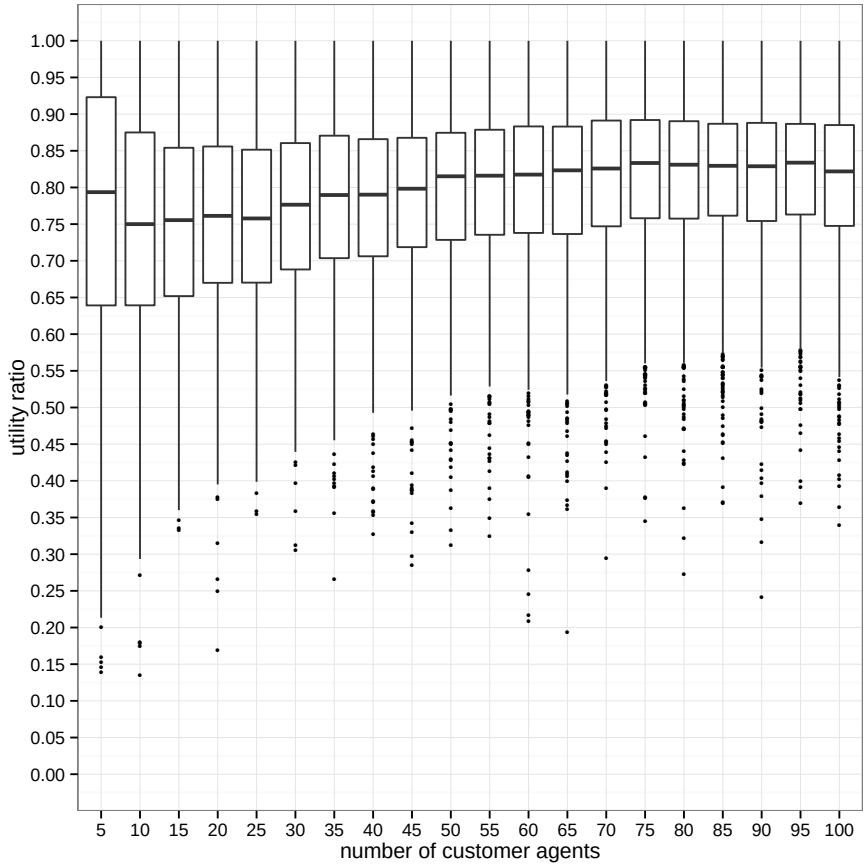

Figure 4.15: Utility ratio as function of number of customer agents for substitutable resources and *IRF* bidding policy.

In table 4.20, mean, median, and standard deviation are shown as a function of the number of SP agent tiers over all experiments for substitutable resources and the *IRF* bidding policy. It also includes the differences from the previous experiment. The median efficiency of the protocol is decreasing for an increased number of tiers. The median utility ratio values lie in the interval $[0.736, 0.848]$ over all experiments. The standard deviation lies between 0.103 and 0.128.

In figure 4.16, the utility ratio is shown as a function of the number of SP agent tiers for substitutable resources and the *IRF* bidding policy. More than 75% of the experiments arrive at an efficiency at or above 0.65.

Table 4.20: Utility ratio as function of number of SP agent tiers for substitutable resources and *IRF* bidding policy.

$\|\Lambda\|$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
1	0.8475	0.8385	0.1087	-	-	-
2	0.8393	0.8244	0.1039	0.0082	0.0141	0.0049
3	0.8169	0.7960	0.1180	0.0223	0.0284	0.0141
4	0.7792	0.7624	0.1278	0.0377	0.0336	0.0098
5	0.7365	0.7265	0.1276	0.0427	0.0359	0.0002
$[1, 5]$	0.8072	0.7895	0.1245	-	-	-

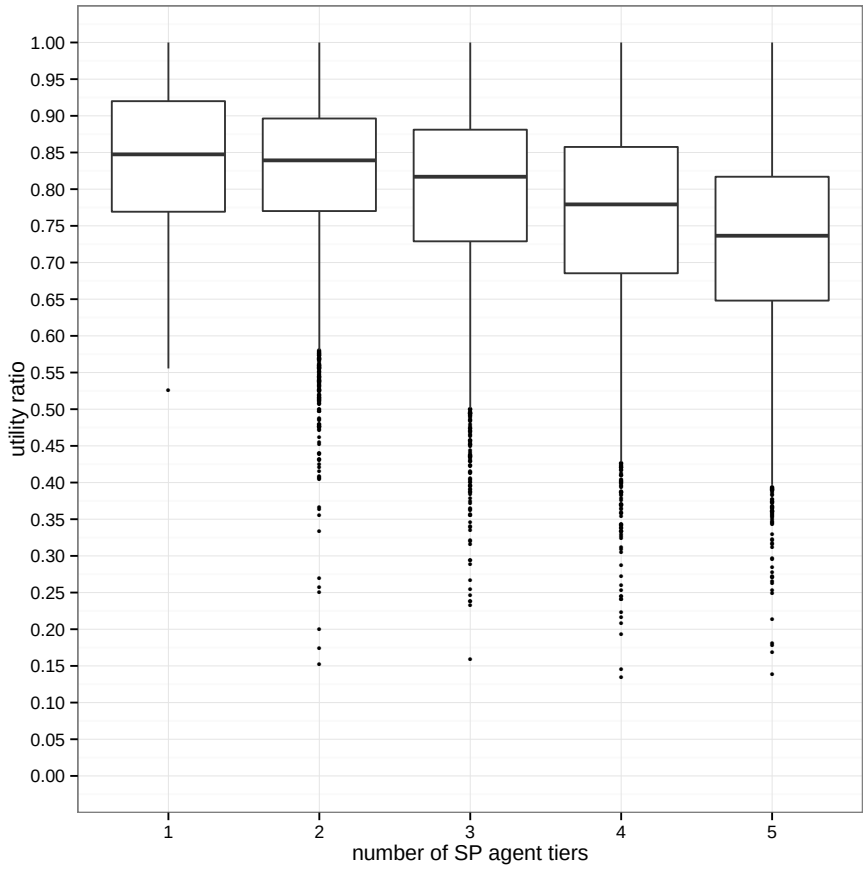

Figure 4.16: Utility ratio as function of number of SP agent tiers for substitutable resources and *IRF* bidding policy.

Table 4.21 shows mean, median, and standard deviation as a function of the number of SPs in tier 1 for substitutable resources and the *IRF* bidding policy. In addition, it shows the differences (Δ) from the previous experiment (line) for median, mean, and standard deviation. The median efficiency of the protocol is increasing for an increased number of SP agents in tier 1. The median utility ratio values lie in the interval $[0.761, 0.829]$ over all experiments.

The standard deviation lies between 0.105 and 0.150 over all experiments, and is considerably higher for two SP agents in the first tier. This results from a low SP-customer-ratio for these experiments and therefore a worse utilization of the resources, due to the myoptic allocation strategy of the SP agents to allocate resources for agents with higher demand first.

Table 4.21: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *IRF* bidding policy.

$\lvert\mathcal{A}_{SP,1}\rvert$	**median**	**mean**	**sd**	Δ_{median}	Δ_{mean}	Δ_{sd}
2	0.7619	0.7492	0.1500	-	-	-
4	0.7966	0.7809	0.1258	0.0347	0.0317	0.0242
6	0.8099	0.7948	0.1153	0.0133	0.0139	0.0105
8	0.8215	0.8073	0.1100	0.0116	0.0125	0.0053
10	0.8284	0.8156	0.1052	0.0069	0.0083	0.0048
$[2, 10]$	0.8072	0.7895	0.1245	-	-	-

Figure 4.17 shows the utility ratio as a function of the number of SPs in tier 1 for substitutable resources and the *IRF* bidding policy. Over all experiments and more than two SP agents in tier 1, 75% of the experiments arrive at an efficiency above 0.7. In all experiments, more than 25% of the experiments arrive at an efficiency above 0.85.

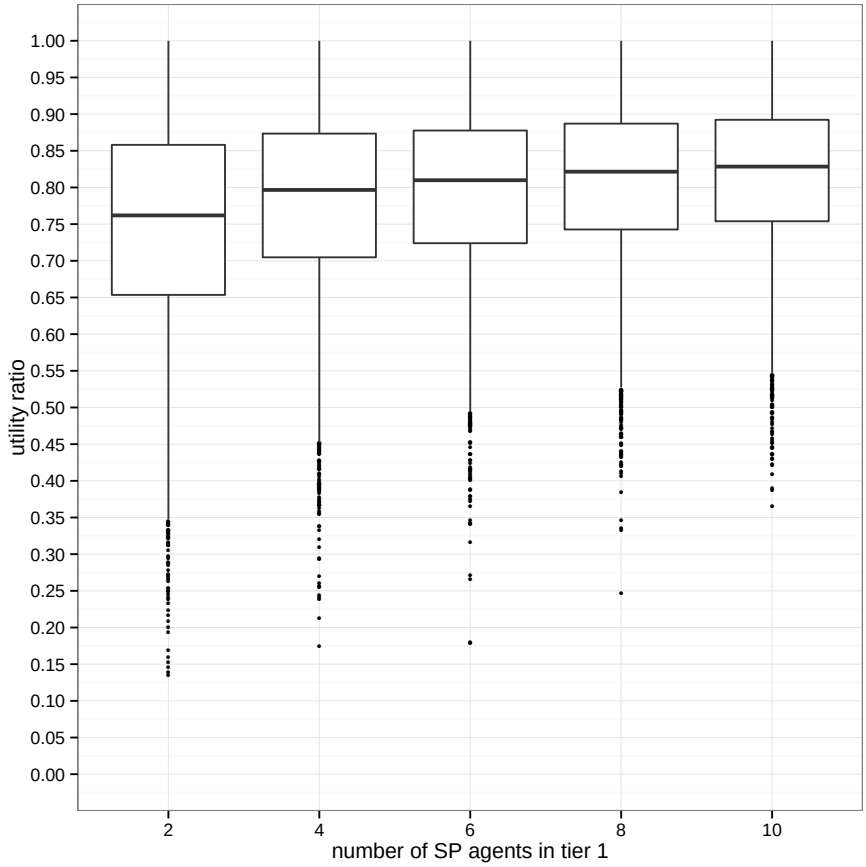

Figure 4.17: Utility ratio as function of the number of SPs in tier 1 for substitutable resources and *IRF* bidding policy.

The results of a linear regression analysis over all experiments for substitutable resources and the *IRF* bidding policy are shown in table 4.22. It contains the utility ratio as the dependent variable, and shows the influence of the number of customer agents ($|\mathcal{A}_C|$), the number of SP agent tiers ($|\Lambda|$), the number of SP agents in tier 1 ($|\mathcal{A}_{SP,1}|$), the number of SP agents ($|\mathcal{A}_{SP}|$), the number of total edges of the SN, and the demand-supply-ratio of the SN on the utility ratio.

All mentioned variables have a significant influence on the utility ratio. The number of customers has a negative effect on the utility ratio, and therewith the protocol implementation's efficiency; since the search space is increased, the probability of allocating resources suboptimally is also increased. The number of SP agent tiers has a negative effect on the utility ratio, and therewith the protocol implementation's efficiency. This results from an increased probability of allocating resources suboptimally in tiers closer to the customer agents first.

In contrast, the number of SP agents in the first tier positively influences the utility ratio. The number of total edges has a positive, the demand-supply-ratio a negative effect on the utility ratio.

Table 4.22: Linear regression analysis results for for substitutable resources and *IRF* bidding policy.

	Dependent variable:
	UTILITY_RATIO
$\lvert\mathcal{A}_C\rvert$	-0.0001^{**}
	(0.0001)
$\lvert\Lambda\rvert$	-0.019^{***}
	(0.001)
$\lvert\mathcal{A}_{SP,1}\rvert$	0.006^{***}
	(0.0003)
$\lvert\mathcal{A}_{SP}\rvert$	-0.005^{***}
	(0.0002)
TOTAL_EDGES	0.001^{***}
	(0.00003)
DEMAND_SUPPLY_RATIO	-0.007^{***}
	(0.0004)
Constant	0.835^{***}
	(0.003)
Observations	$25,000$
R^2	0.211
Adjusted R^2	0.210
Residual Std. Error	$0.111 (df = 24993)$
F statistic	$1,111.687^{***} (df = 6; 24993)$
Note:	*p<0.1; **p<0.05; ***p<0.01

4.3.3 Discussion

The results of the simulation study provide evidence for the efficacy of the proposed protocol in different settings and network topologies, for substitutable and non-substitutable resources, and for the different SP agent bidding policies (for substitutable resources).

The utility ratio decreases with an increasing number of customer agents for non-substitutable resources. This result is intuitive, as the competition for SP agents' resources increases with the number of concurrent customer agents. In addition, the difficulty to find a socially optimal allocation increases with

the number of concurrent customer agents. The same applies for substitutable resources for the *BPRO* bidding policy. For substitutable resources and the *IRF* and *ERF* bidding policies, the utility ratio is increasing with an increasing number of customer agents. Although the competition for SP agents' resources increases with the number of concurrent customer agents in these cases, the negative effects of preferring lower tier (*IRF*) or higher tier (*ERF*) SP agents' resources is counteracted by the higher number of requested resources over all SP agents. That is, with an increasing number of customer agents, less efficient single allocations are antagonized by the higher total number of resources allocated to customer agents.

For substitutable resources, the *IRF* and *BPRO* bidding policies are dominated by the *ERF* bidding policy for an increasing number of customer agents. This results from the fact that SP agents using the *BPRO* bidding policy never combine procured services from upstream SP agents with services they produce using their own resources. Thus, SP agents are more likely to submit partial bids to customer agents; those bids are refused, as they do not fulfill the customer agents' requirements. For the *IRF* bidding policy, SP agents do combine procured services from upstream SP agents with services they produce using their own resources. However, the *IRF* bidding policy limits the choice of resources to the extend of which the required capacity for fulfilling a customer agent's request exceeds the available capacity of the SP agent. Since the effects of choosing suboptimal allocations are counteracted by the effects of the larger solution space, the *ERF* bidding policy converges to an efficiency of about 0.9. Further, the SP agents using the *ERF* bidding policy can still choose from competing SP in upstream SN tiers, potentially allocating the resources from the most cost-efficient SP in the lowest tier.

For non-substitutable resources, the efficiency of the protocol is slightly decreasing for an increased number of SP agent tiers, though it is not monotonically decreasing. The allocation complexity increases with additional tiers, as the solution space increases. This leads to decreased efficiency of the distributed allocation.

For an increasing number of SP agent tiers and substitutable resources the efficiency of the *BPRO* bidding policy decreases monotonically, since, in average over all experiments, the number of SP agents – and therewith the solution space – also increases with the number of tiers. Hence, it is more difficult to approximate optimal allocations. The *ERF* bidding policy's efficiency is quasi-constant, regarding the number of tiers. Here, the converse effects of suboptimal allocations and increased solution space lead to quasi-constant efficiency of the

protocol, independent of the number of tiers. For the *IRF* bidding policy, the efficiency of the allocations decreases with the number of tiers. The reason is that with this bidding policy it is more likely that resources of a suboptimal SP agent are allocated to the provision of a service, as the customer agents (i) do not have access to upstream SP agents and (ii) the SP agents in tier 1 try to sell their own resources, for which the cost can be higher than in upstream SN tiers, first.

All bidding policies obviously show approximately the same efficiency for one SP tier. For $|\Lambda| > 1$, i.e., more than one SP tier, the difference in bidding policy efficiency is obvious. For the *BPRO* bidding policy, the efficiency of the allocations monotonically decreases with the number of tiers in quasi-linear form. The same applies for the *IRF* bidding policy. The *ERF* bidding policy's efficiency leads to quasi-constant efficiency of the protocol, independent of the number of tiers.

For non-substitutable resources, the utility ratio increases for additional SP agents in tier 1. Since the competition cannot be increased as the resources are non-substitutable, the larger number of SP agents in tier 1 does not significantly increase the probability that customer requests can be fulfilled to their full extend after reaching a threshold.

The utility ratio increases for the *BPRO* and *IRF* bidding policies with an increasing number of SP agents in tier 1. It is quasi-constant for the *ERF* bidding policy. An increased utility ratio for additional SP agents in tier 1 for substitutable resources for the *IRF* and *BPRO* bidding policies results from a larger number of SP agents in this tier and therewith an increased competition and probability, respectively, that a customer request can be fulfilled to its full extend. The increased competition in tier 1 increases the probability of more cost-efficient SP agents to be located on this tier. However, for the *ERF* bidding policy, the increased competition is counteracted by prioritization of higher tier SP agents' resources. Thus the utility ratio for additional SP agents in tier 1 for substitutable resources is quasi-constant for the *ERF* bidding policy.

Chapter 5

Conclusions

Concluding the research, this chapter recapitulates the key contributions and summarizes the utilized procedure and methods to develop them. It discusses limitations of the proposed protocol for resource allocation and gives an outlook on future research to complement the results of this thesis.

5.1 Contributions

The contributions of this thesis are a multi-tier resource allocation protocol for composite service provision over multiple SN tiers, along with a formal framework which considers contractual dependencies. To solve the problem of multi-tier dependencies in resource allocation in SNs, this research has applied an interaction protocol engineering perspective. A game-theoretic problem analysis, based on the framework, has been performed. It has been shown that computing socially optimal allocations in SNs is $\mathcal{NP}$-complete and that Nash equilibria exist for these allocations.

A game-theoretic model of the protocol has been presented. It has been proven that the proposed protocol prevents overcommitments. In addition, the protocol is incentive compatible for SP agents, individually rational, and budget balanced; it also constitutes a polynomial time heuristic for the social welfare maximization problem. Under the assumption that resources are non-substitutable between SPs, it guarantees socially optimal allocations for single customer requests.

The protocol specification has been provided in UML sequence diagrams, finite state machines, and a Promela model. To formally verify safety as well as liveness properties, the model checker Spin has been applied.

The proposed protocol has been implemented as a reusable capability for the Jadex BDI agent system. Simulation experiments have been executed, using

this implementation along with Jadex BDI agents for different Cloud computing scenarios. In these experiments, the allocative efficiency of the protocol has been compared to the optimal allocation as a centralized benchmark in different settings (e.g., number of customer and SP agents) for three SP bidding policies. Further, the simulation study has been conducted with and without substitutable resources, i.e., SP agents can decide if they use own resources or procure sub-services from SP agents in the next tier in case of substitutable resources. The simulation-based evaluation has demonstrated the efficacy, utility, and quality of the protocol. This thesis has shown that the protocol provides means for multi-tier resource allocation without centralized control; therefore, the required ad hoc contracting of the needed services is facilitated, honoring the dependencies of agreements over multiple SN tiers.

5.2 Future research

Future research might be conducted in four directions. First, probabilistic theory can be used to decrease response time and communication complexity if the requesting customer requires an immediate response. If the response time is non-critical, SPs can apply the proposed protocol and can store responses from other SP agents. If it is critical, SP agents can use this stored knowledge to respond to customers quickly. As a matter of principle, SP agents cannot avoid overcommitments if they rely on historical data and respond to their customers based on probabilistic data. Second, more work needs to be done on developing (probabilistic) decision models for SPs. Service production functions with different types and combinations of substitutable as well as non-substitutable resources have to be considered.

Third, below-penalty decommitments, respectively leveled commitments, can be combined with the proposed protocol. Together with advanced scheduling for SP agents' resource utilization, these approaches have shown to have merits for the allocative efficiency in settings with uncertainty. Fourth, the protocol can be used with different pricing schemata. In this research, resource utilization is assumed to be certain once an agreement with a customer agent is established. However, resource utilization may be fuzzy to a certain degree, and the application of revenue management methods like overbooking can provide benefits for SP agents. In contrast, an aggressive application of these methods can have a negative effect on the providers' reputation.

Bibliography

Alcatel-Lucent (2012), Alcatel-lucent 2011 global cloud ITDM study, Technical report, Alcatel-Lucent.

Alford, T. & Morton, G. (2009), The economics of cloud computing, Technical report, Booz Allen Hamilton.

Andersson, A., Tenhunen, M. & Ygge, F. (2000), Integer programming for combinatorial auction winner determination, *in* 'Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)', IEEE Computer Society, pp. 39–46.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S. & Xu, M. (2007), 'Web services agreement specification (WS-Agreement)', Open Grid Forum (OGF) Proposed Recommendation GFD.107.

Anthony, P. & Jennings, N. R. (2003), 'Developing a bidding agent for multiple heterogeneous auctions.', *ACM Transactions on Internet Technology* **2**(3), 185–217.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. & Zaharia, M. (2010), 'A view of cloud computing', *Commun. ACM* **53**(4), 50–58.

Arrow, K. J., Sen, A. K. & Suzumura, K., eds (2002), *Handbook of Social Choice and Welfare*, Vol. 1, Elsevier.

Bichler, M. (2000), 'An experimental analysis of multi-attribute auctions', *Decision Support Systems* **29**(3), 249–268.

Binmore, K. (1992), *Fun and Games: A Text on Game Theory*, D. C. Heath and Company.

Bitner, M. J., Faranda, W. T., Hubbert, A. R. & Zeithaml, V. A. (1997), 'Customer contributions and roles in service delivery', *International Journal of Service Industry Management* **8**(3), 193–205.

Blau, B., van Dinther, C., Conte, T., Xu, Y. & Weinhardt, C. (2009), 'How to coordinate value generation in service networks–a mechanism design approach', *Business and Information Systems Engineering (BISE)* **1**(5), 343–356.

Bo, A. & Lesser, V. (2010), 'Characterizing contract-based multi-agent resource allocation in networks', *IEEE Trans. on Systems, Man and Cybernetics, Part B: Cybernetics 40* **3**, 575–586.

Bond, A. H. & Gasser, L. G. (1988), An analysis of problems and research in DAI, *in* A. H. Bond & L. G. Gasser, eds, 'Readings in Distributed Artificial Intelligence', M. Kaufmann, San Mateo, CA, USA, chapter 1, pp. 3–35.

Booch, G. (1993), *Object-Oriented Analysis and Design with Applications*, Addison-Wesley Longman, Amsterdam.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. & Orchard, D. (2004), 'Web services architecture', World Wide Web Consortium (W3C) Working Group Note 11 February 2004.

Bratman, M. E., Israel, J. & Pollack, M. E. (1988), 'Plans and resource-bounded practical reasoning', *Computional Intelligence* **4**, 349–355.

Cardoso, J., Sheth, A., Miller, J., Arnold, J. & Kochut, K. (2004), 'Quality of service for workflows and web service processes', *Web Semantics: Science, Services and Agents on the World Wide Web* **1**(3), 281–308.

Castelfranchi, C. (1995), Guarantees for autonomy in cognitive agent architecture, *in* M. Wooldridge & N. R. Jennings, eds, 'Proceedings of the Intelligent Agents, ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Amsterdam, The Netherlands, August 8-9, 1994', Vol. 890 of *Lecture Notes in Computer Science*, Springer, pp. 56–70.

Castelfranchi, C. (1998), 'Modelling social action for AI agents', *Artificial Intelligence* **103**(1–2), 157–182.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-aguilar, J. A. & Sousa, P. (2006), 'Issues in multiagent resource allocation', *Informatica* **30**, 3–31.

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S. (2001), 'Web services description language (WSDL) 1.1', World Wide Web Consortium (W3C) Note 15 March 2001.

Clarke, E. M., Grumberg, O. & Peled, D. (1999), *Model Checking*, MIT Press.

Cramton, P., Shoham, Y. & Steinberg, R., eds (2006), *Combinatorial Auctions*, MIT Press.

Czajkowski, K., Foster, I. & Kesselman, C. (2004), Grids in context, *in* I. Foster & C. Kesselman, eds, 'The Grid 2: Blueprint for a New Computing Infrastructure', Morgan Kaufmann Publishers Inc., chapter 1, pp. 259–283.

Dasgupta, P., Hammond, P. & Maskin, E. (1979), 'The implementation of social choice rules: Some general results on incentive compatibility', *The Review of Economic Studies* **46**(2), 185–216.

Dobson, G., Lock, R. & Sommerville, I. (2005), QoSOnt: a QoS ontology for service-centric systems, *in* 'Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO '05),', IEEE Press, pp. 80–87.

Dustdar, S. (2004), 'Web services workflows—composition, co-ordination, and transactions in service-oriented computing', *Concurrent Engineering* **12**(3), 237–245.

Endriss, U. & Maudet, N. (2005), 'On the communication complexity of multilateral trading: Extended report', *Journal of Autonomous Agents and Multi-Agent Systems* **11**(1), 91–107.

Endriss, U., Maudet, N., Sadri, F. & Toni, F. (2003), On optimal outcomes of negotiations over resources, *in* 'Proceedings of the second international joint conference on Autonomous agents and multiagent systems', AAMAS '03, ACM, New York, NY, USA, pp. 177–184.

Endriss, U., Maudet, N., Sadri, F. & Toni, F. (2006), 'Negotiating socially optimal allocations of resources', *Journal of Artificial Intelligence Research* **25**(1), 315–348.

Ferber, J. (1999), *Multi-Agent Systems: An Introduction to Distributed Artifcial Intelligence*, Addison-Wesley.

FIPA (2002*a*), 'FIPA communicative act library specification', http://www.fipa.org/specs/fipa00037/.

FIPA (2002*b*), 'FIPA contract net interaction protocol specification', http://www.fipa.org/specs/fipa00029/.

Foster, I., Kesselman, C. & Tuecke, S. (2001), 'The anatomy of the grid: Enabling scalable virtual organizations', *Int. J. High Perform. Comput. Appl.* **15**(3), 200–222.

Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J. & Reich, J. V. (2006), 'The open grid services architecture, version 1.5'. Open Grid Services Architecture WG, Global Grid Forum.

Franklin, S. & Graesser, A. (1997), Is it an agent, or just a program?: A taxonomy for autonomous agents, *in* J. Müller, M. Wooldridge & N. Jennings, eds, 'Intelligent Agents III Agent Theories, Architectures, and Languages', Vol. 1193 of *Lecture Notes in Computer Science*, Springer, pp. 21–35.

Gadrey, J. (2000), 'The characterization of goods and services: An alternative approach', *Review of Income and Wealth* **46**(3), 369–387.

Gannon, J. D., Hamlet, R. G. & Mills, H. D. (1987), 'Theory of modules', *IEEE Transactions on Software Engineering* **13**(7), 820–829.

Garey, M. R. & Johnson, D. S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York.

Gasser, L. (1991), 'Social conceptions of knowledge and action: Dai foundations and open systems semantics', *Artificial Intelligence* **47**(1–3), 107–138.

Genesereth, M. R. & Ketchpel, S. P. (1994), 'Software agents', *Communications of the ACM* **37**(7), 48–53.

Giordano, L., Martelli, A. & Schwind, C. (2007), 'Specifying and verifying interaction protocols in a temporal action logic', *Journal of Applied Logic* **5**(2), 214—-234.

Green, J. & Laffont, J.-J. (1977), 'Characterization of satisfactory mechanisms for the revelation of preferences for public goods', *Econometrica* **45**(2), 427–438.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A. & Lafon, Y. (2007), 'SOAP version 1.2', World Wide Web Consortium (W3C) Recommendation 27 April 2007.

Haugen, O. & Runde, R. K. (2009), Enhancing uml to formalize the fipa agent interaction protocol, *in* K. Fischer, J. P. Mueller, J. Odell & A. J. Berre, eds, 'Agent-Based Technologies and Applications for Enterprise Interoperability', Vol. 25 of *Lecture Notes in Business Information Processing*, Springer, pp. 154–173.

Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004), 'Design science in information systems research', *MIS Quarterly* **28**(1), 75–105.

Hewitt, C. (1991), 'Open information systems semantics for distributed artificial intelligence', *Artificial Intelligence* **47**(1–3), 79–106.

Hill, T. (1999), 'Tangibles, intangibles and services: A new taxonomy for the classification of output', *The Canadian Journal of Economics* **32**(2), 426–446.

Hill, T. P. (1977), 'On goods and services', *Review of Income and Wealth* **23**(4), 537–556.

Hogan, O. & Mohamed, S. (2010), The cloud dividend: Part one, Technical report, Centre for Economics and Business Research (Cebr).

Holzmann, G. J. (1991), *Design and Validation of Computer Protocols*, Prentice Hall, New Jersey.

Holzmann, G. J. (1997), 'The model checker spin', *IEEE Transactions on Software Engineering* **23**(5), 279–295.

Huget, M.-P. & Koning, J.-L. (2003), Interaction protocol engineering, *in* M.-P. Huget, ed., 'Communication in Multiagent Systems, Agent Communication Languages and Conversation Polocies', Vol. 2650 of *Lecture Notes in Computer Science*, Springer, pp. 179–193.

Huhns, M. N. & Stephens, L. M. (1999), Multiagent systems and societies of agents, *in* G. Weiss, ed., 'Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence', MIT Press, pp. 79–120.

Huhns, M., Singh, M., Burstein, M., Decker, K., Durfee, K., Finin, T., Gasser, T., Goradia, H., Jennings, P., Kiran Lakkaraju Nakashima, H., Van Dyke Parunak, H., Rosenschein, J., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T. & Zavafa, L. (2005), 'Research directions for service-oriented multiagent systems', *IEEE Internet Computing* **9**(6), 65–70.

Hurwicz, L. & Walker, M. (1990), 'On the generic nonoptimality of dominant-strategy allocation mechanisms: A general theorem that includes pure exchange economies', *Econometrica* **58**(3), 683—-704.

Huynh, T. D., Jennings, N. R. & Shadbolt, N. R. (2006), 'An integrated trust and reputation model for open multi-agent systems', *Autonomous Agents and Multi-Agent Systems* **13**, 119—-154.

IBM (2011), 'User's manual for CPLEX'. Version 12.3, IBM Corp.

Ingrand, F., Georgeff, M. & Rao, A. (1992), 'An architecture for real-time reasoning and system control', *IEEE Expert* **7**(6), 34–44.

Jaeger, M. C. & Ladner, H. (2006), 'A model for the aggregation of qos in ws compositions involving redundant services', *Journal of Digital Information Management* **4**(4), 44–49.

Jaeger, M., Rojec-Goldmann, G. & Mühl, G. (2004), Qos aggregation for web service composition using workflow patterns, *in* 'Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004), Monterey', pp. 149–159.

Jennings, N. R. (1993), 'Commitments and conventions: The foundation of coordination in multi-agent systems', *The Knowledge Engineering Review* **8**(3), 223–250.

Jennings, N. R. (2000), 'On agent-based software engineering', *Artificial Intelligence* **117**(2), 277–296.

Jennings, N. R. & Wooldridge, M. (2000), Agent-oriented software engineering, *in* J. Bradshaw, ed., 'Handbook in Agent Technology', MIT Press.

Jones, I., ed. (2010), *BREIN Final Demonstrator*. BREIN Deliverable D8.1.1. **URL:** *http://www.eu-brein.com*

Karaenke, P. & Kirn, S. (2010*a*), A multi-tier negotiation protocol for logistics service chains, *in* 'Proceedings of the 18th European Conference on Information Systems (ECIS 2010), June, 6–9, Pretoria, South Africa'.

Karaenke, P. & Kirn, S. (2010*b*), Towards model checking & simulation of a multi-tier negotiation protocol for service chains (extended abstract), *in* van der Hoek, Kaminka, Lesperance, Luck & Sen, eds, 'Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), May, 10–14, Toronto, Canada', pp. 1559–1560.

Karaenke, P. & Leukel, J. (2010), Towards ontology-based QoS aggregation for composite web services, *in* K.-P. Fähnrich & B. Franczyk, eds, 'Proceedings of Informatik 2010, 40. Jahrestagung der Gesellschaft für Informatik e.V. (INFORMATIK 2010), November, 27–October, 1, Leipzig, Germany', pp. 120–125.

Karaenke, P., Leukel, J. & Sugumaran, V. (2013), Ontology-based QoS aggregation for composite web services, *in* 'Proceedings of Wirtschaftsinformatik 2013 (WI 2013), February, 27–March, 01, Leipzig, Germany'.

Karaenke, P., Micsik, A. & Kirn, S. (2009), Adaptive SLA management along value chains for service individualization, *in* R. Alt, K.-P. Faehnrich & B. Franczyk, eds, 'Proceedings of the First International Symposium on Services Science (ISSS2009)', Logos, Berlin.

Karp, R. (1972), Reducibility among combinatorial problems, *in* R. Miller & J. Thatcher, eds, 'Complexity of Computer Computations', Plenum Press, New York.

Kirn, S., ed. (2008), *Individualization Engineering*, Cuvillier.

Kluegl, F. (2001), *Multiagentensimulation - Konzepte, Werkzeuge, Anwendung (German)*, Addison-Wesley.

Kraus, S. (1997), 'Negotiation and cooperation in multi-agent environments', *Artificial Intelligence* **94**(1–2), 79–98.

Laria, G., ed. (2009), *Final BREIN Architecture*. BREIN Deliverable D4.1.3v2. **URL:** *http://www.eu-brein.com*

Luck, M. & d'Inverno, M. (1995), A formal framework for agency and autonomy, *in* V. R. Lesser & L. Gasser, eds, 'Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA', The MIT Press.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F. & Metz, R. (2006), Reference model for service oriented architecture 1.0, OASIS Standard, OASIS.

Malone, T. W. & Crowston, K. (1994), 'The interdisciplinary study of coordination', *ACM Computing Surveys* **26**(1), 87–119.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N.

& Sycara, K. (2004), 'OWL-S: Semantic markup for web services', World Wide Web Consortium (W3C) Member Submission 22 November 2004.

McIlraith, S., Son, T. C. & Zeng, H. (2001), 'Semantic web services', *IEEE Intelligent Systems* **16**(2), 46–53.

Menasce, D. (2004), 'Composing web services: A qos view', *IEEE Internet Computing* **8**(6), 88–90.

Mitchell, T. M. (1997), *Machine Learning*, McGraw-Hill, Inc., New York, NY, USA.

Moulin, H. (1988), *Axioms of Cooperative Decision Making*, Cambridge University Press.

Muñoz Frutos, H., Kotsiopoulos, I., Vaquero, L. M. & Merino, L. R. (2009), Enhancing service selection by semantic qos, *in* 'Proceedings of the 6th European Semantic Web Conference (ESWC 2009)', Springer, pp. 565–577.

Myerson, R. B. & Satterthwaite, M. A. (1983), 'Efficient mechanisms for bilateral trading', *Journal of Economic Theory* **29**(2), 265–281.

Nash, J. F. J. (1950), Non-cooperative games, PhD thesis, Princeton University.

Newell, A. (1982), 'The knowledge level', *Artificial Intelligence* **18**, 87–127.

Nguyen, T. D. & Jennings, N. R. (2004), Coordinating multiple concurrent negotiations, *in* 'Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)', pp. 1064–1071.

Nguyen, T. D. & Jennings, N. R. (2005), 'Managing commitments in multiple concurrent negotiations', *Electronic Commerce Research and Applications* **4**(4), 362–376.

Nisan, N., Roughgarden, T., Tardos, E. & Vazirani, V. V., eds (2007), *Algorithmic Game Theory*, Cambridge University Press.

Oaks, P., ter Hofstede, A. H. M. & Edmond, D. (2003), Capabilities: Describing what services can do, *in* M. E. Orlowska, S. Weerawarana, M. P. Papazoglou & J. Yang, eds, 'Proceedings of the First International Conference on Service-Oriented Computing (ICSOC)', Vol. 2910 of *Lecture Notes in Computer Science*, Springer, pp. 1–16.

OASIS (2007), 'Web services business process execution language version 2.0'. OASIS Standard, Organization for the Advancement of Structured Information Standards.

OMG (2011*a*), 'Business process model and notation (BPMN), version 2.0'. OMG Document Number: formal/2011-01-03, Object Management Group.

OMG (2011*b*), 'Unified modeling language (UML) superstructure, version 2.4.1'. OMG Document Number: formal/2011-08-05, Object Management Group.

O'Sullivan, J., Edmond, D. & ter Hofstede, A. (2002), 'What's in a service?', *Distributed and Parallel Databases* **12**(2-3), 117–133.

Papadimitriou, C. H. (1994), *Computational Complexity*, Addison Wesley.

Papazoglou, M. P. (2003), Service-oriented computing: Concepts, characteristics and directions, *in* 'Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)', pp. 3–12.

Papazoglou, M. P., Traverso, P., Dustdar, S. & Leymann, F. (2008), 'Service-oriented computing: A research roadmap', *International Journal of Cooperative Information Systems* **17**(2), 223–255.

Parkes, D. (2001), Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency, PhD thesis, University of Pennsylvania.

Parkes, D. & Shneidman, J. (2004), Distributed implementations of Vickrey-Clarke-Groves mechanisms, *in* 'Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)', ACM, New York, NY, pp. 261–268.

Parsons, S. & Wooldridge, M. (2002), 'Game theory and decision theory in multi-agent systems', *Autonomous Agents and Multi-Agent Systems* **5**, 243–254.

Piller, F. T., Moeslein, K. & Stotko, C. M. (2004), 'Does mass customization pay? an economic approach to evaluate customer integration', *Production Planning & Control* **15**(4), 435–444.

Pokahr, A., Braubach, L. & Lamersdorf, W. (2005), Jadex: A BDI reasoning engine, *in* R. H. Bordini, M. Dastani, J. Dix & A. E. Fallah-Seghrouchni, eds, 'Multi-Agent Programming', Vol. 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Springer, pp. 149–174.

Preist, C., Bartolini, C. & Byde, A. (2003), Agent-based service composition through simultaneous negotiation in forward and reverse auctions, *in* 'Proceedings of the 4th ACM Conference on Electronic Commerce (EC '03)', ACM, New York, NY, pp. 55–63.

Raiffa, H. (1968), *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, Addison Wesley.

Rasmusen, E. (1989), *Games and Information*, Basil Blackwell, Oxford.

Robinson, M. S. (1985), 'Collusion and the choice of auction', *The RAND Journal of Economics* **16**(1), 141–145.

Rosenschein, J. S. (1985), Rational Interaction: Cooperation Amoung Intelligent Agents, PhD thesis, Stanford University.

Rosenschein, J. S. & Zlotkin, G. (1994), *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, MIT Press.

Rothkopf, M. H., Pekeč, A. & Harstad, R. M. (1998), 'Computationally manageable combinational auctions', *Management Science* **44**(8), 1131–1147.

Russell, S. J. & Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, 2nd edn, Prentice Hall.

Russell, S. J. & Subramanian, D. (1995), 'Provably bounded-optimal agents', *Journal of Artificial Intelligence Research* **2**, 575–606.

Rust, R. T. & Kannan, P. (2002), *E-Service: New Directions in Theory and Practice*, ME Sharpe.

Rust, R. T. & Kannan, P. (2003), 'E-service: a new paradigm for business in the electronic environment', *Communications of the ACM* **46**(6), 36–42.

Sandholm, T. W. (1999), Distributed rational decision making, *in* G. Weiss, ed., 'Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence', MIT Press, pp. 201–258.

Sandholm, T. W. & Lesser, V. R. (1995), Issues in automated negotiation and electronic commerce: Extending the contract net framework, *in* 'Proceedings of the International Conference on Multi-Agent Systems', MIT Press, pp. 328–335.

Sandholm, T. W. & Lesser, V. R. (2001), 'Leveled commitment contracts and strategic breach', *Games and Economic Behavior* **35**(1–2), 212–270.

Schillo, M., Kray, C. & Fischer, K. (2002), The eager bidder problem: a fundamental problem of dai and selected solutions, *in* 'Proceedings of the First international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)', ACM, New York, NY, pp. 599–606.

Sen, A. K. (1970), *Collective Choice and Social Welfare*, Holden-Day.

Si, Y.-W., Edmond, D., Dumas, M. & Hofstede, A. H. M. (2007), 'Specification and execution of composite trading activities', *Electronic Commerce Research* **7**(3-4), 221–263.

Si, Y.-W., Edmond, D., ter Hofstede, A. H., Dumas, M. & Chong, C. U. (2005), Specification of composite trading activities in supply chain management, *in* 'Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE05), March 29 - April 01, 2005, Hong Kong, China'.

Simon, H. A. (1957), *Models of Man: Social and Rational*, John Wiley, New York.

Simon, H. A. (1982), *Models of Bounded Rationality*, Vol. 1, The MIT Press, Cambridge, Massachusetts.

Simon, H. A. (1996), *The Sciences of the Artificial*, 3rd edn, MIT Press, Cambridge, MA.

Singh, M. (1993), 'A semantics for speech acts', *Annals of Mathematics and Artificial Intelligence* **8**(1-2), 47–71.

Smith, C. W. (1989), *Auctions: The Social Construction of Value*, Free Press, New York, NY, USA.

Smith, R. G. (1980), 'The contract net protocol: High-level communication and control in a distributed problem solver', *IEEE Transactions on Computers* **C-29**(12), 1104–1113.

ul Haq, I., Huqqani, A. & Schikuta, E. (2009), Aggregating hierarchical service level agreements in business value networks, *in* U. Dayal, J. Eder, J. Koehler & H. Reijers, eds, 'Proceedings of the 7th International Conference on Business Process Management (BPM 2009)', Springer.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B. & Barros, A. (2003), 'Workflow patterns', *Distributed and Parallel Databases* **14**(3), 5–51.

Vaquero, L. M., Rodero-Merino, L., Caceres, J. & Lindner, M. (2009), 'A break in the clouds: Towards a cloud definition', *ACM SIGCOMM Computer Communication Review* **39**(1), 50–55.

Vehlow, M. & Golkowsky, C. (2011), Cloud computing - navigating the cloud, Technical report, PricewaterhouseCoopers (PwC).

Verma, D. (1999), *Supporting Service Level Agreements on IP Networks*, Macmillan Technical Publishing.

Vickrey, W. (1961), 'Counterspeculation, auctions, and competitive sealed tenders', *Journal of Finance* **16**, 8–37.

von Neumann, J. & Morgenstern, O. (1944), *Theory of Games and Economic Behavior*, Princeton University Press.

Vulkan, N. & Jennings, N. R. (2000), 'Efficient mechanisms for the supply of services in multi-agent environments', *Decision Support Systems* **28**(1–2), 5–19.

Walker, M. (1980), 'On the nonexistence of a dominant strategy mechanism for making optimal public decisions', *Econometrica* **48**(6), 1521—-1540.

Walsh, W. E., Wellman, M. P. & Ygge, F. (2000), Combinatorial auctions for supply chain formation, *in* 'Proceedings of the 2nd ACM Conference on Electronic Commerce (EC00)', ACM, New York, NY, pp. 260–269.

Walsh, W. & Wellman, M. (2003), 'Decentralized supply chain formation: A market protocol and competitive equilibrium analysis', *Journal of Artificial Intelligence Research* **19**, 513–567.

Walton, C. D. (2007), 'Verifiable agent dialogues', *Journal of Applied Logic* **5**(2), 197–213.

Wooldridge, M. (1997), 'Agent-based software engineering', *IEE Proceedings on Software Engineering* **144**(1), 26–37.

Wooldridge, M. (2000), *Reasoning about rational agents*, MIT Press.

Wooldridge, M. (2009), *An Introduction to Multi Agent Systems*, 2nd edn, John Wiley & Sons.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligent agents: Theory and practice', *Knowledge Engineering Review* **10**(2), 115–152.

Wooldridge, M., Jennings, N. R. & Kinny, D. (2000), 'The gaia methodology for agent-oriented analysis and design', *Autonomous Agents and Multi-Agent Systems* **3**(3), 285–312.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J. & Chang, H. (2004), 'Qos-aware middleware for web services composition', *IEEE Transactions on Software Engineering* **30**(5), 311–327.

Zhang, X., Lesser, V. & Abdallah, S. (2005), 'Efficient management of multi-linked negotiation based on a formalized model', *Autonomous Agents and Multi-Agent Systems* **10**(2), 165–205.