TECHNISCHE
UNIVERSITÄT
DARMSTADT

Cross-organizational Service-based Workflows – Solution Strategies for Quality of Service Optimization

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Wirtsch.-Ing. Julian Eckert

Geboren am 29. April 1979
in Darmstadt, Hessen

Vorsitz: Prof. Dr.-Ing. Thomas Hartkopf
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Michael Zink

Tag der Einreichung: 18. August 2009
Tag der Disputation: 06. Oktober 2009

Hochschulkennziffer D17
Darmstadt 2009

ABSTRACT

By the application of the Service-oriented Architecture (SOA) paradigm on business processes, workflows can be decomposed into basic activities that can be realized by reusable services offering a specific business functionality. In order to compose cross-organizational service-based workflows, services can be sourced from internal as well as from external providers. On a large service market, services are offered with varying Quality of Service (QoS) levels and several pricing models. Providing a high level of QoS concerning composite services or service-based workflows is of high importance for an intermediary, acting as a service orchestrator, selling composed service-based workflows to his consumers. Besides efficient workload predictions, fast and efficient solution strategies for QoS and cost optimization are mandatory for the intermediary in order to stay competitive and to enable fast reaction strategies on varying demands of workflow execution requests.

This thesis provides several contributions to the QoS optimization of service-based workflows from the intermediary's point of view. The main contribution is the development and the evaluation of efficient resource planning heuristics, facilitating the fast computation of invocation plans out of services with limited execution capacities, offered by a specific pricing model. Thus, a resource planning optimization model, solved by mathematical optimization with an exact solution as well as by the application of the developed heuristics, is introduced.

Additional contributions address several challenges in the field of QoS optimization of service-based workflows. As a foundation, a classification of pricing models for services is developed and the impact of several pricing models on the service selection process for service-based workflows is presented. Several solution strategies for the QoS optimization are developed supporting the intermediary in the worst- and average-case performance analysis of service-based workflows. In an average-case analysis, key findings of queuing theory are adapted to the concept of service-based workflows and several optimization approaches are developed. These support the intermediary by the optimization of the service utilization incorporating constraints such as the overall response time. Furthermore, major concepts of network calculus are identified and adapted to the concept of service-based workflows. Consequently, optimization approaches are developed facilitating the optimization of QoS parameters such as the delay or the throughput in the worst-case. Finally, an architectural extension for generic QoS management systems for service-based workflows is proposed, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows.

KURZFASSUNG

Eine durchgängige Anwendung des Paradigmas dienstbasierter Architekturen auf Geschäftsprozesse ermöglicht die Dekomposition von Workflows in grundlegende Aktivitäten, die durch Dienste einer bestimmten Funktionalität abgebildet werden können. Gerade im Hinblick auf die Realisierung von unternehmensübergreifenden dienstbasierten Workflows ist es möglich, Dienste sowohl von internen als auch von externen Anbietern einzubinden. Diese Dienste werden auf einem sogenannten Dienste-Marktplatz mit variierenden Dienstgütemerkmalen und verschiedenen Preismodellen angeboten. Für einen Intermediär, der Workflows komponiert und diese an seine Kunden verkauft, ist es von hoher Bedeutung einen Workflow mit hohen Dienstgüteeigenschaften anzubieten. Neben einer effizienten Vorhersage des Anfrageverhaltens sind schnelle und effiziente Lösungsstrategien zur Optimierung von Dienstgüte und Kosten von hoher Relevanz um wettbewerbsfähig zu bleiben und um schnell auf sich änderndes Nachfrageverhalten von Workflow-Ausführungsanfragen reagieren zu können.

Die vorliegende Arbeit liefert verschiedene Beiträge zur Dienstgüteoptimierung von dienstbasierten Workflows aus Sicht eines Intermediärs. Kernbeitrag der Arbeit ist die Entwicklung und Evaluierung effizienter Heuristiken für die Ressourcenplanung, die eine schnelle Berechnung von Ausführungsplänen von Diensten mit begrenzter Ausführungskapazität und einem speziellen Preismodell ermöglichen. Zur Lösung des Problems der Ressourcenplanung werden mathematische Verfahren und Heuristiken eingesetzt.

Weitere Beiträge dieser Arbeit adressieren verschiedene zusätzliche Herausforderungen der Dienstgüteoptimierung von dienstbasierten Workflows. Ausgangspunkt ist eine Klassifizierung von Preismodellen sowie die Untersuchung der Auswirkungen verschiedener Preismodelle auf den Diensteauswahlprozess. Darüber hinaus werden verschiedene Lösungsstrategien der Dienstgüteoptimierung entwickelt, die den Intermediär in der Analyse des mittleren und des schlechtesten Verhaltens hinsichtlich der Dienstgüte eines Workflows unterstützen. Für die Analyse des mittleren Verhaltens werden Erkenntnisse aus der Warteschlangentheorie an dienstbasierte Workflows adaptiert und Optimierungsansätze entwickelt, die es dem Intermediär ermöglichen, die Serviceauslastung unter der Berücksichtigung von verschiedenen Restriktionen, wie beispielsweise die Antwortzeit, zu optimieren. Für die Analyse des schlechtesten Verhaltens werden wesentliche Erkenntnisse aus dem Network Calculus identifiziert und an dienstbasierte Workflows adaptiert. Des Weiteren werden Optimierungsansätze entwickelt, die wesentliche Dienstgüteparameter wie beispielsweise Verzögerung oder Durchsatz optimieren. Abschließend wird eine Architekturerweiterung für Dienstgütemanagementsysteme vorgeschlagen, die die Realisierung und Implementierung der propagierten Lösungsstrategien zur Ressourcenplanung von dienstbasierten Workflows ermöglicht.

ACKNOWLEDGMENTS

First, I would like to thank Prof. Dr.-Ing. Ralf Steinmetz for being my supervisor and for offering me a great job opportunity at the Multimedia Communications Lab at Technische Universität Darmstadt. Also, I would like to thank my co-supervisor Prof. Dr.-Ing. Michael Zink for his help, especially while finishing my work.

A special thank goes to my colleagues of the research group IT-Architectures, especially Dr.-Ing. Nicolas Repp, Stefan Schulte, Michael Niemann, André Miede, Apostolos Papageorgiou, and Dieter Schuller. Also I want to thank my former colleagues Dr.-Ing. Rainer Berbner and Dr.-Ing. Krishna Pandit for the excellent and amicable cooperation.

Furthermore, special thanks go to my colleagues of the Multimedia Communications Lab – KOM for their support in the former years, especially Prof. Dr.-Ing. Matthias Hollick, Dr.-Ing. Nicolas Liebau, and Sebastian Kaune for their advice.

In addition, I would like to thank all persons who proof-read my dissertation and I would like to thank Karola Schork-Jakobi, who supported me in many cases, especially at the conferences of the E-Finance Lab e.V. Furthermore, I would like to thank my students, especially Deniz and Tim. It was great working with you!

A special thank goes to my parents, my brother Elias, and my sister Jana who supported me so much in my life. Thank you for everything!

Finally, I would like to thank you, Charlotte, for your support during writing my dissertation and for your support in my life.

Darmstadt 2009

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

The globalization in recent years is one of the major drivers for enterprises to become flexible and agile. Nowadays, enterprises offer products globally and have a broader market that forces them to increase product diversity. Besides an increasing amount of competitors which are located all over the world, developing countries become new cost beneficial locations for factories and an increasing number of people participate in market growth. The huge amount of competitors and the changing demand behavior of the customers force enterprises to adapt their products very fast to the needs of the customers. Consequently, only those enterprises, which offer products with a short time to market and low production costs, are able to survive in the long-term.

Enterprise cooperations evolve in order to build up strategic alliances with the aim to lower product costs or to achieve benefits from multi-sourcing. Thus, the business processes of the enterprises have to be adapted to the new challenges. They have to be flexible and agile in order to fulfill the business requirements of the enterprise. Continuous reorganization of value chains and networking result in cross-organizational business processes.

In many industry sectors, IT systems have to be adapted or have to be build up from scratch in order to support a large amount of business processes. Concerning enterprise cooperations, the IT systems and the underlying IT architecture have to fit together in order to realize competitive advantages. IT-supported business processes, so-called workflows and especially cross-organizational workflows gain more and more importance in recent years [LR99]. This increases the need for enterprises that own IT systems and IT systems from business partners fit together and are able to communicate with each other. Flexible IT architectures are a prerequisite of dynamic adaptable business processes as business processes have to integrate various heterogeneous legacy systems, platforms, operating systems, and communication mechanisms [Kel02]. In the past, this heterogeneity led to an increased inflexibility and complexity which is difficult to manage.

In particular, the concept of Service-oriented Architectures (SOAs) addresses these challenges and requirements on IT. A recent survey affirms that the main drivers for a SOA implementation are an increased flexibility, a shorter time to market, and process optimization [ERM09]. Especially, in the financial industry SOA is of high importance [ERN+08], [ERS+09], [EBRS09]. SOA, as an architectural paradigm, can be regarded as an approach facilitating demanded flexibility and enabling the alignment of application landscapes to business-driven de-

mands [Mah07]. In contrast to classical software architectures, which describe a complete system structure, SOA focuses on the provision and description of services and functionalities for application integration purposes. These services are characterized by well-defined functional and platform-independent interfaces, hiding the concrete implementation [KBS04]. The concept of SOA supports the ability that workflows can be decomposed to granular services fulfilling the functionality of a particular task whereas a service can be a technical service as well as a service provided by human interaction.

As one possible implementation for a technical service, Web services gained more and more importance concerning the realization of cross-organizational service-based workflows. Services are loosely coupled, reusable software components which are accessible via the Internet and communicate with each other via standardized messages [ACKM04]. Web services are based on open XML-standards such as SOAP [ML07], [GHM+07a], [GHM+07b], Web Service Description Language (WSDL) [BL07], [CMRW07], [CHL+07], and Universal Description, Discovery and Integration (UDDI) [CHvRR04] and can operate independent of the used platform, operating systems, or programming languages.

For the composition of flexible business processes, standardization concerning service publication, discovery, and invocation is necessary. The loose coupling enables service invocations across enterprise boundaries and service compositions to value-added service-based workflows. Besides the functional requirements of a service, the non-functional requirements are of high importance. Consumers, requesting a service at a service provider, want to realize a specific functionality with the help of a specific service invocation. Furthermore, it is desired that the service execution is bound by certain minimum or maximum constraints for, e.g., response time, execution capacity, or delay. This kind of quality assurance is of high importance for services as well as for entire business processes. Typically, Quality of Service (QoS) comprises, e.g., the response time of a business process, the availability of an invoked service, or the error rate of an IT system. Consequently, a large amount of service compositions in which services from internal as well as from external partners are involved, i.e., business critical cross-organizational service-based workflows, require an effective QoS management [BL06].

In a scenario in which a so-called intermediary acts as a workflow orchestrator being responsible for workflow composition and workflow execution for a large variety of workflow requesters, providing QoS is crucial. Besides the functional properties of the services, QoS management is of high importance for the intermediary in order to meet customer requirements and to offer a certain QoS level to the customers. In particular, workflow consumers requesting workflow execution at several QoS levels are classified into several consumer groups. Each of those groups has common QoS requirements concerning the execution of the demanded workflow. Providing workflow execution to his customers, the intermediary has to develop several solution strategies for QoS

optimization in order to meet customer demands and to offer the composed workflow cost-efficiently. Concerning workflows with a high repetition rate such as a generic credit process, one or more services with limited service execution capacities have to be invoked in parallel in order to be able to serve all incoming workflow execution requests. Therefore, it is necessary to develop resource planning mechanisms for workflows determining which services from which service providers have to be invoked at which step of the workflow in order to handle a huge amount of workflow execution requests. Furthermore, the workflow invocation plans have to be determined in order to be able to meet customer requirements and to optimize non-functional QoS parameters and costs.

Addressing those challenges, this thesis focuses on solution strategies, which optimize QoS from the intermediary's point of view. These strategies determine cost-efficient invocation plans for services in several scenarios, taking into account several pricing models for services of a large variety of service providers. The developed approaches use information about the functional and non-functional characteristics of services, described in the Service Level Agreements (SLAs), and determine cost-efficient invocation plans for worst- and average-case scenarios. Furthermore, parallel service invocations are addressed in the resource planning-based optimization approach that is proven to be NP-hard. Thus, heuristics are developed and evaluated in an extensive simulation that handle the addressed resource planning problem at low computation times and at high solution qualities.

1.2 CONTRIBUTIONS OF THIS THESIS

This thesis develops, analyzes, and discusses several solution strategies for the QoS optimization for service-based workflows in detail. Especially, the compliance with several QoS and cost requirements on the workflow are of high importance in order to realize cost-efficient workflow executions. Furthermore, impacts of pricing models on the service selection process are discussed as well as cost-efficient (heuristic) solutions for the QoS optimization approaches are developed and evaluated. Figure 1 describes a classification of the contributions of this thesis by the extended and adapted research agenda of [Pap03].

The main contributions of this work with respect to the research in the field of SOA are classified as follows:

1. Development of QoS optimization approaches comprising
    a) the development of a service pricing model classification for service-based workflows and an analysis of the impacts of pricing models on the service selection process, and
    b) the development of worst- and average-case performance optimization approaches for service-based workflows adapting major findings of network calculus and queuing theory.

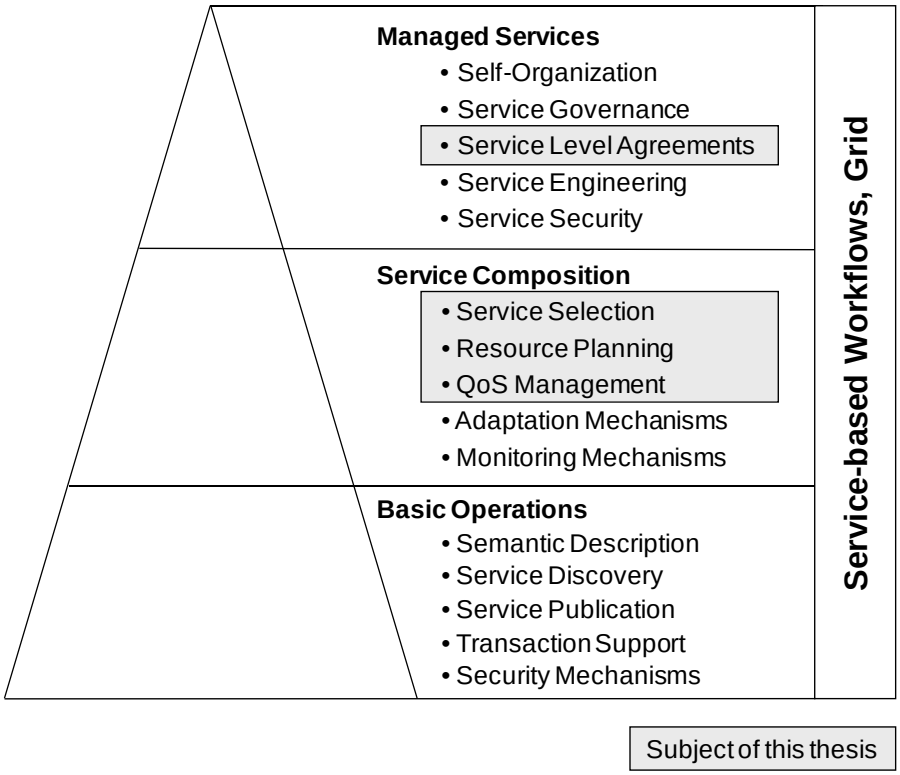


Figure 1: Research agenda for Service-oriented Architectures (according to [Pap03])

2. Development and evaluation of solution strategies for the resource planning of service-based workflows comprising
   a) a cost-efficient Multi Service Selection heuristic – MSS.KOM, a Single Service Selection heuristic – 3S.KOM, and
   b) an architectural Workflow Performance eXtension – *W*PX.KOM.

The following paragraphs present and discuss these contributions in detail:

This thesis focuses on a scenario comprising three involved roles: service provider, service requester, and service intermediary. A huge amount of service providers implies a large amount of service pricing models the intermediary has to deal with when providing workflow execution to his customers. Concerning the provider-intermediary relationship, several pricing models that are specified in the SLAs for charging service usage exist. The developed classification of pricing models for services serves as a basis for the developed QoS optimization approaches for service-based workflows. Several existing pricing models are classified into three high level types of pricing models: variable fee, fixed fee, and hybrid fee pricing models. Furthermore, the impact on the service selection process of services for each of those models is discussed and analyzed in detail (contribution 1a).

For the intermediary, acting as a service orchestrator and providing the service "workflow execution" to his customers, both an average-case as well as a worst-case performance analysis of service-based workflows are of high im-

portance. This enables the intermediary to create the service invocation plan based on an average- and a worst-case analysis concerning QoS requirements and service invocation costs. By using a service, the intermediary is faced with specific Costs of Service (CoS) as he has to pay a specific amount for service utilization.

In detail, the average-case performance analysis is conducted by the adaptation of major findings of queuing theory to the concept of service-based workflows. Mapping services to queuing systems facilitates the analysis of service utilization and other important QoS parameters of services as well as the analysis of the performance behavior of the entire workflow. Besides a straight average-case performance analysis, this consideration is extended by an optimization model that optimizes service utilization taking into account response time constraints by providing a solution with low service invocation costs. This approach represents a cost-efficient average-case performance optimization, avoiding a large increase in response time. In addition to the average-case performance analysis, the worst-case performance analysis supports the intermediary by the creation of service invocation plans for service-based workflows. For this purpose, major insights of network calculus are adapted to the concept of service-based workflows. Key concepts of network calculus such as arrival curve, service curve, backlog bound, and delay bound are mapped to the considered provider-intermediary scenario and the offered services. Furthermore, several optimization models are developed for cost minimization, delay minimization, and throughput maximization for service-based workflows in a worst-case consideration (contribution 1b).

Concerning service-based workflows with a high repetition rate and limited service execution capacities (resources), multiple (parallel) service invocations are necessary in order to be able to serve all incoming workflow execution requests. The creation of invocation plans comprising single and parallel service executions considering several QoS parameters and service invocation costs is referred to as the resource planning process of service-based workflows. In order to realize the resource planning process, approaches in the field of mathematical optimization, operating based on SLAs as well as on measured values such as response time, are used. In general, this selection process can be realized during run-time as well as at design-time. Due to varying service providers and varying provision of services over time, solving those optimization problems becomes time-critical. The analysis of related problems has proven that those optimization problems, with resource-constrained side conditions, are NP-hard [DD98], [ZBN+04], [Ber08]. Consequently, it is not possible to determine exact solutions for realistic scenarios at short computation times.

Coping with this challenge, a heuristic solution, referred to as MSS.KOM, is developed. The appropriateness of this heuristic is proven by an implementation and a detailed evaluation. For the evaluation of the heuristic a simulation environment is developed. In detail, the evaluation compares the solution of

the heuristic MSS.KOM with respect to solution quality and computation time (with varying side conditions as restriction strength, problems size, and QoS-CoS correlation of the test cases) with an exact solution and another heuristic solution 3S.KOM. The evaluation outlines the high performance of the developed heuristic solutions as solutions with a low computation time resulting in a very high solution quality (contribution 2a). In order to realize the resource planning process, an architectural extension for QoS management systems for service-based workflows, referred to as *WPX.KOM*, is proposed, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows (contribution 2b).

1.3 STRUCTURE OF THIS THESIS

The remainder of this thesis is structured as follows. Chapter 2 describes basics which are necessary to understand key concepts of this thesis. The architectural concept and the paradigm of SOA are introduced, as well as the involved roles and the specific characteristics. Furthermore, business processes and workflows, so-called IT-supported business processes, are introduced as well as QoS. In particular, the description focuses on QoS for service-based workflows and on SLAs that play an important role in relationships between service provider and service consumer. In addition, this section provides an introduction to pricing theory which is necessary for the understanding of Chapter 4, dealing with pricing models for services in detail.

Chapter 3 gives an overview on the considered scenario of this thesis. Especially, the considered roles and relationships between service provider, service consumer, and intermediary are presented. Furthermore, this chapter describes the point of view of this thesis.

Chapter 4 presents a detailed analysis of existing service pricing models. In addition, this chapter presents the developed in-depth classification of pricing models for services divided by fixed fee, variable fee, and hybrid fee pricing models and discusses the impacts of service pricing models on service selection.

Dealing with the average-case performance analysis of service-based workflows, Chapter 5 introduces basics of queuing theory and related work in this field of research. After specifying the system model, the adaptation of major findings of queuing theory to the concept of service-based workflows is presented. In particular, the utilization analysis of services, describing the average-case performance behavior of service-based workflows, is presented. Finally, this section provides an optimization approach facilitating the maximization of service utilization, providing a solution with low service invocation costs and considering further constraints such as response time.

Chapter 6 describes the worst-case performance analysis of service-based workflows, primarily introducing the basics of network calculus comprising

fundamental assumptions and implications such as the delay bound and the backlog bound. Furthermore, this chapter discusses the considered system model and presents an in-depth description of the adaptation of major findings of network calculus to the concept of service-based workflows. Especially, the choice of service providers is discussed followed by the description of optimization approaches for service-based workflows facilitating delay minimization, throughput maximization, and/or cost-efficient workflow execution in a worst-case scenario.

Chapter 7 describes a detailed resource planning approach for service-based workflows. After an overview on related work, the system model of this approach is presented. A detailed overview on the resource planning approach comprising instructions for the non-functional service property aggregation as well as the optimization model is followed by the description of the developed efficient three-step multi service selection heuristic MSS.KOM and the single service selection heuristic 3S.KOM. Furthermore, this chapter outlines the suggested architectural extension *W*PX.KOM for generic QoS management systems for service-based workflows, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows.

A detailed evaluation of the developed heuristics with respect to solution quality and computation time incorporating side conditions as the QoS-CoS correlation of the test cases, the restriction strength, and the problem size is presented in Chapter 8.

The thesis closes with a conclusion of the fundamental findings and gives an outlook on future work in this field of research.

2 BASICS

This chapter introduces the basic terms, definitions, and concepts used in this thesis.

2.1 SERVICE-ORIENTED ARCHITECTURE

The general outcome of the industrialization and consolidation in many industries leads to a dramatically increasing demand for agile and flexible business processes and business models. Furthermore, competitive markets intensify requirements in terms of cost-efficiency and performance characteristics of processes. Enterprises are forced to allocate and manage modular software artifacts at run-time. In this context, a certain QoS level as well as an effective business process management have to be established in order to meet customer requirements.

Especially in case of cross-organizational workflows, when services from internal as well as external partners are invoked in a single workflow, effective business process management is essential for reliable operations. The on-demand integration of external, loosely coupled services and the integration of internal legacy systems are provided by a SOA [Pap03], [PTDL06]. The realization of service-based workflows by loosely coupled, interoperable software artifacts (services) aims at empowering the invocation of distributed components in an event-driven or asynchronous fashion reflecting the underlying business process needs [PvdH07]. Thus, SOA is regarded as an approach to facilitate the needed flexibility and feasibility in aligning application landscapes to business-driven demands [Mah07]. The purpose of this architecture is to address the requirements of loosely coupled, standards-based, and protocol-independent distributed computing. Usually, the encapsulated functionalities, referred to as services, are well defined, self-contained modules providing standard business functionality. In addition, these services are independent of the state or context of other services. In general, a SOA should allow developers to overcome many distributed enterprise computing challenges including application integration, transaction management, and security policies [ACKM04].

SOA offers flexibility and agility through services, with a specific granularity, which can be executed or composed to other services or workflows. The general philosophy of a SOA as a design philosophy is independent of any specific technology, e.g., Web services. There are not many implementation restrictions for services. A SOA merely requires that services are described in a specific description language. The intention of a service is the representation of a specific functionality (service implementation) with a standardized inter-

face. Besides others, three fundamental characteristics for services in a SOA can be identified:

- Services have to be self-contained, i.e., the service maintains its own state.
- Services have to be platform independent.
- Services can be dynamically located, invoked, and (re-)combined.

Large applications can be modularized into services and a service user, independent from the operating system and from the programming language, may search for this service and invokes it into his own business process.

SOA has been mentioned the first time in literature by Gartner in 1996 [SN96]. MacKenzie et al. provide a comprehensive and useful definition of a SOA. SOA is "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations." [MLM+06]

Besides this definition, Channabasavaiah et al. describe a SOA by three core principles [CHT03a], [CHT03b]:

- All functions are defined as services. This includes purely business functions, business transactions composed of lower-level functions, and system service functions.
- All services are independent. They operate as black boxes; external components neither know nor care how they perform their function merely that they return the expected result.
- In the most general sense, the interfaces are invocable; that is, at an architectural level, it is irrelevant whether they are local (within the system) or remote (external to the immediate system), what interconnect scheme or protocol is used to effect the invocation, or what infrastructure components are required to make the connection.

Besides these definitions and characteristics, a useful characterization of a SOA, comprising other definitions used in this thesis is as follows:

- Usually, a SOA is based on self-contained services, i.e., involved services maintain their own state. Services have to be independent from the state or context of other services and communicate with each other, requesting execution of their operations in order to collectively support a common business task or process [PvdH07].
- The involved services are platform independent [PvdH07]. The result is the duality of service contract and service implementation, i.e., separation of technical/implementation-dependent data communication and

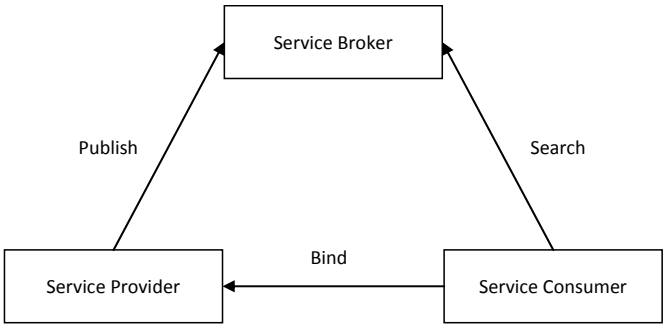


Figure 2: Roles in a SOA

universal professional functionality. Thus, SOA utilizes the principles of encapsulation and separation of concerns, known by object orientation, on a higher level of abstraction of application modules. The aim is to integrate services on a business process level. Consequently, fulfilling business functionality, services follow the concept of "input, processing, and output" instead of "everything is an object" [Zac05].

- A SOA facilitates that services can be dynamically located, invoked, and (re-)combined to other services with specific value-added functionality empowering flexibility in aligning enterprise information systems to the underlying business needs. The composition of service-based workflows can achieve a strong congruency of an enterprise's information system to its business processes. Additionally, by the use of service brokers, a geographic transparency is achieved, as services can be found and bound independent of their physical address or location [Pap03].

The concept of service-orientation represents a distinct approach for separating concerns. The logic for solving problems can be decomposed into a collection of smaller, related pieces.

2.1.1 *Roles*

A SOA environment typically involves three different roles. A *service provider*, a *service consumer/service requester*, and a *service broker* as depicted in Figure 2. A service in this context is a program or a software component, which can be used locally or via a network. Therefore the interface of this service has to be published for public access and the service description has to be stored in a machine readable format [DJMZ08], [KBS04].

A service provider implements a specific functionality as a service and offers this service on his platform. Furthermore, the service provider registers the service with its interface description at a specific repository which is operated by a *service broker*, so that the service consumer is able to find and invoke this service. This can be done at one registry or at more registries at the same time. Typically, the service provider is also responsible for the accessibility and the operation of the offered services as well as for authentication

and authorization purposes. The service provider not necessarily always implements the service. Instead, it is also possible that the provider uses several functionalities via a network and encapsulates this functionality, combines this with other functionalities to a value-added service and offers this as a so-called composite service to its customers. Independent of the fact, whether a service is developed and hosted locally or via a network from a third party, during service invocation the service provider is always responsible that a specific QoS level is met in order to provide a service at a negotiated QoS level to the service consumers. These QoS levels are usually specified in the Service Level Agreements (SLAs), representing a contract between a service provider and a service consumer (see Section 2.3.3).

In the case that a service consumer is searching for a specific service with a specific functionality, he can search in the repository of the service broker for a service with a specific functionality and with a specific QoS level and invoke this service at run-time into his workflow. In general, many repositories from various independent brokers may exist for service publication.

The task of the service broker is the intermediation between service consumer and service providers. The service broker stores the published services of the service providers in the repository and offers searching capabilities via an interface to the service consumers. After finding the service, the service consumer interacts with the service providers directly and invokes the service via a standardized interface without inclusion of the service broker. Therefore, standards for the interface definition and for the message exchange are necessary.

2.1.2 *Characteristics*

After defining typical roles in a SOA environment, this section focuses on SOA characteristics. In the literature several characteristics describing the concept of a SOA exist. Service-orientation can be characterized by the following key concepts:

- *Reusability:* Services have to be designed to be reusable, i.e., design standards have to be met that services are reusable in various contexts. Inter-application interoperability, easy composition, and the creation of utility services have to be supported. If an encapsulated logic of a service is useful to more than one service consumer, it can be considered reusable.
- *Service contract:* Service contracts should provide the service endpoints, the service operations, input and output messages supported by each operation, and rules and characteristics of the service and its operations. Furthermore, semantic information can be provided explaining how a service may accomplish a particular task. Due to the binding character of such a contract, it has to be carefully maintained and versioned after its release.

- *Loose coupling:* Services have to be loose coupled. The coupling of a service is a measure for its dependability on other involved services, i.e., the service acquires knowledge of another service while still remaining independent of that service. This can be achieved through the communication via messages, the existence of service contracts, and the interaction within predefined parameters. In a SOA, it is advantageous, if the dependency factor (a measure for the degree of loose coupling) is low.
- *Abstraction:* Services may act as black boxes, hiding their implementation details. There is no limit of the amount of logic a service can represent. Due to the encapsulation of the service the implementation details are invisible for the service consumer.
- *Composability:* Services can also act as a part of a service composition. This requirement has to be irrespective whether the service itself composes others to accomplish its work or is part of another composite service. This is supported by the concept of orchestration where a higher level service invokes other services and orchestrates them in order to fulfill a specific functionality. Therefore, also the operations have to be standardized with an appropriate level of granularity in order to maximize composition opportunities.
- *Autonomy:* Services have to be autonomous, i.e., dependencies to other services have to be eliminated. At the time of service execution, the service has the control of whatever logic it represents.
- *Statelessness:* The amount of state information the service has to manage has to be minimized, i.e., individual operations of a service need to be designed with stateless processing considerations. Statelessness promotes reusability and scalability in a SOA environment.
- *Discoverability:* It is very important that not only the specific functionality of a service is described, also metadata should be provided in order to guarantee that the service can be found in a repository. Discoverability also implies that there exists at least one repository where services are listed and can be found.

After a description of a SOA, the involved roles, and its characteristics, the next section focuses on business processes and workflows that can be realized by services.

2.2 BUSINESS PROCESS AND WORKFLOW

This section introduces the basic concepts of business processes and workflows including descriptions and definitions.

2.2.1 *Business Process*

In the past, organizations have been functional oriented focusing on division of labor and competencies that has proven not to scale up to nowadays' busi-

nesses anymore. Hammer propagates process thinking as a cross-functional view, focused on the company creating value by its real work [Ham90]. A business process is typically associated with a loan handling process in a bank, a claim processing in an insurance company, or an engineering development process [LR99]. Thus, business processes comprise an identification and sequencing of activities, tasks, resources, decisions, roles, and responsibilities across time and place that can be performed by several persons. The detailed order of the activities and the defined patterns are usually described in the process model. Usually, a business process has defined conditions triggering its initiation in each new instance (e.g., the arrival of a claim) and defined outputs at its completion. Formal as well as relatively informal interactions between participants can be involved in a business process whereas the duration of process execution may vary widely. A process may consist of automated activities, capable of workflow management, and/or manual activities. A business process represents a collection of activities that takes one or more kinds of input and creates an output that is of value for the consumer. Clear process-responsibilities have to be installed that have not been provided by the common functional organization structures [MW01].

A process can be organized and contained within a single organizational unit as an enterprise or it may span several different organizations, such as in a customer-supplier relationship. Furthermore, a business process can be spanned across enterprise boundaries. Its execution is not bounded to any organizational or geographic borders. Concerning globalization of value chains and business process outsourcing, cross-organizational business processes become increasingly important in reality [Rie03]. Also highly specialized corporations depend on the success of those value networks [LW04]. Therefore, multiple entities across different organizational units (e.g., customers, partners, providers) can be involved in those cross-organizational business processes. Those processes need to be continuously monitored, reviewed, altered, and streamlined in order to facilitate that the enterprises remain competitive [MW01]. The Workflow Management Coalition, as an international standardization organization that concentrates on processes, defines a business process as [WMC99]

"a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships."

An activity in this context is referred to as

"a description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution; where human resource is required an activity is allocated to a workflow participant."

A so-called process definition consists of many process activities that are logically related in terms of their contribution to the overall realization of the business process. An activity is typically the smallest unit of work which is scheduled by a workflow engine during process enactment (e.g., using transition and pre/post-conditions), although one activity may result in several work items being assigned (to a workflow participant). In addition, manual activities may form part of a business process and can be included within its associated process definition, but do not form part of the automated workflow resulting from the computer-supported execution of the process. An activity may therefore be categorized as manual, or automated.

With the concept of continuous improvement, an increasing of efficiency and effectiveness in an organization's business process is achieved. Hence, for business process reengineering purposes, business process models including an appropriate and common modeling language are essential in order to achieve clear understanding. This can enforce to identify improvement potential, such as potential for business process automation, elimination of unnecessary media changes, reduction of delays or assuring correct termination (e.g., identification of deadlocks). For this purpose several graphical notations for business process modeling exist, such as Event-driven Process Chains [KNS92] or Business Process Modeling Notation [WM08].

2.2.2 *Workflow*

The Workflow Management Coalition defines a Workflow Management System as

"a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications."

Software components in the Workflow Management System store and interpret process definitions, create and manage workflow instances as they are executed, and control their interaction with workflow participants and applications. Such systems typically provide administrative and supervisory functions such as to allow work reassignment or escalation, plus audit and management information.

A workflow is defined by the Workflow Management Coalition as

"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

The process definition specifies the automation of a business process that identifies various process activities, procedural rules, and associated control

data used to manage the workflow during process execution. Many individual process instances may be operational during process enactment, each associated with a specific set of data relevant to that individual process instance. A loose distinction is sometimes drawn between production workflow, in which most of the procedural rules are defined in advance, and in an ad hoc workflow, in which the procedural rules may be modified or created during the operation of the process. A workflow does not necessarily represent the entire automatization of a process; instead, technical resources are used for workflow implementation [JBS97], [vdAH02], [Wes07].

2.3 QUALITY OF SERVICE

This section presents the used definition of QoS in this thesis. Furthermore, it provides an overview on QoS for service-based workflows as well as an introduction in SLAs.

2.3.1 *Definition*

The related literature in research does not provide a common definition for the term QoS. Several existing definitions are strongly dependent on the application scenario. In this thesis the definition of Schmitt is used in order to describe QoS [Sch01]:

"QoS is the well-defined and controllable behavior of a system with respect to quantifiable parameters"

Usually, this kind of QoS is treated as the QoS from a technical point of view, whereas QoS also subsumes non-technical aspects such as the reputation of a service provider. In practice, a service is always described with the help of QoS parameters that are oriented at the QoS parameters of network services. The ISO-Standard 9126 provides quality criteria for the review and evaluation of software products. This standard offers the following six categories for the measurement of software quality:

- *Efficiency*: Determines the relationship between the level of performance of the software and the amount of used resources (time behavior, resource behavior).
- *Usability*: Determines the learnability, understandability, and operability of software products.
- *Maintainability*: Determines the stability, analyzability, changeability, and testability of software products.
- *Portability*: Determines the installability, replaceability, adaptability, and conformance of software products.
- *Functionality*: Determines the suitability, accuracy, interoperability, compliance, and security of software products.

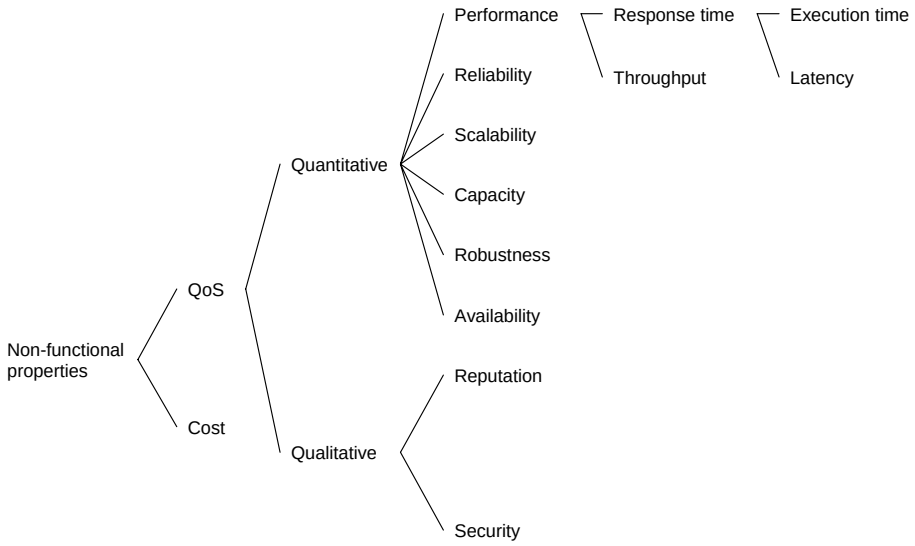


Figure 3: Classification of QoS parameters

- *Reliability*: The reliability of a software product provides information about maturity, fault tolerance, and recoverability.

These six characteristics are often used when measuring the quality of software products, but for determining the QoS parameters for services in general, the proposed parameters have to be adapted and extended to services as presented in the following section.

2.3.2 *Quality of Service for Service-based Workflows*

The proposed definition of QoS by Schmitt was originated in the field of communication networks and multimedia and is also adapted to Peer-to-Peer systems [Ste00], [SN04], [Kov09]. In the field of multimedia and especially in the field of audio and video a lot of research exists for determining QoS in such scenarios where QoS is very important in order to guarantee, e.g., a good quality for video streaming [MT04], [OSS02], [Hec06a], [Hec06b], [Zin05], [ZSS05].

The research in the field of QoS for service-based workflows represents a new research topic. Service providers have to guarantee a specific QoS level to the service consumer and have to implement a QoS management. Without such QoS agreements the service consumers are not willing to request services, because the risk of performance degradation is too large and the own application in which external services are integrated can be affected as well [RM08], [ERM09]. In this context the QoS-aware selection of services becomes mandatory and business critical when invoking services from third parties and composing them to a workflow [Ber08], [BGR+07], [CPEV05b], [CPEV08], [CSMA02].

The quality criteria of the ISO-Standard 9126 can be applied and adapted to services. The World Wide Web Consortium (W3C) specifies QoS requirements for Web services [LJL+03]. Besides this, many different views of important QoS

parameters for Web services exist in the literature [KKL03], [PSL03], [Ran03]. Figure 3 presents an adapted classification of QoS parameters according to Berbner [Ber08]. This classification of QoS parameters includes performance, reliability, scalability, capacity, accuracy, reputation, availability, and security. The characteristics of the QoS parameters are as follows:

- *Performance [Men02], [GKG03], [ZBD+03]*:
  Service performance indicates how fast a service request can be completed. Performance in this context subsumes the four main components: throughput, response time, execution time, and latency. The throughput indicates the number of service requests a service is able to serve within a certain time interval. The time required to complete a service request is referred to as response time that can be divided in:
  - Execution time at the service provider: time spend for processing the request.
  - Latency: the round trip delay between sending a request and receiving the response (transport time in the network for data transport).
- *Reliability* [GKG03]:
  Reliability in the context of services describes the probability a service is able to offer its functionality in a certain time interval. It can also be described as the ability of a service to perform its required functions under stated conditions for a specified time interval. From the point of view of the service requester, reliability determines to which degree the service maintains its service quality. It can be described in failures per minutes, days, weeks, or months. These failures may include delayed messages, lost messages or services that are not available at present.
- *Scalability* [LJL+03]:
  The capability of increasing the computing capacity of service provider's computer system and system's ability to process more user requests, operations or transactions in a given time interval is determined by the scalability.
- *Capacity* [ESR+08]:
  The execution capacity of a service is the number of possible service requests within a certain time interval with guaranteed performance. The service provider guarantees the service execution up to a specific execution capacity to its customers in a specific time interval.
- *Robustness* [LJL+03]:
  The degree to which a service operates as specified by the service provider even in the presence of invalid, incomplete or conflicting inputs is characterized with the term robustness. The objective of a service designer should be that a service is still working even if parameters are incomplete at the service request invocation.
- *Reputation* [MS02a], [MS02b]:
  In general, reputation represents the opinion of the public towards one

individual or a group of people. In the case of services, the opinion of the service requesters towards a service provider is the reputation of the service provider. Past experiences as well as current experiences with the service provider affect the reputation. If positive experiences or positive references have been collected in the past, a service provider can be rated with a high reputation. The service consumer can therefore decide whether he will source services from this provider or not. Reputation represents the sole QoS parameter that cannot be specified by the service provider. Instead, this measure is dependent on experiences of others in the past.

- *Availability* [LJL+03], [Men02]:
  The availability of a service is the probability that the service can be invoked and addressed immediately by a service consumer, i.e., the service provider is able to provide his offered functionality immediately. In a workflow consisting of several services, a high availability is mandatory in business critical workflows that have to be executed very fast and very often. The availability is the probability that an offered service can be used by a service consumer.

- *Security* [KC08]:
  Concerning service-based workflows, security becomes a major issue. In such a scenario, also services from external partners can be sourced. It has to be ensured that the communication between service provider and service requester is secure and that the interfaces do not represent an additional point of attack. Basic security mechanisms are mandatory for secure service-based workflows as, e.g., authentication, authorization, integrity, confidentiality, responsibility, integrity, and encryption [MGK+09]. There are also several specific requirements on the security for service-based workflows. The "SOA-Security-Kompendium" of the Bundesamt für Sicherheit in der Informationstechnik (BSI) examines several important security issues concerning SOA [Bun08]:

  – *Authentication*:
    Authentication supports the identification of several entities in a system. In the considered scenario it supports the identification of the service requester, the service provider, and the service itself. In contrast to monolithic systems, where authentication is often ensured via a central login, in a SOA, the authentication of many services has to be supported. Due to a variety of service providers, several authentication mechanisms have to be handled in such a scenario.

  – *Authorization*:
    Authorization assures that the entities in a system have the eligibility to use a resource. In the considered scenario each service requester has to be authorized to use a certain service. Several roles and rules for the services have to be implemented due to the increasing number of policy enforcement points (authorization of services and service requesters).

- *Integrity*:
  Integrity determines the reliability of information and resources. If the integrity of a message is not assured, exchanged messages between service provider and service requester can be modified. Integrity ensures that the sent information between both parties is not falsified by a third party.
- *Confidentiality*:
  Confidentiality can be achieved by using several encryption mechanisms in order to ensure that authenticated and authorized entities have access to the exchanged information. Encryption at message-layer assures that a third party cannot read exchanged messages.
- *Non-repudiation*:
  Due to the fact that service-based workflows are also used in business scenarios with an interaction of several parties, non-repudiation is an important security issue. Non-repudiation ensures that each entity in the system is responsible for its activities and can be identified uniquely.

Besides these QoS parameters, cost is an important non-functional property of a service. Costs of Service (CoS) have to be taken into account when selecting services from a subset of services with the same functionality, but with different QoS parameters. Typical pricing models for services are, besides others, pay-per-use or flat-rate models. A detailed discussion of pricing models and the impact on service selection are presented in Chapter 4.

2.3.3 *Service Level Agreements*

In the context of cross-organizational service-based workflows, i.e., workflows with a lot of involved internal as well as external service providers, an effective management of the functional as well as the non-functional service properties is crucial. The concept of SLAs copes with this challenge by contracting the offered and requested services with respect to its QoS, cost, and functional parameters. Thus, SLAs represent formal contracts defining the offered functionality, the QoS level, and the costs of the service [CCDS03], [ZA04a]. Besides the offered utility of the service, penalties can be negotiated between service providers and service consumers and can be contracted in the SLAs in case of SLA violations [ULMS08]. Recent research develops distributed monitoring mechanisms in order to detect SLA violations and to react to the deviation very early [RBHS07], [RES+08], [RMNS08]. Furthermore, cooperation mechanisms for SOA environments are proposed [MBR+09], [MBP+09]. Usually, a SLA comprises the following elements [LKD+03b]:

- A description of the involved parties, including their roles (provider, consumer).
- A specification of the service level parameters with predefined metrics that measure those items.

- The determination of the service level objective considering the service level parameters.

The Web Service Level Agreement (WSLA) Language specification has been developed especially for Web services [LKD+03b], [LKD+03a], [KL03]. It comprises provider-consumer agreements and further obligations of the involved parties in such a relationship. The WSLA language is based on XML; it is defined as an XML schema. A WSLA document (referred to as a WSLA) defines assertions of a service provider to perform a service according to defined QoS levels (e.g., throughput, response time) and actions that have to be performed in case of SLA violations, such as a notification of the service consumer. Furthermore, WSLA defines which party monitors the services, how metrics are measured and aggregated, and interactions among the parties. WSLA can be used by both, provider and consumer and is extensible in order to include specific types of operation descriptions, e.g., using WSDL to describe a Web service operation.

WSLA supports the reporting of SLA deviation, but it does not treat a detailed specification of recommended reactions in case of SLA deviations. For this purpose WS-Re2Policy, which is based on the WS-Policy-Framework of the W3C, supports the modeling of requirements and reactions to deviations from the perspective of a service consumer [RMNS08].

2.4 PRICING THEORY

The concept of SOA facilitates a large variety of application scenarios including several relationships and roles as mentioned in Section 2.1.1. Besides the intra-organizational scenario of a SOA also cross-organizational scenarios can be assumed, in which services are also sourced from external partners. Business processes and workflows in such a scenario can be cross-organizational with a distinct impact on these roles and the interaction between the different parties. Services can be offered from an internal service provider but also by a third party, affecting the service provider, the service requester, and the requester-provider relationships. The negotiated contracts between these roles are usually described in dedicated SLAs that define rights and duties of the provider and the consumer. The service provider is responsible for service execution and the offered functionality by a distinct QoS level. The service consumer has to pay for each service invocation a specific fee [ZA04b]. This fee varies due to different pricing schemes of the providers. Such schemes for charging the service usage can be described by pricing models that are examined in detail with respect to the specifics of service pricing in Chapter 4.

Research in the field of pricing models becomes an important research issue especially in the field of SOA and Service-oriented Computing where different providers may offer their services with varying pricing schemes. Those schemes affect the provider-requester relationships to great extent and cause several peculiarities. Pricing schemes in which the service execution is offered

in a distinct time period (fixed fee pricing) imply a strong binding between provider and requester in contrast to pay-per-use pricing schemes where the consumer may switch between providers as it is beneficial. Besides this, the service market can be influenced as well when assuming long-term contracts between provider and requester or short-term contracts or a mix of both. In particular, those relationships may be short, mid, or long-term. The determined prices to be charged and the designed pricing schemes affect the market behavior according to the principle functions of price [TA06].

The achieved price and the underlying pricing scheme of the service provider affect the IT resource allocation and the resource provisioning of the own resources as the provider has to provide an infrastructure for service execution with several assumptions of a demand for service execution. This demand, affected by the pricing model, has to be forecasted with past values that is not always easy to implement. Thus, the management of IT resources of the provider is an important issue in order to stay competitive and operate cost-efficient.

In order to manage the provided services and the underlying IT infrastructure, the set price for service consumption becomes an important parameter for the management of a service provider's IT infrastructure. A service, as a provided product may also have limited execution capacities within a limited time interval and is therefore treated as a *service resource*. In order to assign these service resources, several authors propose price-based mechanisms to avoid overload and to guarantee fair or optimal assignment of these services to users, i.e., requesters (e.g., [LZR03], [YGL06]). Lin et al. state that purely technology-based approaches for resource allocation are subject to serious limitations and legitimate the application of price-based approaches that aim on market-related notions of efficiency [LZR03].

As mentioned earlier, different pricing schemes may have significant impact on the service resource planning approach, i.e., the planned invocation of limited service resources as discussed in Chapter 4. Thus, the following provides a theoretical foundation of pricing models and the fundamental terminology of pricing theory.

The price of a good, as a result of negotiation between two or more parties, is a result of a complex decision process with a set of factors that affect this process. Rowley classifies these sets of factors into organizational, customer, and market factors [Row97]:

- *Organizational* factors are related to the resources and objectives of an enterprise with regard to the existing product portfolio.
- *Customer-related* factors directly influence decisions on price setting over the service demand and its interdependency to the price.
- *Market-related* factors are concerning the market environment and the competition between service providers.

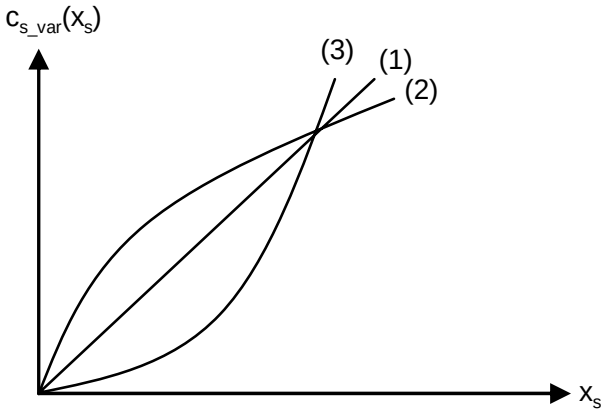


Figure 4: Characteristics of variable costs

Typically, the bargaining power of a service provider is strongly affected by the supply and the price of services that are offered by competitors of similar or substitute goods. Besides these, also social, technological, economical, and political factors directly affect the price setting [Homoo].

The expense of a good or a service, i.e., the costs for service production, is the most important internal determinant of a price. Costs in this context can be seen as priced consumption of resources for maintaining operational reliability and production of goods and services [DS05]. Assuming an offered service S of a service provider, its costs c_s are defined as the sum of expenses $e_{s,i}$ spent for any involved resource for production $r = 1, ..., R$. Furthermore, these costs can be classified in expenses that vary with the amount of produced goods and those that are invariant to the level of operation as depicted in Equation 2.1. The first are referred to as *variable costs* c_{var} and the latter are referred to as *fixed costs* c_{fix} that are invariant to the amount of service usage x_s. Considering variable costs, several relations between variable costs $c_{s\_var}(x_s)$ and the amount of service usage x_s exist. Figure 4 presents these relations as linear (1), on a diminishing scale (2), and on a progressive scale (3).

$$c_s = \sum_{i=1}^{R} e_{s,i} = c_{s\_var}(x_s) + c_{s\_fix} \tag{2.1}$$

As presented in Equation 2.1 the summation of variable costs and fixed costs are the *total costs*. Usually, the fixed costs of an involved resource r are also assignable to a single unit of x_s.

A major objective of a service provider is to generate profit with the produced goods (services) by selling them to its customers. The overall profit p_s of an organization for one unit is the sum of income i_s less the sum of expenses c_s as denoted in Equation 2.2.

$$p_s = i_s - c_s \tag{2.2}$$

The price of a good determines the likeliness of a customer to purchase, i.e., the enterprise will set the price in dependency to the expected demand.

For an enterprise, the price-demand function (functional relation of price p and sold units x) and its cost function (functional relation of costs c and produced units x) is mandatory in order to determine an optimal price in terms of profit maximization. Besides several cost curves, a variety of price-demand-curves exist. In these cases, the demand can be linear, multiplicative, exponential, semi-logarithmic, or stepwise linear in relation to the price. These price-demand-curves reflect the aggregation of all individual consumers' willingness-to-pay, i.e., the maximum amount of money they want to spend for a specific good (service).

After a detailed description of the basics of this thesis, the next chapter presents an overview on the scenario.

3

SCENARIO

The comprehensive scope of this work is the *Internet of Services* [JRS08]. The Internet of Services assumes the existence of a multitude of services with a specific functionality, which can be managed and listed in central or decentralized repositories comprising search and discovery abilities for the services. Furthermore, the Internet of Services supports SLA negotiation, service monitoring, service pricing, and service billing. The involved roles in the Internet of Services are the *service provider*, the *service consumer*, the *service intermediary*, the *service executor*, the *platform host* as well as a *service marketplace*. The first two roles offer services and request service execution respectively. The service intermediary, acting as a connector between service consumer and service provider, creates composite services or composes workflows out of a multitude of services and offers these with value-added functionality. The role at which the service is executed is not necessarily the service provider or the platform host. Instead, service execution can also be accomplished by a third party, the so-called service executor. The platform host is primarily responsible for the provisioning of the platform (search, discover, service registry), but it can also adopt the role of the service executor, service provider, or any other arbitrary role. The available services are offered on a big scale on a so-called service marketplace.

In particular, the scenario of this thesis focuses on cross-organizational service-based workflows. Assuming a large set of workflow consumers, requesting a specific workflow at a specific time, workflow provision to the consumers at demanded QoS properties has to be ensured. An arbitrary intermediary has to be able to determine optimal service invocation plans for all workflow consumers. As presented in Figure 5, all workflow consumers request workflow executions. For workflow execution, internal services as well as external services from third parties can be invoked and composed to a workflow [ZBN+04]. In order to be able to execute a huge amount of simultaneous workflow execution requests, an intermediary ensures serving of all requests at the demanded QoS levels. The considered scenario comprises three basic roles with several tasks as described in the following:

- Service provider: Offers services with a specific functionality at varying QoS levels such as response time, service execution capacity, and specific costs (several cost models are possible).
- Workflow requester: Requests a specific workflow execution with specific QoS and cost requirements.
- Intermediary: Monitors, aggregates, and prioritizes all workflow execution requests, and implements and applies a resource planning process [EEM+08].

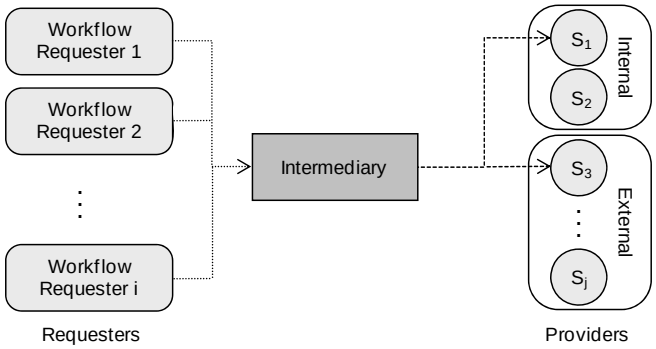


Figure 5: Research scenario

The considered generic workflow in this scenario is a recurring workflow with a sequential order of tasks and a high repetition rate (see Figure 6). This kind of workflow can be an arbitrary workflow such as a credit process or similar workflows with a high repetition rate. In this setting, the existence of many workflow consumers, requesting a specific workflow execution, and many service providers offering a large variety of services, are assumed. The intermediary is responsible for the detailed resource planning of workflows, i.e., the selection and invocation of services while taking into account several constraints such as limited service execution capacities and several pricing models.

The intermediary may also be another service provider dynamically utilizing limited resources, such as processing performance, network bandwidth, and in particular, existing services to provide a novel composite (value-enhanced) service to his customers (service requesters). An intermediary combines existing services, e.g., from other service providers, to an aggregate service (respectively service-based workflow). The intermediary's major objective is to realize the allocation of available services (resources), in order to minimize costs by being compliant to the QoS requirements and the amount of service execution requests. This procedure is referred to as the resource planning of service-based workflows. Resource planning in this context does not treat processing performance of the processing unit or server workload; instead, the term resource focuses on services having limited resources in terms of service execution capacity.

The considered workflow can be decomposed into several basic activities, which can be executed by specific services S_i $(i = 1, ..., n)$, fulfilling the required functionalities of the considered tasks. These tasks are, depending on the granularity of the task decomposition, decomposed in a way that there exist ranges of services $S_{i,j}$ $(j = 1, ..., m_i)$ which fulfill the required functionality of the considered task i. The intermediary is responsible for the execution of a workflow. He has to handle all execution requests and has to ensure that all requests can be served within a specific time period. Furthermore, the objective of the intermediary is to serve all incoming workflow execution requests at minimal costs and demanded QoS properties. The workflow itself can be

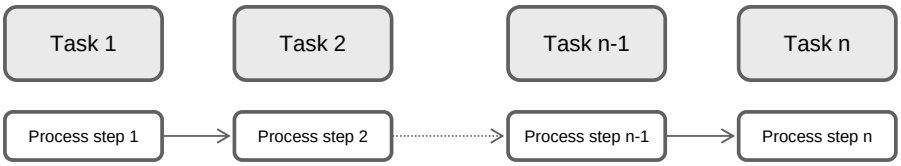


Figure 6: Sequential workflow

executed by invoking several services and composing them to a workflow. For each task, the intermediary has to choose the appropriate service that fulfills the required functionality, which is described in the SLAs. In the considered workflow consisting of n different activities the intermediary has to ensure that task $i (i = 1, ..., n)$ has to be executed before task $i' (i' = 1, ..., n)$ if $i < i'$. The intermediary has to create an invocation plan in order to use the execution capacity of each service optimally, i.e., selecting the services in the most efficient way.

The setting and the relationships between the different roles are presented in more detail in Figure 7. As a simplification, one workflow requester represents several workflow requesters in the scenario, requesting workflow execution for one or more than one workflow to the intermediary. This request is made with specific QoS requirements, a specific deadline, and specific cost associations. The incoming workflow execution requests can further be aggregated and prioritized by the intermediary in order to create priority classes. Assuming several priority classes such as high, middle, and low priority requesters, optional invocation plans with several QoS levels have to be specified. Considering large amounts of incoming workflow execution requests, multiple services have to be invoked in parallel and an effective resource planning becomes crucial in order to avoid performance degradation. The challenge is to determine cost-efficient invocation plans which fulfill the QoS demands and guarantee the feasible execution of all workflow execution requests.

The intermediary is faced with a large service market with a broad variety of services with specific functionalities. A multitude of services with the same functionality are available on a commercial market. Here, supply and demand meet in an ad hoc manner, resulting in competition and utilization of market efficiency. Such a scenario offers the potential for an intermediary to decide the procurement of services at run-time. Dynamical, run-time planning of resources according to specific QoS requirements, such as execution capacity and response time, can be enabled. Several pricing models with specific influences on the service selection process as exhibited in Chapter 4 can be assumed and applied by the service providers. Each provider-intermediary relationship is attributed with a distinct pricing model, whereas we assume that for one considered scenario only one pricing model is possible.

The number of planned service invocations have to correspond to the amount of workflow execution requests and to the QoS requirements of all workflow requesters. Furthermore, profitability is important for the intermediary in or-

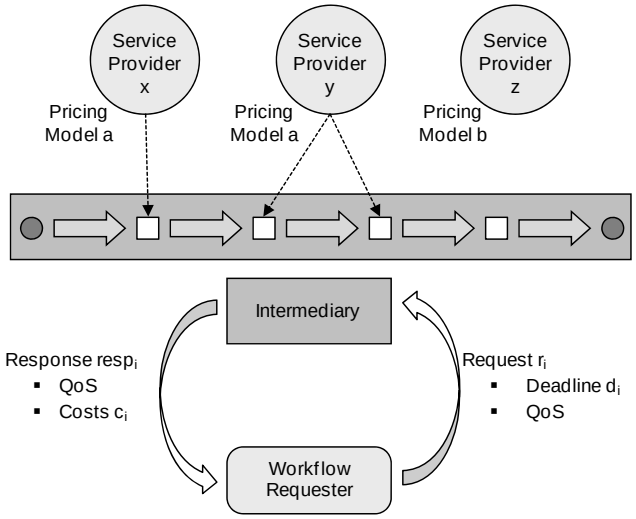


Figure 7: Overview on the scenario in detail

der to reduce costs and operate efficiently. Consequently, the intermediary has to plan the selection process at minimal costs. The intermediary is charged for the selected and invoked services, as well as he can charge the workflow requesters for the entire workflow. The benefit for the intermediary in this setting is the difference between the fee charged to his customers for workflow execution and the costs for service invocation. The intermediary can adopt two different roles associated with two different views as described in the following:

- Consumer view: The intermediary is confronted with several pricing models of the service providers and has to manage and implement an efficient resource planning process.
- Provider view: The intermediary charges his customers with the help of a specific pricing model for the provision of the requested workflow.

The problem of charging his customers, i.e., workflow requesters, with which pricing model is out of scope of this thesis and will not be considered in the following; instead this thesis focuses on the first scenario, the consumer view. As QoS and CoS are correlated, the correlation can be positive or negative depending on the characteristics of the QoS parameter. A higher QoS level (e.g., shorter response time) of a service usually implies higher costs. Concerning profit maximization, the intermediary is faced with a trade-off between QoS and CoS. Thus, the following sections deal with different aspects how to optimize several QoS properties with several constraints in order to stay competitive and operate cost-efficiently. The minimization of costs in compliance with the quality restrictions of the users results in profit maximization under restrictions. Resource allocation, in this thesis, aims at the allocation of available resources, that the resulting profit is maximized while users' qualitative requirements are met.

Traditional IT Outsourcing	Utility Computing
Long-term contracts	Ad hoc, Short-term contracts
Fixed expenses, Periodical payments	Variable expenses, Pay-per-use
Risk of lock-in high	Risk of lock-in reduced

Table 1: Traditional IT Outsourcing versus Utility Computing

The invocation of services from third parties, i.e., external providers, causes several different provider-requester relationships. The relationships have a wide range in their intensities as they range from ad hoc usage of a market for single transactions and static contribution to a long-term contract. Traditional *IT Outsourcing* is usually based on a long-term contract between an organization and an outsourcing provider. In contrast, *Utility Computing* in this context stands for advanced development of outsourcing with a more dynamic character of the provider-requester relationship. Those short-term contracts have to be negotiated in a short period of time in order to reduce the risk of being provider dependent or constrained by long-term contracts. In case of long-term contracts, it is not beneficial to switch between providers because switching to other providers can cause sunk costs. Sunk costs are strongly dependent on the pricing models used by the service provider (see Chapter 4) and may restrict the service requester's freedom of choice, as switching the provider can be uneconomic. In an Utility Computing scenario, the intermediary (acting as a service requester) is in a more flexible situation, being able to select the most appropriate service at any time if the pricing model of the service provider allows this. In contrast, traditional IT Outsourcing provides a specific certainty for both, the outsourcing enterprise and the provider by using long-term contracts. Table 1 presents a detailed comparison of the peculiarities of IT Outsourcing and Utility Computing. The motives for the implementation of Utility Computing are as follows:

- Maximizing economies of scale and maximizing the learning curve effect by benefiting from know-how of specialized service providers.
- Minimizing the risk of own development by having the flexibility of acquiring state-of-the-art solutions from external providers.
- Splitting fixed costs into variable costs by using, e.g., pay-per-use pricing models.

The use of services and the implementation of a SOA facilitate the demanded reusability and exchangeability that is necessary for the realization of Utility Computing. Instead of long-term agreements, contracts can be negotiated faster and have a short-term character. Different pricing models may have different effects on the trade-off between QoS and CoS. The discussion of pricing models and their dynamics concerning resource planning are the objectives of the considerations presented in the next section.

4

PRICING MODELS FOR SERVICES

Multiple service providers, using several service pricing models, exist in the considered scenario. Hence, this chapter presents an overview on service pricing models and the coherency of Web services and information products in general. Furthermore, a developed pricing model classification as well as the impact of pricing models on service selection is presented [EEP+09].

4.1 WEB SERVICES AS INFORMATION PRODUCTS

Pricing schemes and pricing models significantly affect the relationship between the intermediary and the service provider. Service pricing in this context and the analysis of the impact on resource allocation and service invocation requires an extensive examination of pricing models as services can be considered as information products having a special cost structure. Specific cost structures and accounting models in Peer-to-Peer systems are discussed in detail in the work of Liebau [Lie08].

An information product can be described as a non-material good, satisfying specific needs, which is developed and distributed with the help of information technology (e.g., Web services, telecommunication services). Information products usually cause significant first copy costs (fixed costs) for investments and research and development and cause only low marginal costs of reproduction [SV99]. The production of a further portion of the good is not expensive in contrast to the development and initial production of this good.

Services, and especially Web services, represent digital goods, which can be easily reproduced and distributed digitally without any physical storage medium. Thus, the variable distribution expenses become very small and the relation of fixed costs to variable costs becomes very high, resulting in vast economies of scale. Because the higher the relation of fixed costs to variable costs, the more significant the total average costs per unit decrease with additional units produced (and sold). Competitors of a specific service provider with a high market share usually also have less costs per unit sold as others. Furthermore, smaller production costs (costs per unit sold) result in more profit or the ability of sooner price reduction. In case of the latter, the market share will increase when everything else remains unchanged. This will again intensify the decreasing of unit costs with the respective results on earnings that is usually denoted as increasing returns [Art99].

Considering a market where products have a high ratio of fixed to variable costs, only those providers which are able to reduce their costs per unit very fast can compete with the others. Providers with a high market share already

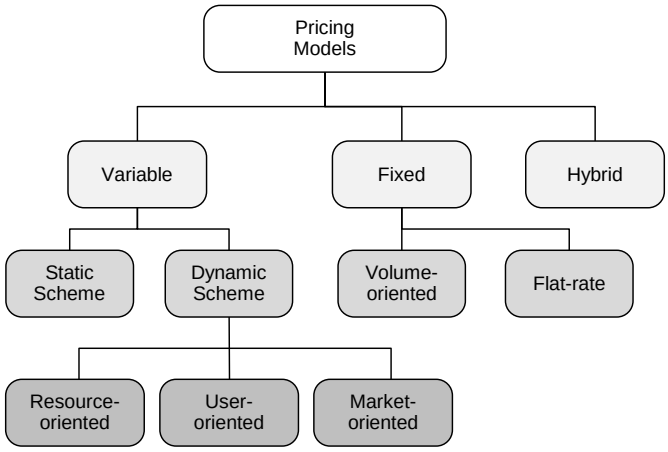


Figure 8: Pricing models for services

reduced their costs per unit sold and can offer their products less expensive at the same QoS level. This relation in the cost structure is of high importance especially for the competition of service providers in service markets. In general, providers of information products have to reduce their high fixed costs while avoiding competition to lead prices towards marginal costs of reproduction (zero). In a competitive market, commodity products are priced close to marginal costs of production in the long run, i.e., (Web) services would be priced close to zero and thus, without contribution margin. Therefore, providers, offering information products, cannot apply cost-based pricing, as it would not allow recovering of investments for the development. Also pricing according to competitors inhabits a high risk, as it reinforces ruinous price competition that only highly capitalized participants can survive in the long run. In this context, service or product versioning is introduced, pointing at customer differentiation in terms of offering several versions of a product. Differentiation criteria in this context are non-functional parameters such as delay, response time, availability, and service execution capacity, facilitating the adaptation of price to the specific willingness-to-pay for several consumers.

Relevant for the optimization approaches depicted in Chapter 5, 6, and 7 is, according to the theory of information products, that service providers offer services in classes with different QoS levels as a form of second degree price discrimination based on product quality. This is specifically useful for services, which are offered in a cross-organizational scenario to several internal as well as external service consumers, as degrading QoS properties imply the implementation of services with the same functionality but with several QoS levels, which is not expensive in the realization in practice.

4.2 PRICING MODEL CLASSIFICATION

Several pricing models for services exist and can be used in the considered scenario. The following outlines different pricing models and provides a classi-

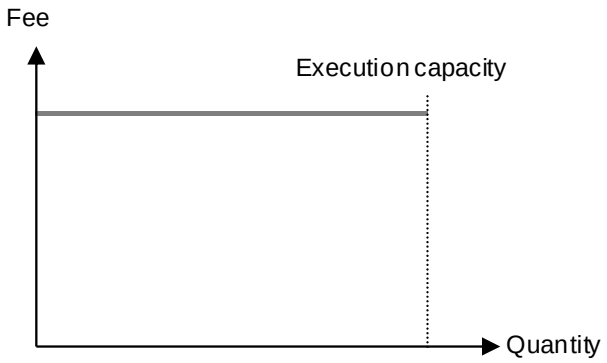


Figure 9: Volume-oriented pricing model

fication of pricing models for services. Usually, the cost structure of a provided service by a service provider has high fixed costs for development and variable costs for providing and maintaining services according to QoS requirements. Considering pricing models, fixed fee, variable fee, and hybrid fee pricing models are differentiated at the highest level of pricing models as shown in Figure 8.

4.2.1 *Fixed Fee Pricing Model*

In a scenario where the service provider applies *fixed fee pricing*, the service requesters are charged by a fixed fee, usually granting unlimited access to the service for a certain time period independent of the amount of service execution requests (*flat-rate model*). The other pricing model limits the access to the service by a specific amount of service execution requests within a specific time period. This form of service charging is a type of bounded fixed fee pricing models and is referred to as *volume-oriented pricing model*. This type of pricing model is especially examined in Chapter 5 and 7. Using a service, charged by a volume-oriented pricing model, the service intermediary, acting as a service requester, can invoke the service a limited amount of times up to the maximum execution capacity in a specific time period as described in Figure 9. Flat-rate pricing models usually imply long-term contracts whereas volume-oriented pricing models have shorter contract periods.

Because of using pricing models in which the service usage is charged by a fixed fee, usually the consumers are not faced with the true marginal costs of service production and distribution. Instead, this often leads to inadequate usage of narrow resources [AC01]. On the one hand, the services can be overdimensioned with respect to the theoretical service execution capacity within a specific time period, resulting in high fixed costs for the service provider. It is not necessarily guaranteed that the service provider can reduce his high costs to low costs per unit sold due to the fact that the service is not necessarily used up to its designed upper bound for service execution. On the other hand, a high amount of workflow execution requests and peak loads respectively may exceed the service execution capacity and some requests cannot be pro-

cessed, resulting in financial risks and a decrease of a provider's reputation in the long-term. Furthermore, using a fixed fee pricing model, the service intermediary, is faced with a high obligation to the service provider within the flat period. Switching the service provider within this time period can turn the flat fee into sunk costs and is therefore not as easy as in case of a pay-per-use model.

As a result it can be deduced, that fixed fee pricing models usually decrease a consumer's flexibility in switching service providers and cause a higher binding between service provider and service consumer. In contrast, these pricing models have the advantage, using an intelligent service invocation plan, the service consumer may also benefit from this approach. Concerning a volume-oriented pricing model the service consumer's interest is the invocation of as many services as possible within the specific time period up to the defined service execution capacity in order to reduce the costs per invocation (costs per unit) and to utilize the services efficiently. An optimization approach for this pricing model is depicted in detail in Chapter 5 and 7.

4.2.2 *Variable Fee Pricing Model*

A well-known and widely used pricing model is the *variable fee pricing model* (also known as pay-per-use). The service provider charges the service consumer with a fee, which depends on the actual usage of the service. The fee directly correlates with the amount of service execution requests. The fee y is defined as a function $y = f(x)$ of the supplied quantity x. Hereby, the function $f(x)$ can be of any continuous or discontinuous shape. Moreover, besides the quantity x of supplied service, any other external variables (e.g., technical conditions as the current server usage or other market prices) can be determinants of the fee (i.e., $y = f(x, v_1, ..., v_n)$). In this case, the relation between fee y and quantity x is subject to variances caused by other determinants v_i. According to this, we distinguish between variable fee pricing models with *static scheme* and *dynamic scheme*.

The main difference of these two models is the time at which the fee for service invocation is assigned, i.e., a fee y induced by any quantity x can always be determined a priori. Usually, in case of a service usage with a static scheme, the fee per request is known in advance. This is beneficial for the service consumer in order to be able to calculate the overall service invocation costs in advance for creating a detailed invocation plan, as the prices for any number of requests do not change over time. A very common and easy to implement pricing model with static scheme is the one-dimensional, linear usage-sensitive pricing scheme with a fee equal to $y = f(x) = a * x$ [SRL01]. The service consumer is charged for each service invocation a fixed fee that is time and volume invariant – this allows switching between service providers very easily due to the loosely coupled character of the provider-requester relationship.

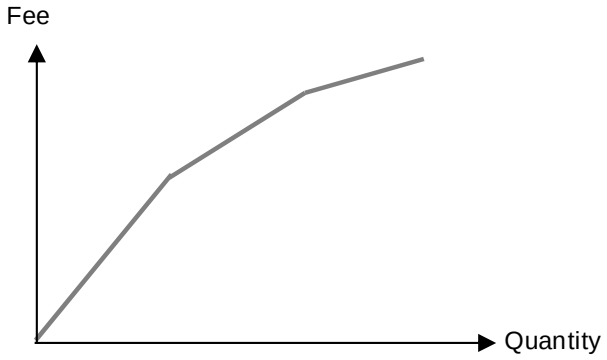


Figure 10: Non-linear pay-per-use pricing model

As a disadvantage of this model, large-scale excessive service users are penalized because there is no incentive to accumulate service execution requests and invoke services, e.g., in a burst. Furthermore, there is no incentive for common service users to schedule or adapt the processing of requested workflows. In order to overcome these challenges and to introduce incentives for large-scale users, price differentiation according to the amount of service invocations can be used. Those volume-based tariffs with volume discounts usually have a non-linear relation between the amount of service invocations and the charged fee for service execution that can be of any continuous or discontinuous shape as depicted in Figure 10 [Sun03]. This allows the convergence of service prices to the willingness-to-pay of the consumer. An intermediary faced with this pricing model can minimize the average expense per service invocation, i.e., maximize the contribution margin, by operating in the system of volume-thresholds.

In contrast to the pricing models with static scheme, the characteristic of pricing models with dynamic scheme is the variation of the fee y induced by quantity x over time according to any other defined parameter v_i. Those external parameters v_i in the context of service pricing can be originated in technical conditions of the service provider's resources (e.g., bandwidth, capacity utilization), the user's willingness-to-pay (e.g., bids in an auction system, price formation through supply and demand), or any other market condition (e.g., prices of competitors). Thus, a further differentiation of variable pricing models with dynamic scheme is between *resource-oriented*, *user-oriented*, and *market-oriented* pricing models. Dynamic service pricing models can usually benefit both, the provider in terms of profit and the requesters in terms of customer surplus resulting in optimal welfare allocation [YGL06]. In contrast, those pricing models are very complex, difficult to implement and in contrast not easy to handle from the service provider's and the service consumer's points of view. Thus, due to the limited practicability and the lack of diffusion of dynamic pricing schemes, they will not be considered further in this thesis.

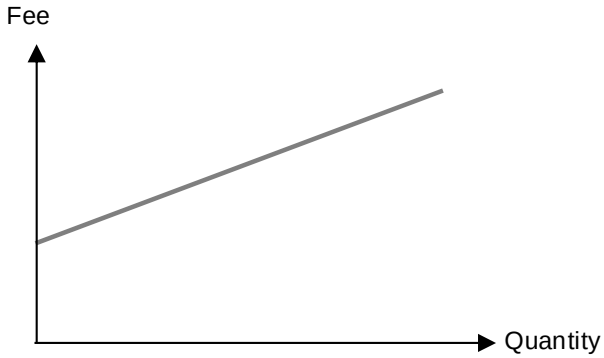


Figure 11: Hybrid pricing model

4.2.3 *Hybrid Fee Pricing Model*

A *hybrid fee pricing model* combines a fixed fee with a variable fee pricing model as presented in Figure 11 [Suno3]. Common methods for service pricing nowadays are mainly fixed fee (flat-rate) models and minor static pay-per-use models, mostly following a quantity- and/or priority-based price discrimination [HL03], [LZR03].

Offering quality or priority classes is another pricing mechanism, which increases flexibility for the service provider in terms of considering the price sensitivity of several consumers. These additional means can be used and applied in any pricing model mentioned in this section. QoS research on economics-based network resource allocation discusses the offering of several QoS and priority classes as means for using market mechanisms to suppress low value data traffic by differentiating prices according to priority classes. These approaches can be transformed into usage-based service pricing where each customer pays according to the quantity and quality of provided service [LZR03], [LLSW01], [GSW97], [LOSW02]. Hence, the service provider can offer one functionality, wrapped as a service, with several QoS properties at several cost levels.

In contrast to this approach, in which prices are set from one side, the provider side, other approaches exist such as auctioning models for services [YGL06]. The following section discusses the impact of different pricing models on the service selection and the resource planning process of service-based workflows.

4.3 IMPACT OF PRICING MODELS ON SERVICE SELECTION

The pricing models described in the previous section significantly affect the service invocation and resource planning process as discussed in the following.

As described earlier, by using a service that is charged by a fixed fee, the consumer has unlimited access to the service for a specific time or a specific request volume. The intermediary, in the role of a service consumer, purchasing a service with a fixed fee, has a high binding to the provider within this flat period. Switching the service providers can possibly turn the flat fee into sunk costs. This causes decreased flexibility and extended procurement risk by obligation to specific providers. Inversely to the disadvantages of inflexibility, this pricing approach benefits both in terms of transaction costs. The intermediary and the service provider only have low overhead for pricing and billing [Odl01]. From the perspective of the intermediary, the contribution margin per workflow execution increases with the number of service executions at one service provider, because the average unit costs per service invocation decreases with increasing service usage. Furthermore, by selling the service "workflow execution" to a third party with a fixed revenue per workflow execution, the intermediary can increase his contribution margin in this fixed fee time period by skimming the available resource volume.

Concerning variable pricing models with a static scheme, customer binding is avoided in this pricing model, because there is no obligation to the service provider, and the consumer can switch to other providers, if it is beneficial. This pricing scheme is independent of the amount of service executions and does not offer economies of scale towards customers. From a technical perspective, it suffers like any static policy from inflexibility, as it does not allow any reaction to varying conditions. Especially, large-scale customers are penalized in such a model, because it does not provide any incentives to adapt usage behavior, as usage is always accounted with a uniform linear factor. The intermediary has no incentive to schedule or adapt the processing of requested workflows, respectively the concrete task item in any way when confronting him with a linear pay-per-use pricing model. Regardless of any process step in the workflow, it is likely that the intermediary will process any workflow execution request immediately, as the contribution margin does not change over time. Some of the disadvantages are avoided by using pricing schemes with a non-linear relation as these schemes represent a form of price discrimination [Sun03]. Figure 10 shows an example of a pay-per-use model with volume-discounts with several volume thresholds. Concerning this pricing model, the unit costs per requested service decrease with increasing volume-interval, i.e., the more services are processed at this service provider. Thus, the intermediary can minimize the average expense per service invocation, i.e., maximize the contribution margin, by operating in the system of volume-thresholds. It will be advantageous for the intermediary to process the workflow execution requests in bursts in order to minimize the average unit costs per request instead of executing each workflow execution request immediately. A naive priority rule would be to queue requests in a first-in first-out approach as long as possible (regarding QoS) in order to accumulate enough quantity to aim a certain average volume-discount per accounted request.

Besides pricing schemes in which the price is determined a priori, dynamic pricing schemes do not provide this ability. Instead, the intermediary faces a situation in which he has to charge a uniform fee per workflow execution to his customers, whereas the invoked services induce non-uniform expenses, dynamically determined during run-time. Accordingly, the contribution margin will vary and inhabit financial risks that can be hardly forecasted and estimated. For this reason, invoking services with a dynamic pricing scheme is only beneficial when charging dynamically for the workflow execution as well.

The financial risk of the service provider is reduced by using the hybrid pricing model while it will increase the risks of a lock-in for the intermediary in contrast to the variable pricing model. Usually, this model is a combination of both, the fixed and the variable fee pricing model. This model has the advantage that the contractual binding has not such a long-term character as in case of a fixed fee but more than in case of a variable fee pricing model.

Summing up, pricing models significantly influence the behavior of an intermediary at the selection and composition process of services to service-based workflows. Based on these findings, the following chapters present several solution strategies for QoS optimization for service-based workflows by introducing several optimization approaches.

5 AVERAGE-CASE PERFORMANCE ANALYSIS

This chapter focuses on the average-case performance optimization of service-based workflows. With the help of queuing theory, it is possible to determine the average-case performance behavior of service-based workflows. Furthermore, an optimization approach is introduced, facilitating the maximization of service utilization of the invoked services considering a given cost model (volume-oriented pricing model) by providing a solution at low service invocation costs [EPR+07].

After a brief introduction of the basics of queuing theory, this chapter gives an overview on related work in this field of research. Furthermore, peculiarities of the considered scenario are described in the system model and the adaptation of queuing theory to service-based workflows, as well as the developed optimization approach are presented.

5.1 QUEUING THEORY BASICS

This section focuses on the introduction of queuing theory basics with special references to the textbooks of Haverkort [Hav98] and Bolch et al. [BGdMT06]. Queuing theory, as the oldest model for packet switched networks, is well studied in literature [LZGS84], [Baloo]. Basics of this theory can be found in the work of Kleinrock [Kle75], [Kle76]. Queuing is a phenomenon that can be realized and observed in reality in several situations such as in front of a supermarket check-out or at the airline check-in counter at the airport. Queuing occurs because arrival patterns of the customers differ from the service patterns. Furthermore, this phenomenon is very important in the areas of logistics, manufacturing lines, and communication systems.

Usually, a queuing system is described as shown in Figure 12. The arrivals to the system may be requests to a printer, a server, a service provider, or a workflow in general. For the execution of the incoming execution requests, appropriate scheduling mechanisms are necessary. The most prominent scheduling mechanisms are as follows [BGdMT06]:

- FCFS: First-Come-First-Serve; jobs are served in order of the arrival.
- LCFS(PR): Last-Come-First-Serve, with or without Preemption; the job that arrived last is served first.
- SIRO: Service-In-Random-Order; the job to be served next is selected randomly.
- RR: Round-Robin; if the servicing of a job is not completed at the end of a time slice of specified length, the job is preempted and returns to the

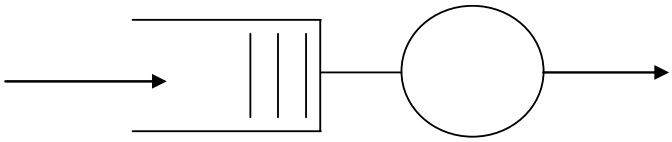


Figure 12: Basic queuing system

queue, which is served according to FCFS. This procedure is repeated until the servicing of the job is completed.

- PRIO: Prioritized Scheduling; jobs are served according to a specific prioritization scheme.

Kendall's notation, as a widespread notation, is often used when describing queuing stations. In detail, the notation of Kendall consists of six identifiers as shown below, whereas only the first four identifiers are important for the further considerations in this section, because the population is assumed to be infinite large and for the scheduling First-Come-First-Serve is assumed:

Arrivals | Services | Servers | Buffersize | Population | Scheduling

In detail, the arrival process to the servers is denoted as *Arrivals*, the characteristics of the servicing of the incoming execution requests, i.e., the servicing characteristics, is denoted as *Services*. Furthermore, the number of servers in the considered queuing system is denoted as *Servers*, and the maximum number of customers in the queuing system as *Buffersize*. The characteristics of the first two identifiers can be of several properties. In general, Exponential Distribution (memoryless), General Distribution, or Deterministic Distribution can be assumed. Markovian or Memoryless (M) is used when the interarrival or the service times are exponentially distributed. General (G) is used whenever the times involved are arbitrary distributed, and Deterministic (D) whenever the times involved are constant. Besides these prominent assumptions, also other distributions may be possible [Hav98]. The described optimization approach in this chapter is based on a system with (a series of) M|M|1 queuing stations as described in the following.

5.1.1 M|M|1 *Queuing Model*

The simplest queuing system is the M|M|1 queuing model assuming that the arrivals are distributed in a Poisson manner (as depicted in Equation 5.1) with the arrival rate λ [Hav98].

$$P(X = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad k \geqslant 0 \quad \forall \quad k \in N \tag{5.1}$$

The Poisson process has three major properties:

1. The probability of a specific number of arrivals in a time interval of length t is only dependent on the length of the time interval (stationary).

2. The amount of arrivals in not overlapping intervals are stochastically independent (homogeneity).

3. For small intervals of length Δt it holds that $X_{\Delta t} = 0$ or $X_{\Delta t} = 1$.

The average number of arrivals in a specific time period t is $E(k) = \lambda t$ and the variance ν is $\nu = \lambda t$.

The M|M|1 queuing model assumes, that the service times follow an exponential distribution at the server. In a Poisson process, the customer interarrival times are exponentially distributed. This memoryless characteristic implies that the next state x_{n+1} does not depend on the time the process has spent in state x_n. The context between a birth-death process, as a special case of a Markov process, and the M|M|1 queue is that a birth reflects the arrival of a new customer, either to the queue or directly to the server (depending on the status of the queue). A death represents that a customer has already been served and has left the system. Usually, the queue is modeled as a black box where λ represents the requests respectively jobs arriving per time unit on average. The number of incoming jobs in a specific time period λ is called the *arrival rate* or the *arrival intensity*. As a scheduling strategy First-Come-First-Serve is applied in the following [Buz73]. In case that the queue already contains some waiting requests, the incoming request has to wait in the queue until the server is empty and the job can be processed.

5.1.2 *Arrival Rate and Service Rate*

The arrival rate and the service rate are of high importance when analyzing a queuing station. The rate at which the incoming execution requests arrive at the server is denoted by λ with the exponentially distributed interarrival time $t = \frac{1}{\lambda}$. Considering several parallel incoming Poisson processes $\lambda_1, ..., \lambda_n$, the resulting process λ_r is a Poisson process which can be calculated as shown in Equation 5.2.

$$\lambda_r = \sum_{i=1}^{n} \lambda_i \tag{5.2}$$

At the server, the service times are exponentially distributed with the mean service time $x = \frac{1}{\mu}$, whereas μ denotes the rate at which the jobs are processed.

5.1.3 *Overload Avoidance and Utilization*

Overload avoidance is a major issue when ensuring that the server operates in stable state and avoiding the accumulation of an infinitely large waiting queue. In order to avoid overload, as a precondition, the average number of arrivals per time unit may not exceed the average number of jobs the server is able to

process per time unit, i.e., $\lambda < \mu$. The effective utilization ρ_i of an arbitrary server i can be calculated as shown in Equation 5.3.

$$\rho_i = \frac{\lambda_i}{\mu_i} \tag{5.3}$$

Following this precondition, the system (server) can only operate in stable state if $\rho_i < 1$. Otherwise more incoming jobs would arrive than the server is able to serve per time unit.

5.1.4 *Basic System Characteristics*

Two important characteristics of a queuing station are the average number of jobs N_i in the system and the average system time T_i. The average number of jobs in the system is the summation of the number of jobs that are waiting in the queue and the number of jobs that are in the server at a specific time. The average number of jobs N_i in the system yields to:

$$N_i = \frac{\rho_i}{1 - \rho_i} \tag{5.4}$$

The average system time T_i is the time a job spends in the system (queue and server) until the job is processed and has left the system. The calculation of T_i is shown in Equation 5.5.

$$T_i = \frac{\frac{1}{\mu_i}}{1 - \rho_i} \tag{5.5}$$

5.1.5 *Burke's Theorem*

The theorem of Burke is of high importance considering the departure process of a server in this scenario. The departure process γ_i of a single server M|M|1 queue in stable state with an arrival rate λ_i and a service rate μ_i respectively, is again a Poisson process with the rate λ_i [Hav98].

5.1.6 *Feed-forward Queuing Networks*

After focusing on the general behavior of M|M|1 queues, a special focus is on Feed-forward Queuing Networks (FFQNs), i.e., on a series of M|M|1 queues where the departure process at queue i represents the arrival process at queue $i+1$, assuming that there is no cross-traffic at intermediary queues. Considering the first queue, the external arrival rate to the system is λ and the service rate of queue i equals μ_i. In order to ensure stability of the queues, the following condition has to hold:

$$\rho_i = \frac{\lambda}{\mu_i} < 1 \quad \forall \quad i = 1, ..., n \tag{5.6}$$

In the case of $\rho_i \geqslant 1$, queue i would build up an infinitely large waiting line and the average response time T would increase ad infinitum. The bottleneck in such a system of a series of M|M|1 queues is the queue with the smallest ρ_i. Assuming there are no arrivals to the queuing network in between any two queues and no departures from the queuing network, applying *Burke's Theorem* the arrival rate at any M|M|1 queue i is λ.

After introducing queuing theory basics, the following section provides an overview on related work.

5.2 RELATED WORK

Queuing theory is widely used in the field of traffic management, manufacturing systems, and production lines. Concerning manufacturing lines, the service stations correspond to servers and the buffers at the stations to the queuing capacity [PHB93]. Furthermore, production processes comprising assembly lines with finite buffers as well as simple processing can be analyzed with the help of queuing theory [Mano8]. However, concerning the adaptation of queuing theory to services or workflows, current work is limited. Although, a set of related work treating research in this field is depicted in the following.

Peng et al. discuss capacity planning for composite services using queuing network-based models and map findings of queuing theory to services [PYY+04]. For several possible topologies, the requests are passed through the system as iterations of services or sequences of services. Queuing network-based models are used in order to determine the performance of composite services. With the help of this approach it is not necessary to run extensive simulations, instead with the help of an analytical analysis the performance can be determined.

Litoiu proposes a reengineering approach for the migration of legacy applications towards a system based on services [Lito4]. The aim of this approach is to present how performance models can help to support migration decisions and to introduce a mechanism of building and solving performance models, which are derived from a system based on services. Impacts of the migration process to several performance metrics, such as response time, scalability, and throughput are considered in his work. The introduced approach uses layered queuing networks as performance models in order to predict the performance of this system.

Patel and Darlington state that services become more and more important in order to facilitate service-oriented distributed computing. In a Grid environment the provided workflows have to fulfill a certain QoS level [PDo6c]. The authors address workload allocation techniques for Grid workflows representing compositions of services from the Grid. In this consideration the services are modeled as M|M|1 queues, i.e., the arrivals are modeled as Poisson processes. The derived numerical solution for missed deadlines (failures) of Grid

workflow tasks is evaluated through an experimental simulation. In a further consideration the authors model the considered services as G|G|1 queues and minimize failures (QoS requirement violation) of jobs by solving a mixed integer non-linear programming problem [PD06a], [PD06b].

The described approaches mainly focus on the adaptation of major findings of queuing theory to entire systems or focus on some peculiarities of workflows in general. However, none of these approaches focus on the service selection problem of service-based workflows and on the addressed optimization approach as depicted in this chapter.

5.3 SYSTEM MODEL

As mentioned in the scenario (Chapter 3), the considered business processes can be decomposed into several basic activities that can be executed by a specific service S_i $(i = 1, ..., n)$ which fulfills the required functionality of the considered task. For each task i services m_i are available offering the required specific functionality; they solely vary in their QoS characteristics. As mentioned earlier, the selection and composition of services can be done from internal services as well as from external partners in order to create service-based workflows [ZBN+04].

Queuing theory allows the analysis of complex networks (and therefore complex workflows) containing, e.g., loops, iterations, forking, and joining paths [Hav98]. In the further analysis, workflows with a sequential execution of the tasks as depicted in Figure 6 on page 27 are considered. These tasks, dependent on the granularity of the task decomposition, can be decomposed in a way that there exist ranges of services $S_{i,j}$ which fulfill the required functionality of the considered task i. The intermediary is responsible for the execution of a workflow. He has to handle all execution requests and has to ensure that all requests can be served within a specific time period as it is his objective to serve all incoming workflow execution requests at minimal costs and at minimal response time.

Assuming a huge amount of incoming execution requests, the assumption that the arrivals can be modeled as a Poisson process can be assumed in various scenarios [GWW02]. The considered scenario assumes that the incoming workflow execution requests can be modeled as a Poisson process as described in Section 5.1. For each task, the intermediary has to choose the appropriate service, which fulfills the required functionality described in the SLAs. The considered workflow consists of n different activities. For each task i the intermediary has to choose a service which fulfills the required functionality out of $j = 1, ..., m_i$ available services per task i. Furthermore, the intermediary has to ensure that task $i (i = 1, ..., n)$ is executed before task $i' (i' = 1, ..., n)$ if $i < i'$. Each service has a specific execution capacity, i.e., a specific amount of requests that can be served within a specific time period. The service invocation is charged via a volume-oriented pricing model (see Section 4.2.1). The

services differ solely in the provided service rates $\mu_{i,j}$ and the costs $c_{i,j}$. For the correlation between service rate and the costs, we assume that the higher the service rate $\mu_{i,j}$ the higher are the costs $c_{i,j}$ for each service.

As it is the objective to use the execution capacities of each service optimally, in order to increase the throughput by an optimal utilization of the invoked services, the intermediary has to create an invocation plan meeting those demands. Furthermore, he has to maximize the throughput, and to realize an optimal resource usage in order to use the execution capacities of the services optimally.

5.4 ADAPTATION OF QUEUING THEORY TO SERVICE-BASED WORKFLOWS

The considered workflow, as mentioned in Section 5.3, consists of several tasks that have to be implemented by services. Furthermore, the number of incoming workflow execution requests in a specific time period can be modeled as a Poisson process with a specific arrival rate λ. The intermediary is faced with this arrival rate and has to select the available services out of a range of services in a way that all incoming execution requests can be served. For service selection of the first task, the intermediary can choose from a subset m_1 of services $S_{1,j}$ fulfilling the required functionality. After invoking a service for the first task, the intermediary has to invoke a service for the second task out of a range of m_2 services fulfilling the required functionality of task 2, denoted by $S_{2,j}$. The specific aspect of this consideration is, that due to a large amount of incoming execution requests and varying service times of the services a queue of execution requests may occur at the selected services.

Adapting the findings of Section 5.1 to a service, being faced with a specific arrival rate λ and a service rate μ, the throughput respectively output γ in the long run will be the same as the input rate λ, i.e., $\gamma = \lambda$, if it holds that $\lambda < \mu$, i.e., $\rho < 1$.

The accumulated queue at this service occurs due to random arrivals and the fact that requests can occur in a burst. The intermediary stores the requests in the queue and forwards them to the subsequent service as soon as the preceding request is processed. The higher the utilization ρ of the service, the better the execution capacity of the considered service is used. The calculation of the average number of pending execution requests at the service and the average system time, representing the average response time, can be calculated with the help of Equation 5.4 and 5.5.

As described in Section 4.2, several pricing models for services exist. The following considerations assume a volume-oriented pricing model, as a special case of the fixed fee pricing model. Typically, each offered service provides a specific service execution rate $\mu_{i,j}$, representing the amount of requests the service is able to process within a specific time period. These services are offered by a service provider which charges the intermediary a specific amount $c_{i,j}$ for the usage of the service $S_{i,j}$. These incidental costs are usually independent of

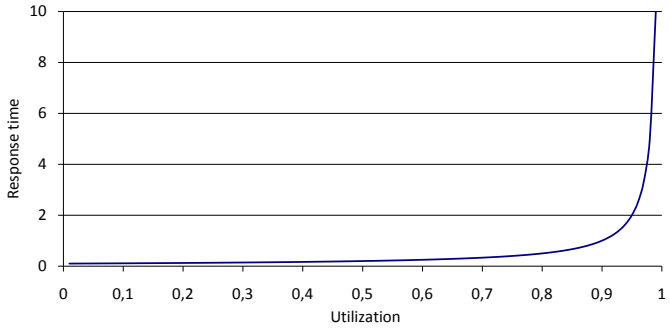


Figure 13: Utilization vs. response time

the real usage of the service. It is not necessary that the service requesters request service execution up to $|\mu_{i,j}|$, but it will be preferable in terms of operating cost-efficient. The objective of the intermediary is an efficient usage of the offered execution capacities $\mu_{i,j}$.

5.4.1 *Utilization Analysis of Services*

As it is the aim of the intermediary to optimally use the execution capacity of each single service S_i, the utilization ρ_i has to be as high as possible, i.e., close to 1. As presented in Equation 5.4, the average number of customers in the system N increases by an increasing value of the utilization ρ. Thus, the average response time of a service execution increases as well.

As presented in Figure 13, there is no optimal *operating point* for the specific service in terms of an optimal service usage. Instead, after a specific utilization of a service, the response time strongly increases. Thus, the intermediary has to consider this growth in average response time in order to avoid a high overall response time of the entire workflow.

5.4.2 *Throughput Analysis of Service-based Workflows*

Both, the throughput of the entire workflow as well as the overall response time are very important QoS parameters for workflow execution. Throughput in this scenario is the maximum number of execution requests being executed by the specific selection and invocation of services. The considered workflow, as described in Section 5.3, in which the tasks are arranged in a sequential order, can be described and analyzed with the help of FFQNs (see Section 5.1). Consequently, each service can be modeled as a M|M|1 queue in which the outgoing flow of queue i equals the incoming flow of queue $i+1$. Furthermore, the presentation of a composed workflow behaves as an acyclic queuing network and each invoked service, representing a M|M|1 queue, operates in a stable state ($\lambda < \mu$). This association, i.e., the relation between a service-based workflow and a FFQN is shown in Figure 14.

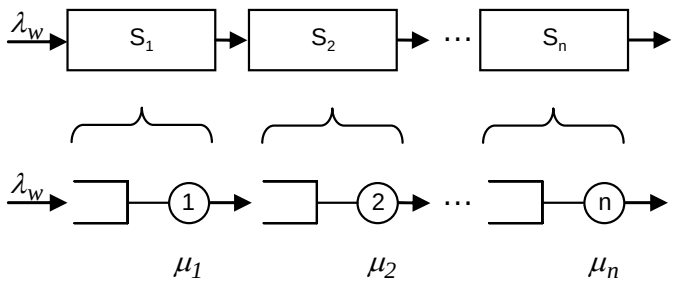


Figure 14: FFQN model for service-based workflows

Avoiding overload of the services and ensuring that the services operate in stable state implies that the utilization ρ_i for each invoked service S_i has to be smaller than 1, i.e., $\rho_i = \frac{\lambda}{\mu_i} < 1$. The incoming workflow execution requests, representing an aggregation of several execution requests from several service requesters, have an aggregated arrival rate λ_w, i.e., the arrival rate λ_1 at S_1 equals $\lambda_1 = \lambda_w$. According to Burke's theorem, presented in Section 5.1, the output γ_1 at S_1 equals the input rate λ_w, if $\rho_1 < 1$. The invoked service S_2 of task 2 has the input rate $\lambda_2 = \gamma_1 = \lambda_1 = \lambda_w$ if $\rho_2 < 1$ on average.

This consideration holds for each invoked service of the workflow. Thus, the overall throughput γ_w of the entire workflow is the same as the input rate, i.e., $\gamma_w = \lambda_w$ if for all invoked services S_i $i(i = 1, ..., n)$ holds $\rho_i < 1$ and the workflow is a closed system, i.e., the invoked services are not used by any other workflow and there is no cross-traffic. In the case of $\rho_i > 1$, an infinitely large waiting queue of service execution requests and an infinitely large response time would be accumulated. In case that the selected service is not able to hold the stability condition, this service should be replaced by another service with a higher service rate or a service should be invoked in parallel in order to serve all incoming execution requests.

5.5 OPTIMIZATION APPROACH

After the description of the general implications of the adaptation of queuing theory to service-based workflows, this section presents an optimization approach for an efficient usage of invoked services. This ensures an efficient utilization of the execution capacities of all invoked services of the workflow. Thus, some constraints such as response time and number of workflow execution requests currently pending are considered as well.

$$\sum_{j=1}^{m_i} x_{i,j} = 1 \quad \forall \quad i = 1, ..., n \tag{5.7}$$

For all $n * m_i$ services a binary variable $x_{i,j}$ is introduced to model whether a service $S_{i,j}$ is used for the workflow execution or not. In this consideration, sufficient execution capacities of the available services in order to execute all incoming execution requests are assumed, i.e., it is not necessary to invoke

some services in parallel. Equation 5.7 avoids that more than one service of one task is used at the same time.

5.5.1 *Objective Function*

The objective of this optimization approach is the maximization of the service utilization for an efficient usage of the service execution capacities. With the help of this approach, the intermediary can create a service invocation plan, which maximizes the overall utilization. Thus, the intermediary has to calculate all potential utilizations $\rho_{i,j}$ of all services $S_{i,j}$ given the aggregated arrival rate of all incoming workflow execution requests of the requesters λ_w and the individual service rates $\mu_{i,j}$. Equation 5.8 presents the objective function for this approach, ensuring that the average service utilization is maximized and that the invoked services are able to serve all incoming execution requests. With these utilization parameters, the intermediary is able to compute the maximum average utilization.

$$Max\ F(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \rho_{i,j} x_{i,j} \tag{5.8}$$

As a recapitulation, Figure 13 highlights that the higher the utilization of a service the higher is the average response time. Thus, in order to avoid an increase in overall response time, some constraints have to be introduced.

5.5.2 *Constraints*

The overall average response time T_w of the considered workflow can be calculated by the summation of all average response times $T_{i,j}$ of all invoked services. As the maximization of service utilization is the aim of the optimization approach, the invoked services will have a high utilization $\rho_{i,j}$ and therefore a high response time for the execution requests. In order to avoid an infinite increase of the overall average response time, the overall average response time of the workflow has to be constrained with a maximum average response time T_{max} as shown in Equation 5.9.

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} T_{i,j} x_{i,j} \leqslant T_{max} \tag{5.9}$$

Besides an increase in the average response time of the workflow execution, the average outstanding workflow execution requests increase with a higher service utilization. The pending workflow execution requests N_w can be calculated by the summation of all average outstanding requests at each service as shown in Equation 5.10.

$$N_w = \sum_{i=1}^{n} \sum_{j=1}^{m_i} N_{i,j} x_{i,j} \tag{5.10}$$

A further constraint, avoiding a large increase in workflow execution requests currently pending N_w is described in Equation 5.11 with the help of an appropriate boundary N_{max}.

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} N_{i,j} x_{i,j} \leqslant N_{max} \tag{5.11}$$

Considering the volume-oriented pricing model, the intermediary has to pay a fixed amount $c_{i,j}$ for the usage of service $S_{i,j}$ with the service rate $\mu_{i,j}$ independent of the real service usage. The overall service invocation costs C_w of the entire workflow execution can be calculated by the summation of all costs for each invoked service $S_{i,j}$ as presented in Equation 5.12.

$$C_w = \sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{i,j} x_{i,j} \tag{5.12}$$

As it is the objective of the intermediary to maximize the service utilization, he will invoke services with small service rates that still exceed the arrival rate in order to avoid overload. Furthermore, because of the positive correlation between service rate $\mu_{i,j}$ and costs $c_{i,j}$, the developed approach creates a solution with low overall service invocation costs.

$$\rho_{i,j} < 1 \quad \forall \quad i = 1, ..., n \tag{5.13}$$

In order to avoid an infinite accumulation of workflow execution requests due to overload, the utilization of each invoked service has to be smaller than 1 as shown in Equation 5.13.

5.6 SUMMARY

This chapter describes the adaptation of major findings of queuing theory to service-based workflows, facilitating the identification of the overall average-case performance behavior of the workflow. This performance behavior can be determined by modeling each service as a M|M|1 queue and by applying Burke's theorem and the concept of FFQNs to this scenario. In particular, it is possible to determine the service utilization as well as the service invocation costs, the average response time of workflow execution requests, and the average number of outstanding workflow execution requests currently pending.

Furthermore, the developed optimization approach facilitates the optimization of the service utilization by determining the optimal selection of a set of services to a service-based workflow. The identified approach offers a solution with low overall service invocation costs, because maximization of service utilization implies minimization of service rates and therefore minimization of service invocation costs. In addition, the introduced constraints ensure that the overall average response time of workflow execution requests and the overall number of execution requests currently pending do not exceed a specific value.

6 WORST-CASE PERFORMANCE ANALYSIS

After the development of an average-case performance optimization approach, supporting the intermediary in the service selection process, this chapter focuses on a solution for the worst-case performance optimization. Major findings of network calculus are adapted to the concept of service-based workflows, facilitating the analysis of the worst-case performance behavior of service-based workflows [EPR+07]. The basic performance bounds network calculus is dealing with, are the delay, the backlog bound, and the throughput. Furthermore, this section describes the developed performance optimization approach facilitating the optimization of the worst-case performance behavior of service-based workflows [ESN+08].

After introducing basics of network calculus, an overview on the considered system model is given. Furthermore, the adaptation of network calculus to service-based workflows and the developed optimization approaches are presented.

6.1 NETWORK CALCULUS BASICS

A concept for modeling and evaluating QoS has gained importance in judging and improving network performance in the last decade. This concept, referred to as network calculus represents a theoretical and systematical approach that tries to apply methods known from classical system theory to computer networks. The originator of this approach is Cruz [Cru91a], [Cru91b].

Network calculus, as a system theory for deterministic queuing systems, has been developed in the 1990s and is widely used in the context of deterministic QoS in packet switched networks. The system theory approach generally models a phenomenon by three entities: a system, an input, and an output whereas the system maps an input to an output. In the field of electrical engineering, especially in communication networks and control systems, this approach is widely spread. Network calculus brings the system theory approach to computer networks. In this application scenario, the input to the system is represented by the traffic flow that is originated at the sender, the system model is represented by the network, which introduces, e.g., delay and packet loss. The output is the traffic flow that arrives at the receiver. The performance of a network can be analyzed with this model by evaluating whether the traffic flow arriving at the receiver is appropriate. In contrast to classical system theory, as a particularity, network calculus is based on min-plus algebra as a mathematical foundation. The strong analogy of network calculus and system theory is further elaborated in the work of Fidler and Recker [FR05a], [FR05b], [FR06].

The concept of network calculus can be described as follows [BT01]:
"Network calculus is a theory of deterministic queuing systems found in computer networks. It can be described as the system theory that applies to computer networks. The main difference with traditional system theory, as the one that was successfully applied to design electronic circuits, is that another algebra is considered, where the operations are changed as follows: addition becomes computation of the minimum, multiplication becomes addition."

In min-plus algebra the addition operator changes to the *minimum* operator and the multiplication operator changes to the *plus* operator. As a reason for this, it can be stated that *plus-times* algebra is well suited for describing physical systems, *min-plus* algebra is suited for describing man-made systems [BCOQ92]. In particular, a man-made system is usually characterized by several customers sharing a resource while having a goal. Considering computer networks, the customers are the data packets, the shared resource is represented by the network, and the goal is, e.g., to maximize the throughput or to minimize the delay.

After the description of the main concept of network calculus, the following sections present essential definitions and findings that are important for the further analysis. The according proofs can be found in [BT01]. At first, an input and an output function of a system is defined in order to describe the concept of an arrival curve and a service curve.

6.1.1 *Input and Output Functions*

Definition 1 [Input function]:
$R(t)$ is called an *input function*, if $R(0) = 0$ and R is wide-sense increasing, that is for all $t_1 \leqslant t_2$ holds $R(t_1) \leqslant R(t_2)$. $R(t)$ denotes the number of bits arriving in the interval $[0, t]$. The time t and $R(t)$ can either be discrete or continuous. If both t and $R(t)$ are continuous, we say we have a *fluid model*.

Definition 2 [Output function]:
$R^o(t)$ is called an *output function* for a system S, if it cumulates the output (in bits) of S in the interval $[0, t]$ for all $t \geqslant 0$. In this definition S might be a single buffer, a network node or a complete network. As above, t and $R^o(t)$ may be discrete or continuous. For a system S it should (obviously) hold $R^o(t) \leqslant R(t)$. This assumption implies having a lossless system without routing loops.

6.1.2 *Arrival and Service Curve*

Besides the introduction of an input and an output function, it is necessary to define conditions for service guarantees, i.e., on the one hand the arrival curve which constrains the arrival process and on the other hand the service curve, which describes the worst-case service behavior when packets are served.

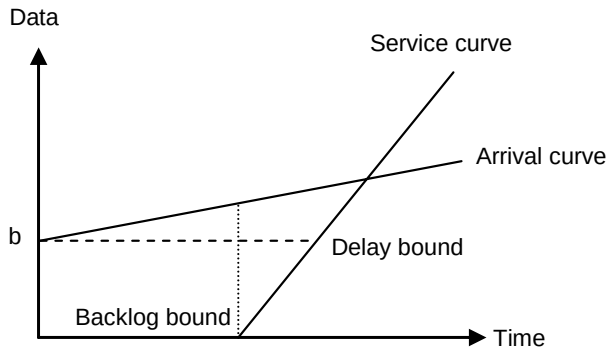


Figure 15: Arrival curve, service curve, backlog bound, and delay bound

The input to the system is represented by the arrivals, corresponding to the upper bound for the input traffic, unlike in classical system theory, where the input is the actual signal that enters into the system. The arrival curve denotes the maximum traffic that the sender may introduce into the network. Usually, it is given as a function, in which the x-axis denotes the time interval and the y-axis denotes the maximum amount of data that may be sent in the corresponding interval. An example of an arrival curve, along with the other concepts presented in this section is shown in Figure 15.

Definition 3 [Arrival curve]:
Given a wide-sense increasing function α defined for $t \geqslant 0$, we say that a flow R is constrained by α if and only if for all $s \leqslant t$ holds:

$$R(t) - R(s) \leqslant \alpha(t - s) \tag{6.1}$$

In this consideration R has α as an arrival curve, or rather R is α-smooth. In general, several arrival curves can be assumed for traffic regulation purposes. Leaky Bucket and Token Bucket are the most prominent examples for traffic regulation algorithms [Tur86]. The Token Bucket arrival curve is given by Equation 6.2 where at $t = 0$, the amount of b data packets are at the system and for $t > 0$ the packets arrive with a constant rate r_a. With this arrival curve the maximum size of bulk arrivals can be modeled as well as the sustained rate for incoming execution requests.

$$\alpha(t) = b + r_a t \quad \text{for} \quad t > 0 \tag{6.2}$$

In contrast to the arrival curve, as an upper bound for the arrival process to a system, the service curve represents a lower bound for the service rate. It indicates the amount of data that a node serves in the worst-case. The service curve is usually given as a function, where the x-axis denotes a time interval and the y-axis the minimum amount of data that must be served of the node. The Latency-Rate (LR) service curve is a widespread service curve that is often used when modeling service behavior [Pano6]. The LR service curve has two parameters: After a specific latency l, the packets are served at a constant rate r.

Definition 4 [Service curve]:
Consider a system S and a flow through S with input function R and output function R^o. We say S offers a *service curve* β to the flow, if and only if β is wide–sense increasing and $R^o \geqslant R \otimes \beta$ for all $t \geqslant 0$.

In this definition the operator $\otimes$ represents the min-plus convolution which is specified in more detail in the work of Boudec and Thiran [BT01]. An analytical description of the properties of a LR service curve is presented in Equation 6.3.

$$\beta(t) = r(t-l) \quad ; \beta(t) = 0 \quad t < l \tag{6.3}$$

The rate at which requests are served is denoted by r and the time needed for initialization (latency) is defined by l.

6.1.3 *Backlog Bound and Delay Bound*

The following introduces two basic bounds of network calculus: the backlog bound and the delay bound. The backlog bound denotes the amount of data that is currently in the network. As presented in Figure 15, the backlog bound is the maximum vertical distance of the arrival curve and the service curve. The delay bound denotes the maximum delay of a packet in the network. It is given by the maximum horizontal distance of the arrival curve and the service curve.

Definition 5 [Backlog bound]:
Assume a flow, constrained by an arrival curve α, traverses a system that offers a service curve β. The backlog $R(t) - R^o(t)$ for all t satisfies:

$$R(t) - R^o(t) \leqslant \sup_{t \geqslant 0}\{\alpha(t) - \beta(t)\} \tag{6.4}$$

Definition 6 [Delay bound]:
Assume a flow rate R(t) constrained by arrival curve α traverses a system S that offers a service curve of β. The virtual delay d(t) for all t satisfies:

$$d(t) \leqslant \sup_{t \geqslant 0}\{\inf_{\tau \geqslant 0}\{\alpha(t) \leqslant \beta(t+\tau)\}\} \tag{6.5}$$

6.1.4 *Concatenation*

The concatenation theorem allows the analysis of networks consisting of multiple nodes. Analog to the classical system theory, the transfer function of a system is given by the convolution of the transfer functions of the subsystems. Considering a sequence of nodes where packets are routed via a distinct path

in the network, the service curve of this path is given by the min-plus convolution of the service curves of the nodes along the path. The min-plus convolution is given by Equation 6.6, where $\beta_1(t)$ and $\beta_2(t)$ are the service curves of two arbitrary nodes 1 and 2 and $\beta(t)$ is the resulting service curve of the path consisting of those two nodes. This concatenation is arbitrary extensible and allows the determination of the aggregated service curve of multiple nodes in a network.

$$\beta(t) = \min_{0<\tau\leqslant t}\{\beta_1(\tau) + \beta_2(t-\tau)\} \tag{6.6}$$

After the introduction of network calculus basics, the following section focuses on the description of the analyzed system model.

6.2 SYSTEM MODEL

As presented in Chapter 3, we assume that the workflow execution requests may arrive in different forms. Possibilities are bulk arrivals, constant rate arrivals, or stochastic arrivals. As this chapter focuses on the deterministic worst-case behavior, purely stochastic arrival models are neglected, as they might cause the worst-case performance to be arbitrary bad, even if the probability for that case is low. Considering the arrival of workflow execution requests, we assume that the Token Bucket arrival curve is appropriate in order to capture the incoming request arrivals. This arrival curve allows to realize a maximum size of bulk of arrivals at $t = 0$ as well as the sustained arrival rate.

Concerning services, realizing specific tasks in a service-based workflow, the response time and the rate at which requests can be served are two important parameters. This can be realized by the well-suited concept of service curves in order to describe the performance of a service. Especially the LR service curve grasps exactly the two aforementioned parameters [Pano6]. The service invocation usually implies a dedicated latency before execution requests can be processed. The latency performance of SOAP implementations is depicted in more detail in the work of Davis and Parashar [DP02].

Beyond that, it is possible to have service providers, offering services in the form of "the first 50 request are serviced with a rate of 20 executions/sec. and the remaining ones with a rate of 105 executions/sec." This is grasped by the L2R service curve [Scho2], which will be neglected in the further analysis due to the fact that this service behavior is not as common as in case of the LR service curve. Hence, the service provider may use sophisticated service curves in order to do cost differentiation and offer services at several QoS levels. When determining the appropriate parameters of the service curve, the effect of other workflows being executed at the server, possibly leading to congestion, must be incorporated as well. Again, subsets of services exist offering a specific functionality (see Section 5.3). We assume that the services differ solely in the provided service rates $r_{i,j}$, the latency $l_{i,j}$, and the costs $c_{i,j}$. For the correlation between service rate and costs, we assume that the higher the service rate $r_{i,j}$, the higher are the costs $c_{i,j}$ for each service.

Packet Switched Networks	Service-based Workflows
Path through the network	Workflow
Node in the network	Service
Packet	Request
Throughput	Rate at which requests can be processed
End-to-end delay	Time until a workflow execution is completed

Table 2: Analogies between packet switched networks and service-based workflows

6.3 ADAPTATION OF NETWORK CALCULUS TO SERVICE-BASED WORKFLOWS

The considered workflow consists of a set of tasks in a sequential order that can be implemented by services. Hence, if the characteristics of services in general can be described by service curves, the entire workflow can be described as the concatenation of these service curves. This connection is an analogy to a network path, where the service curve of the path consists of the concatenation of the service curves of the nodes. These analogies between packet switched networks and service-based workflows are summarized in Table 2. For a method how to intuitively convolve arbitrary functions, the reader is referred to Pandit et al. [PSKS06] (elaborated in [Pan06]).

Besides the analogies described above, also the other theorems of bounds (backlog bound and delay bound), described in network calculus, can be adapted to the concept of service-based workflows:

- The delay bound indicates the worst-case response time that a workflow execution request may have. It occurs when the largest possible bulk of workflow execution requests arrives and the service provider offers exactly one specific service curve.
- From the backlog bound it can be deduced how many workflow execution requests are currently pending, which equals the maximum amount of workflow execution requests not yet executed.

6.3.1 *Optimal Choice of Service Providers*

In the considered scenario, the intermediary has to identify an optimal service invocation plan regarding the offered services of the service provider. In order to identify the best-suited services realizing the execution of the workflow, the intermediary has to determine the overall service curves for the workflow for each possible composition of services. Out of these, the intermediary has to select an appropriate aggregated service curve, which maximizes the overall throughput and minimizes the worst-case delay of the entire workflow.

An example of the adaptation of service curves to service-based workflows is shown in Figure 16. The arrival curve $\alpha(t)$ describes the aggregated behavior of incoming workflow execution requests. The service curve $\beta_{i,j}(t)$ describes

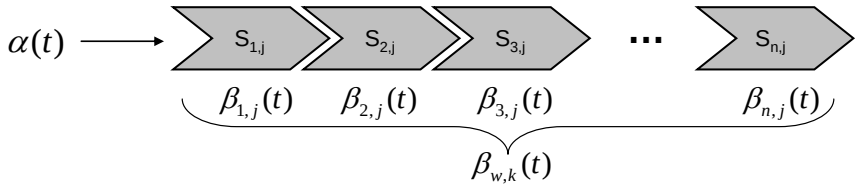


Figure 16: Adaptation of service curves to service-based workflows

the worst-case processing of each service $S_{i,j}$. By the convolution of all service curves of the invoked services the overall worst-case service curve of the workflow yields to $\beta_{w,k}(t)$. The parameter w indicates an aggregated service curve and k denotes the executable invocation plan (with $k = 1, ..., o$). For a workflow consisting of n different basic tasks and m_i different available services per task (service category), $o = m_i^n$ different service compositions are theoretically possible. Summing up, it can be stated that the arrival curve $\alpha(t)$ is not the real input of the system and $\beta_{w,k}(t)$ does not describe the real scheduling, instead they represent bounds for the arrival and service characteristics.

With the concepts and results presented in Section 6.1, the intermediary is able to determine the optimal combination of service providers and services respectively with respect to a certain goal. Minimizing the worst-case response time of a first request in a bulk is achieved by choosing for each task the service with the service curve having the lowest latency and the highest execution rate. The throughput can be maximized by determining for each service the service curve with the highest rate. Furthermore, the rates of each invoked service have to be compared and the lowest of them is the highest achievable rate for workflow execution, i.e., the highest achievable throughput. Since it does not help if other service providers offer higher rates than the highest achievable rate, out of these, service curves offering the highest achievable rate can be chosen. The optimization to a joint rate and a specific delay requirement is done by selecting from each service provider the service curve with the lowest delay under the condition that the rate requirement is met.

6.3.2 *Exemplarily Considerations*

In an exemplary consideration a generic credit process as presented in Figure 17 is assumed, which can be decomposed into the subprocesses loan request, credit assessment, servicing, and workout. There will be a large amount of workflow consumer requesting workflow execution, i.e., the appropriate service-based workflow has to be executed many times in a specific time period in order to serve all incoming workflow execution requests. We assume that the incoming workflow execution requests arrive at the intermediary and can be constrained by a Token Bucket arrival curve $\alpha(t)$ with the rate r_a and a depth b. The amount of requests to each offered service $S_{i,j}$ within a specific time period – specified respectively negotiated with the service provider in the SLAs – can be modeled as a service curve $\beta_{i,j}(t)$ in a worst-case consideration.

Figure 17: Generic credit process

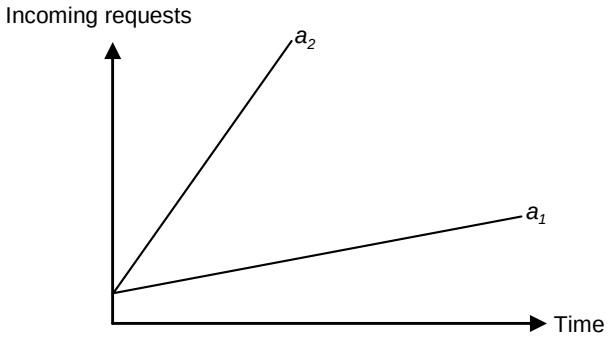


Figure 18: Arrival curves for incoming workflow execution requests

For modeling the incoming workflow execution requests to the credit process, two different Token Bucket arrival curves, as depicted in Figure 18, are assumed. For the execution of each task, a set of services, with varying QoS levels, offering a specific functionality are available. In order to reduce the complexity of this consideration and present the impacts of this adaptation on service-based workflows, we assume that three services are available for the first task and two services for each of the remaining tasks. In the following, the services for the first task are denoted as $S_{1,1}$, $S_{1,2}$, $S_{1,3}$ where the first index denotes the process step and the second index the number of the considered service (in category $i = 1$), which fulfills the required functionality of this process step. The services for the second task are denoted as $S_{2,1}$, $S_{2,2}$. The same holds for task three and task four.

After some time for connection establishment and initialization (latency $l_{i,j}$) the service starts request execution with a specific rate $r_{i,j}$. The differences between the offered services are on the one hand the time needed for initialization, i.e., the latency $l_{i,j}$, and on the other hand the execution rate $r_{i,j}$. Figure 19 shows this behavior with a LR service curve for the first task.

In this example the execution of $S_{1,1}$ starts with a small latency $l_{1,1}$ and with a slow execution rate $r_{1,1}$. $S_{1,2}$ has the same latency as $S_{1,1}$ but a higher execution rate $r_{1,2}$. The highest execution rate $r_{1,3}$ with the largest latency of $l_{1,3}$ has $S_{1,3}$. Figure 20 describes the service curves for the offered services for the remaining tasks. It is the aim for the intermediary to invoke those services, which, as a composition (applying the concatenation theorem), achieve the lowest worst-case delay and the highest worst-case throughput respectively.

Assuming the intermediary selects the services $S_{1,1}$, $S_{2,1}$, $S_{3,1}$, and $S_{4,1}$ for workflow execution, the overall service curve results to $\beta_{w,1}(t)$ with the delay d_1 as shown in Figure 21. The maximum delay that a request will experience in the worst-case to complete execution is denoted by the dashed line. This delay d_1 sets in for the last request, arriving in the largest possible bulk

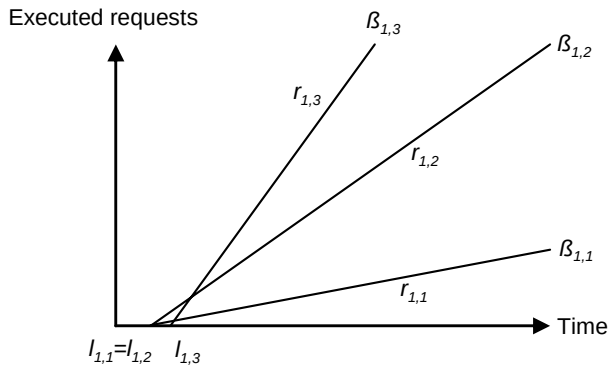


Figure 19: Service curves for the first task

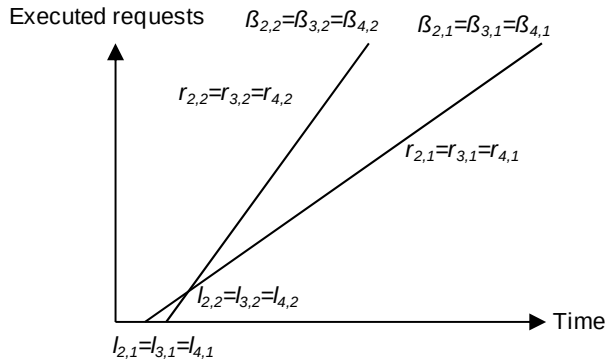


Figure 20: Service curves for the remaining tasks

allowed by the arrival curve, and the service provider needs its full latency before starting to serve requests. Furthermore, the achieved throughput can be increased by choosing the service curve $\beta_{1,2}(t)$, but it is not required in this case, as the rate of the arrivals can be served this way as well. However, the delay could be reduced when using $\beta_{1,2}(t)$. In these deliberations, it becomes clear that optimizing the choice of services is a non-trivial problem that will be discussed in Section 6.4.

For throughput maximization purposes, the services with the highest execution rates have to be chosen by the intermediary. This would comply with $S_{1,3}$, $S_{2,2}$, $S_{3,2}$, and $S_{4,2}$. The overall service curve results to $\beta_{w,2}(t)$ which is shown in Figure 22. In this case, it becomes clear, that even though the rate is higher, the latency prior to the first request processed is higher than in the previous example.

After the description of the basic implications for the adaptation of network calculus to service-based workflows, the next section presents optimization approaches, facilitating the optimization of several workflow characteristics.

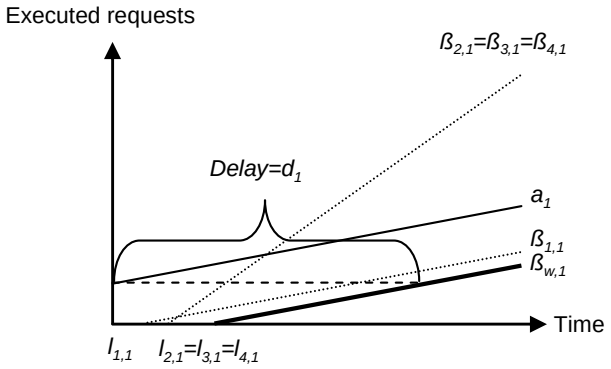


Figure 21: Delay for an aggregated service curve

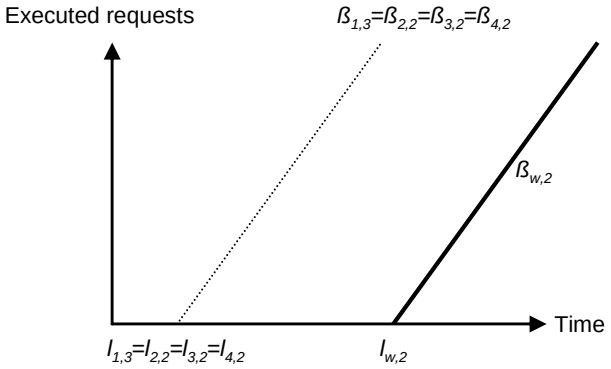


Figure 22: Throughput maximization

6.4 PERFORMANCE OPTIMIZATION APPROACHES

This section presents optimization approaches facilitating the minimization of the worst-case delay and the maximization of the throughput of the considered service-based workflow.

As described in the system model (Section 6.2), several services $S_{i,j}$ for each category exist that solely vary in their QoS levels, i.e., the differences of these services are the latency $l_{i,j}$, the rate $r_{i,j}$, and the costs $c_{i,j}$. An analytical description for these services is presented by Equation 6.7.

$$\beta_{i,j}(t) = r_{i,j}(t - l_{i,j}) \quad ; t \geqslant l_{i,j} \tag{6.7}$$

In addition, the aggregated service curve $\beta_{w,k}(t)$, representing the convolution of the service curves of the invoked services, can analytically be described as shown in Equation 6.8.

$$\beta_{w,k}(t) = r_{w,k}(t - l_{w,k}) \quad ; t \geqslant l_{w,k} \tag{6.8}$$

with

$$r_{w,k} = \min\left\{r_{1,j}, r_{2,j}, ..., r_{n,j}\right\} \tag{6.9}$$

$$l_{w,k} = \sum_{i=1}^{n} l_{i,j} \tag{6.10}$$

The resulting execution rate of the aggregated service curve $r_{w,k}$ is the minimum of all execution rates $r_{i,j}$ (with $i = 1, ..., n$) of the chosen services (see Equation 6.9). The overall latency of the aggregated service curve is the summation of all latencies $l_{i,j}$ (with $i = 1, ..., n$) of the chosen services (see Equation 6.10). Differences of the invoked services, as presented in Section 6.2 occur concerning the service rate $r_{i,j}$, the latency $l_{i,j}$, and the costs $c_{i,j}$ (positive correlation between costs and service rate is assumed).

$$\sum_{j=1}^{m_i} x_{i,j} = 1 \quad \forall \quad i = 1, ..., n \tag{6.11}$$

For optimization purposes for all m services ($j = 1, ..., m_i$) of task i a binary variable $x_{i,j}$ is introduced to model whether a service $S_{i,j}$ is included in the invocation plan or not. The constraint presented in Equation 6.11 ensures that only one service per task is used at the same time.

6.4.1 *Throughput Maximization*

Concerning the convolution of several service curves $\beta_{i,j}(t)$, the resulting service curve $\beta_{w,k}(t)$ has the slope of the service curve with the smallest slope as shown in Equation 6.9. This yields to the fact that the maximum achievable execution rate $r_{w,k,max}$ of the workflow is the minimum of all maximum execution rates $r_{i,j}$ of the available services per process step i. Equation 6.12 depicts an upper bound for the maximum achievable throughput in the considered workflow.

$$r_{w,k,max} = min_{i=1}^{n} \left\{ max_{j=1}^{m_i} \left\{ r_{i,j} \right\} \right\} \tag{6.12}$$

Accordingly, it is dispensable to invoke services with the highest execution rate $r_{i,j}$ per process step i in order to achieve a maximum throughput of the entire workflow. Instead, the overall throughput is bounded by the process step i with the weakest throughput. Hence, the intermediary may reduce the costs by rejecting the invocation of services with execution rates $r_{i,j}$ higher than the highest achievable execution rate $r_{w,k,max}$.

Assuming the intermediary has a fixed budget, denoted by C_{max}, a service composition with a maximum throughput can be computed by solving the optimization problem with the objective function 6.13 and the constraint 6.14.

$$Max\ F(\vec{x}) = min_{i=1}^{n} \left\{ \sum_{j=1}^{m_i} r_{i,j} x_{i,j} \right\} \tag{6.13}$$

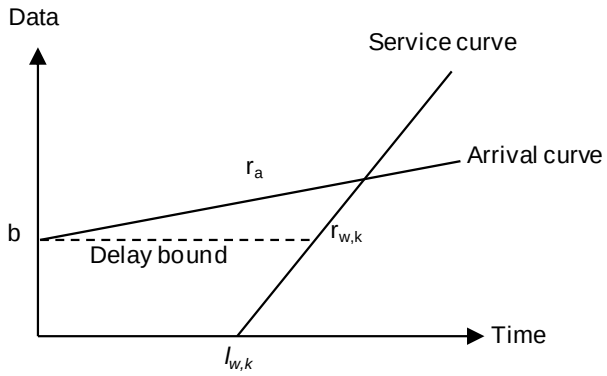


Figure 23: Delay bound

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{i,j} x_{i,j} \leqslant C_{max} \tag{6.14}$$

As there exists a positive correlation between service rate and costs, a purely throughput maximization approach would yield to an increase in overall costs C. Thus, this approach facilitates to constrain the overall service execution costs with C_{max}.

Concerning throughput maximization, the maximum achievable throughput is bound by the lowest execution rate of the invoked services in the entire workflow. A purely throughput optimization approach would be to identify the maximum achievable execution rate and to invoke only those services for the other tasks that have a higher or the same execution rate and that minimize the overall costs.

6.4.2 *Delay Minimization*

The delay of a workflow execution request is specified by the elapsed time between the arrival of an execution request at the service provider and the point of time at which the request is processed and forwarded back to the intermediary. The worst-case delay for a given workflow and a specific arrival behavior, as shown in Figure 23, can be easily computed by Equation 6.15 or 6.16 which are derived from Equation 6.8. In Equation 6.15 and 6.16, b determines the number of execution requests at $t = 0$ of the corresponding arrival curve (bulk arrival at $t = 0$).

$$d_{w,k} = \inf\{\tau : \beta(\tau) \geqslant b\} \tag{6.15}$$

$$d_{w,k} = l_{w,k} + \frac{b}{r_{w,k}} \tag{6.16}$$

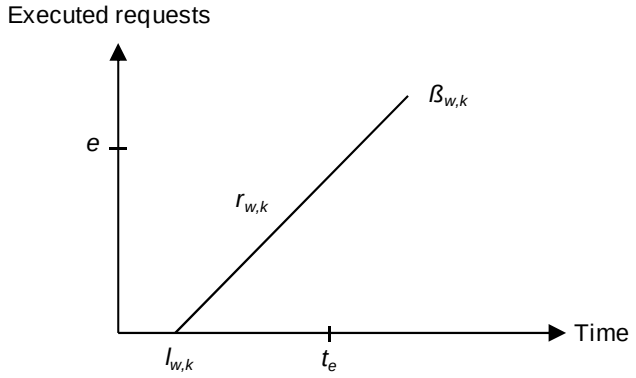


Figure 24: Fixed amount of workflow executions

For stability conditions and in order to be able to determine the worst-case delay $d_{w,k}$, the workflow execution rate $r_{w,k}$ has to be larger than the request arrival rate r_a. In case this condition is violated, this would imply that in the long run more requests arrive at the intermediary than it is able to serve. Besides the throughput optimization approach of the intermediary, another objective is the minimization of the worst-case delay of the entire workflow. Equation 6.17 describes the objective function for this optimization problem.

$$\text{Min } d_{w,k} = l_{w,k} + \frac{b}{r_{w,k}} \tag{6.17}$$

As depicted in the throughput optimization approach, this objective function will not be adequate in order to describe this problem, because without a constraint, the intermediary would always choose the services with the highest execution rates $r_{i,j}$ (at the highest costs). Hence, an important constraint in this optimization model is the introduction of a certain boundary C_{max} for the overall costs of the workflow execution as shown in Equation 6.14, in order to realize a cost-efficient invocation plan.

Summing up, a reasonable delay minimization approach would be to estimate a maximum acceptable delay for the service consumer and to minimize the costs by invoking those services which guarantee this worst-case delay bound.

6.4.3 *Cost Minimization*

Besides the aforementioned optimization approaches, a purely cost minimization approach is beneficial for the intermediary when ensuring that a specific amount of workflow executions e until a fixed deadline t_e has to be executed as depicted in Figure 24.

This optimization problem can be formulated by the main objective function shown in Equation 6.18 and the constraint depicted in Equation 6.19. The aim of the intermediary is to reduce the costs as much as possible with the con-

straint that the composed workflow is able to serve the incoming workflow execution requests e until the deadline t_e.

$$\text{Min } F(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{i,j} x_{i,j} \tag{6.18}$$

$$\beta_{w,k}(t_e) \geqslant e \tag{6.19}$$

In the case that there exists more than one possible solution for this optimization problem, the intermediary has to choose the service invocation plan, which minimizes the overall worst-case delay and maximizes the overall throughput.

6.5 SUMMARY

This chapter describes the adaptation of major insights of network calculus to service-based workflows, providing an analytical description of the worst-case performance behavior. Describing workflow arrivals and service execution patterns with arrival and service curves, key concepts of network calculus such as the delay bound are analyzed. Furthermore, optimization approaches concerning delay, throughput, and costs are described in detail. These optimization approaches support the intermediary to optimize the worst-case behavior of the workflow execution before the workflow execution starts.

The proposed optimization approaches support the intermediary in his decision process, i.e., the identification of the worst-case behavior of the workflow, in order to meet customer requirements. By applying these optimization approaches, the intermediary is able to optimize the worst-case behavior of the requested service-based workflow in advance and is able to avoid performance degradations before they occur.

7 RESOURCE PLANNING OF SERVICE-BASED WORKFLOWS

The presented QoS optimization approaches described in the two previous chapters have limited practicability concerning the development of a detailed service selection approach as they solely cover an average- or a worst-case performance analysis. In addition, these approaches assume a specific workflow execution arrival rate and do not support multiple parallel service selections. In a scenario with many workflow requesters and many service providers, multiple service invocations become necessary in order to serve a large amount of workflow execution requests. Thus, this chapter focuses on this challenge for the intermediary and presents a detailed resource planning approach for service-based workflows.

After an overview on related work, the system model of the analyzed scenario is presented. Furthermore, the detailed resource planning problem including an optimization approach and a description of the problem complexity is given. As a consequence, the developed heuristic solutions for this NP-hard problem and the architectural Workflow Performance eXtension – *WPX.KOM* for QoS management systems for service-based workflows are presented, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows.

7.1 RELATED WORK

This section gives an overview on related work in this field of research, aiming at the problem of service allocation in a scenario with limited service execution capacities and a large amount of concurrent workflow execution requests as exhibited in Chapter 3.

Research, coping with the challenges described in Chapter 3 progressed significantly in recent years. Especially in the context of application integration, many efforts have been done from both an industry and an academic point of view concerning standardization and interoperability of Web services as a means for realizing workflow composition. Industry standards, such as Business Process Execution Language (WS-BPEL) [LR02], Business Process Modeling Language (BPML) [Ark02], Web Service Description Language (WSDL) [CMRW07], [CHL+07] and Web Ontology Language for Web services (OWL-S) facilitate the workflow composition with the help of Web services and services in general.

Aggarwal defines the Web service composition and execution process as *"the process of realizing the requirements of a new Web service using the capability specifications of the existing component Web services"* [ACMS08]. A broad discussion concerning service composition approaches can be found in literature. Canfora

states in his work, that due to automated mechanisms it is possible that from a functional description, services can be found at run-time and invoked and composed to a specific workflow [CPEV05b]. For this purpose, it is necessary that besides well-defined functional interfaces the services are well-defined semantically [SERS08], [SRSS09]. The developed approach by Canfora aims at maximizing a distinct utility value for the workflow, being compliant to given SLAs and other side conditions such as cost, response time, availability, and reliability. In order to determine a solution with a low computational overhead, the developed heuristic is based on genetic algorithms.

Berbner et al. define QoS-aware Web service composition as *"the selection of Web Services maximizing the QoS of the overall Web Service composition, taking into account preferences and constraints defined by the user"* [BSR+06b]. In the mentioned literature, only service composition scenarios for a small number of concurrent execution requests are considered, assuming that the services are able (in terms of their execution capacity) to deal with the load of the workflow execution requests.

Especially the modeling of functional characteristics and the service composition problem concerning service-based workflows is under broad discussion. Mostly, semantic [Car02], [NM02], [SdF03] or rule-based [CIJ+00], [PF02] approaches have been proposed for this problem. However, the analysis of non-functional parameters, i.e., QoS and costs, and the impact on the service selection scenario need to be addressed in more detail. Those approaches treat the QoS behavior of entire workflows, by optimizing the selection of services concerning several QoS constraints. Preferences of the workflow orchestrator are modeled as restrictions and/or optimization criteria in an optimization model. Usually, as a measure for the overall QoS characteristics of a workflow, the specific QoS parameters of each workflow have to be aggregated to a merged value in order to optimize the service composition with regard to a specific optimization criterion.

As assumed previously, for each specific required functionality subsets of services are available on service markets, fulfilling those demands. The developed optimization approaches correspond to combinatorial selection problems that are proven to be NP-hard [MT87], [YL05], [PHD05]. In order to solve those optimization problems in a short computation time, which is needed to compose the workflow on-the-fly, service selection approaches require extremely fast and efficient heuristics.

Approaches for solving this problem are usually solved with graph-based models and combinatorial models. Graph-based approaches map the service selection problem to a Multi-constrained Optimal Path Problem. Heuristics for the Multi-constrained Shortest Path Problem with polynomial complexity are described in the work of Yu and Lin that is limited in its concrete implementation concerning time-critical requirements [YL05]. The authors state, that combinatorial approaches perform better and graph-based models can solve

multiple process plans at the same time (e.g., alternative workflow routes).

Combinatorial approaches map the composition problem to a Multi-dimensional Multi-choice Knapsack Problem (MMKP). Some authors solve those problems with the help of the integer linear programming method in order to receive an exact solution as an instance of the MMKP [ZBD+03], [AVMM04]. Due to its large computational overhead these approaches are considered as too complex [YL05], [BSR+06b], [YCNCC06]. In order to reduce the computational overhead, meta-heuristics can be used for solving the MMKP such as genetic algorithms [KBH94] , [CPEV05a], and simulated annealing [Ber08], improving run-time behavior and scalability compared to linear programming.

Currently, the most promising approaches in terms of computation time are greedy heuristics. Iterative improvement of an initially feasible solution can be done with the help of a replacement strategy, i.e., by replacing items or services with a low utility in correspondence to the aggregate resource consumption in each subset with items of a higher utility [KLMA02]. Local optima can be avoided by creating infeasible solutions, followed by iterative replacement in order to achieve feasibility [YL05].

Furthermore, Akbar et al. present an efficient greedy heuristic by applying a transformation technique to map the multi-dimensional resource space to a single dimension with the help of convex hulls [ARK+06]. Based on an adapted algorithm of Lee et al. [LLRS99], the authors heuristicly find a near-optimal solution for the MMKP.

A promising heuristic of Ykman-Couvreur et al. uses a greedy approach in order to solve the MMKP under time-critical conditions in embedded systems [YCNCC06]. The authors use pareto-filtering, followed by a dimension reduction of the search space. With the help of a penalty vector, the multi-dimensional search space can be reduced to a two-dimensional one in order to sort the achieved items according to an angular metric referring to the relative utility. In addition, a greedy replacement algorithm achieving a near-optimal feasible solution is applied to the sorted list.

Relaxation-based heuristics such as the approach of Berbner solve the linear programming relaxation of the combinatorial integer programming formulation (by simplex algorithm) and uses a backtracking algorithm to create a feasible solution based on the result of the relaxed integer program [Ber08]. Parra-Hernandez and Dimopoulos relax the MMKP to a Multi-dimensional Knapsack Problem [PHD05], which performs better in terms of solution quality than [KLMA02] but is worse in terms of computational complexity [YCNCC06].

Combinatorial approaches allow solving the depicted service selection problem under time-critical conditions showing significant outperformance in terms of computational complexity. However, graph-based approaches are able to

solve multiple composition plans at the same time.

However, after an extensive analysis, those approaches do not address the resource planning problem with a large amount of workflow execution requests and limited service execution capacities in which the workload is much larger than the services are able to serve within a specific time period. In recent research, it is always assumed that out of a subset of services only one service has to be selected, i.e., each of the services possesses a service execution capacity that is sufficient for the amount of workflow execution requests and that fulfills further QoS requirements. In order to cope with the challenges of the resource planning problem, the following sections provide an optimization approach and present efficient heuristics for solving this problem.

7.2 SYSTEM MODEL

As described in Chapter 3, this work focuses on cross-organizational service-based workflows. These workflows usually implement a certain business process with the help of services. The considered workflows consist of a finite number of process steps n that have to be executed consecutively (see Figure 6 on page 27). Furthermore, each workflow can be decomposed into several basic activities (tasks), each with a specific functionality. We assume that these tasks are arranged in a sequential order and that one task can start processing if its predecessor has already completed processing. Concerning service-based workflows, these tasks have to be implemented by services, which possess a particular functionality.

On a service market many services offering different functionalities at several cost and QoS levels exist. Thus, for task implementation, multitudes of services offering this specific functionality exist. These sets of services are referred to as service candidates. These service candidates can be services from internal as well as from external partners offered to specific conditions and on basis of specific hardware and network connections. Furthermore, these services vary in their non-functional characteristics. Out of these service candidates for the entire workflow, several service categories can be determined subsuming a set of services offering a specific functionality to the service requesters. Therefore, for each process step (task) a service category has to be defined, fulfilling a specific functionality as exhibited in Figure 25.

The determination of an invocation plan consisting of several services, providing the required functionality and offering a specific maximum utility given a specific optimization objective, is the challenge in solving the resource planning problem of such a service-based workflow. This invocation plan has to determine at which process step, which services have to be invoked, and how many services, if necessary, have to be invoked in parallel. A parallel invocation is necessary if the forecast load of the requested workflow execution exceeds the limited execution capacity of the considered service. Thus, for each workflow step one or more services out of a specific service category have to

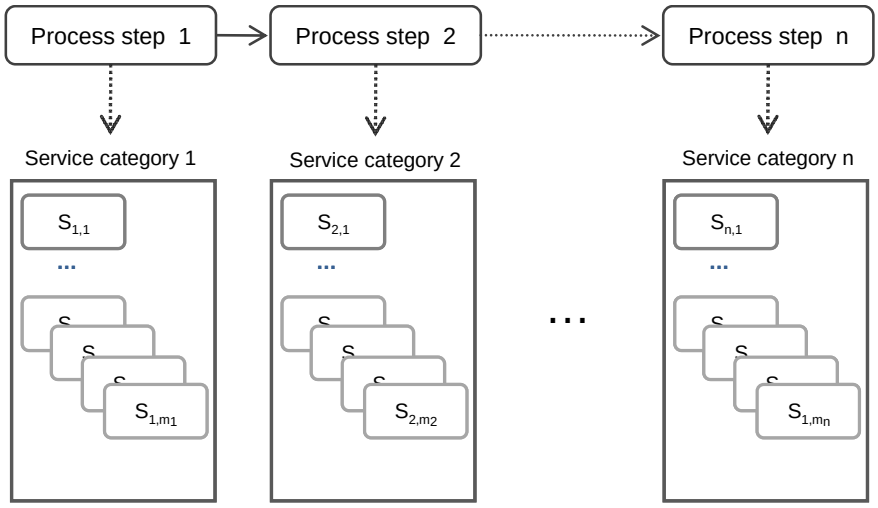


Figure 25: Service categories

be selected. The control logic of the tasks follows a sequential order whereas services may have to be invoked in parallel in order to serve all workflow execution requests. The goal of the intermediary is the cost-efficient execution of the workflow, meeting the QoS requirements of the workflow requesters and being able to serve all workflow execution requests. This implies a maximization of a defined utility provided by the entire workflow. This utility can be, e.g., costs or response time that has to be minimized or workflow execution capacity (resulting from the service execution capacity) that has to be maximized. In addition, the QoS properties have to cope with the demands of the workflow requesters. Several requester groups may have several needs concerning, e.g., the response time or the costs. The overall QoS behavior of the entire workflow is dependent on the QoS characteristics of the chosen services. In the described resource planning problem we assume that the services have two QoS parameters (response time and service execution capacity) and that the requesters are charged via a volume-oriented pricing model. The correlation between response time and costs is negative and the correlation between service execution capacity and costs is positive.

7.3 RESOURCE PLANNING APPROACH

The considered resource planning approach focuses on the selection and invocation of services with limited execution capacities to a workflow in order to be able to serve a large amount of workflow execution requests. This approach facilitates modeling of several workflows independent of the number of process steps and the number of services in each service category as described in the following.

7.3.1 *Non-functional Service Property Aggregation*

As mentioned earlier, each service category comprises a set of services with the same functionality. These services only differ in their QoS properties, the non-functional properties. Each service is attributed with a finite number of non-functional QoS parameters and costs, which are all specified by numeric parameter values (e.g., average response time = 60 ms).

The overall QoS property of the entire workflow can be calculated as an aggregated value as a result of the selected services in each category. For calculating the aggregated value, the intermediary has to distinguish between an aggregated value for the *service category* and for the entire workflow, whereas the latter is a result of the aggregation of the so called *service category QoS values*. Concerning value aggregation, four different mathematical operators are used in order to determine an aggregated QoS value. For this aggregation process, the *additive*, the *multiplicative*, the *min-* and the *max-operator* are used, whereas the application depends on the circumstances, i.e., on the type of QoS parameter and the aggregation context. Using the additive operator implies the summation of parameter values. This operator can be used for, e.g., the resulting costs in a workflow. Using the multiplicative operator causes the multiplication of parameter values (e.g., multiplication of the reliability of the utilized services of each category). By using the min-operator, the minimum of all parameter values is calculated (e.g., the maximum number of requests a composed workflow is able to serve within a specific time period is represented by the minimum of all execution capacities of the invoked services in each category). In contrast to the min-operator, the max-operator calculates the maximum value out of a subset of items. This is usually used, e.g., to determine the response time of each category as the maximum of all response times of the chosen services in one category.

For the QoS parameter aggregation, two kinds of aggregation on different levels have to be differentiated. On the one hand, the aggregation can be accomplished on *workflow level* and on the other hand on *category level*. The category level represents the aggregation of QoS parameters of services that are selected within a specific category whereas the workflow level uses the aggregated QoS parameters of the service category and specifies the aggregation of QoS parameters for the entire workflow. Dependent on the QoS value aggregation context, i.e., the level of aggregation and the QoS parameter, specific aggregation operators have to be applied. Table 3 provides an overview on the mathematical operators that have to be used in dependence of the aggregation level and the aggregated QoS parameter.

Concerning the resulting costs of service invocations, there is no difference in the aggregation level. Both, on category level as well as on workflow level the additive operator has to be used for determining the overall service invocation costs.

QoS Parameter	Category Level	Workflow Level
Costs	Additive	Additive
Execution capacity	Additive	Min-operator
Response time	Max-operator	Additive

Table 3: QoS aggregation specifications

Examining the execution capacity of a service, two kinds of operators have to be used for parameter aggregation. For the category level the additive operator is used, because services have to be invoked in parallel in order to be able to serve a defined amount of workflow execution requests. Concerning the workflow level, each aggregated overall execution capacity of the services used in one category can represent a potential bottleneck for the entire workflow – the min-operator has to be used in this case.

The services from a subsequent service category can be invoked at the earliest when all services from the previous service category have completed execution. This implies the usage of the max-operator on category level and the additive operator on workflow level for the response time. Following these aggregation specifications, the overall QoS behavior of a set of chosen services has to be calculated by the aggregation on category level and then on workflow level.

The service candidates for a specific workflow are usually denoted with $S_{i,j}$, whereas i represents the workflow step (service category) and j the specific service out of category i. For the indication of a specific non-functional parameter, a further parameter $p_{i,j,q} \geqslant 0$ is introduced. The index q ($q = 1, ..., z$) denotes the specific considered non-functional parameter. As costs, response time, and execution capacity are used as non-functional characteristics of services, three non-functional parameters are important in the considered model ($z = 3$). Execution capacity in the analyzed context is the maximum amount of concurrent service execution requests that are executable within a specific time period $t_{i,j}$ (response time of the service). This QoS value is of high importance, as the intermediary has to ensure that the chosen services (single or parallel invocations of one or multiple services) provide enough execution capacity for a large workload. Up to the execution capacity $cap_{i,j}$, a service $S_{i,j}$ always reveals a fixed response time $t_{i,j}$ and fixed costs $c_{i,j}$.

As for task execution one or multiple services can be invoked, in case of multiple services, invocation requests are processed in parallel, i.e., the response time of a task (category response time) equals the maximum response time of invoked services (e.g., if services with response times 24 ms, 31 ms, and 26 ms are invoked in parallel, the task is completed after 31 ms). The summation of all maximum response times of invoked services in each category represents the response time of the entire workflow, i.e., $\sum_{i=1}^{n} max\left\{t_{i,j} | x_{i,j} = 1\right\}$. Following this deliberation, a service $S_{i,j}$ can execute up to $cap_{i,j}$ service execution requests at a given response time $t_{i,j}$ irrespective of the actual service usage by

the intermediary. For the response time, this assumption reflects a worst-case observation in order to be able to guarantee a certain QoS level. In terms of costs, these assumptions imply a volume-oriented pricing model as discussed in Section 4.2.1.

In order to verify if a chosen set of services fulfills the QoS requirements, the non-functional parameters have to be aggregated in a two-step process (category level and workflow level aggregation), resulting in the calculation of p_q. Thus, for each non-functional parameter q in category i, an aggregated category parameter $p_{i,q}$ has to be computed as depicted in Equation 7.1, whereas $\diamondsuit$ is a placeholder for the specific operator (addition, multiplication, min-, or max-operator) that has to be used in dependence of the non-functional parameter q and the aggregation level. The binary variable $x_{i,j}$ indicates whether the considered service is selected or not.

$$p_{i,q} = \diamondsuit_{j=1}^{m_i} p_{i,j,q} * x_{i,j} \tag{7.1}$$

The aggregation of the category parameters $p_{i,q}$ for all i results in the specific non-functional workflow parameter p_q and can be calculated with Equation 7.2.

$$p_q = \diamondsuit_{i=1}^{n} p_{i,q} \tag{7.2}$$

In detail, the particular aggregation specifications on category level for the operators are as follows:

- Additive operation: $p_{i,q} = \sum_{j=1}^{m_i} p_{i,j,q} * x_{i,j}$
- Multiplicative operation: $p_{i,q} = \prod_{j=1}^{m_i} p_{i,j,q} * x_{i,j}$
- Minimum operation: $p_{i,q} = \min_{j=1}^{m_i} p_{i,j,q} * x_{i,j}$
- Maximum operation: $p_{i,q} = \max_{j=1}^{m_i} p_{i,j,q} * x_{i,j}$

The overall aggregated value of the non-functional parameters can be computed on workflow level as follows:

- Additive operation: $p_q = \sum_{i=1}^{n} p_{i,q}$
- Multiplicative operation: $p_q = \prod_{i=1}^{n} p_{i,q}$
- Minimum operation: $p_q = \min_{i=1}^{n} p_{i,q}$
- Maximum operation: $p_q = \min_{i=1}^{n} p_{i,q}$

Model 1 Optimization problem

Objective function:

$$MinF(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{i,j} * x_{i,j} \tag{7.3}$$

Constraints:

$$\sum_{j=1}^{m_i} cap_{i,j} * x_{i,j} \geqslant W \quad \forall i = 1, ..., n \tag{7.4}$$

$$\sum_{i=1}^{n} max\left\{t_{i,j} | x_{i,j} = 1\right\} \leqslant \tau \tag{7.5}$$

$$\sum_{j=1}^{m_i} x_{i,j} \geqslant 1 \quad \forall i = 1, ..., n \tag{7.6}$$

$$x_{i,j} \in \{0, 1\} \tag{7.7}$$

7.3.2 *Optimization Approach*

The objective of the considered resource planning approach is the optimization of a specific defined utility, i.e., in the considered scenario this can be costs, response time, execution capacity, or a combination of those.

As it is the objective of the intermediary to minimize the overall service invocation costs, the optimized non-functional parameter in this consideration is cost. The other non-functional parameters, as the response time and the execution capacity, have to cope with other constraints in the model. The developed optimization problem is defined in Model 1, specified by Equation 7.3 to 7.7. The indices and parameters used for the aggregation of parameter values and the optimization approach are depicted in Table 4.

As described in the optimization problem, the costs accounted for the execution of each service $S_{i,j}$ are denoted as $c_{i,j}$, whereas $t_{i,j}$ denotes its response time and $cap_{i,j}$ its specific execution capacity. In Equation 7.7 the binary variable $x_{i,j}$ is described, indicating whether the corresponding service $S_{i,j}$ is chosen in the respective composition or not.

The objective function, depicted in Equation 7.3, assures the selection of a set of services at minimal overall costs. Equation 7.4 ensures that in all process steps ($i = 1, ..., n$) a set of services is selected, that is able to handle all incoming workflows execution requests W. This can either be achieved by choosing one specific service with a potential execution capacity exceeding the amount of execution requests, i.e., $cap_{i,j} \geqslant W$ or a set of services whose sum of processing

Parameter / Index	Definition
$i = 1, ..., n$	Index for workflow step (category)
$j = 1, ..., m_i$	Index for service candidates in category i
$t_{i,j}$	Response time of service $S_{i,j}$
$cap_{i,j}$	Execution capacity of service $S_{i,j}$
$c_{i,j}$	Costs for the invocation of service $S_{i,j}$
$\tau > 0$	Response time restriction of the entire workflow
$W > 0$	Amount of workflow execution requests (workload)

Table 4: Indices and parameters of the optimization problem

capacities exceeds the requested execution capacity, e.g., $cap_{1,2} + cap_{1,5} \geqslant W$.

As described in Section 7.6, the workflow predictor prioritizes the requests into several classes with specific workflow execution deadlines. Furthermore, specific user groups are allocated to specific classes, attributed with a specific execution deadline τ. In order to guarantee workflow execution within τ time units, a bottleneck consideration has to be applied considering the execution capacity. The longest path through the workflow can be determined by considering the selected service with the maximum response time in each workflow step. Summing up the maximum response times in each workflow step, the overall resulting response time for passing all workload through the workflow is obtained. Equation 7.5 ensures, that the summation of all maximum response times of the selected services in each category holds the deadline τ, i.e., that the determined invocation plan meets a specific deadline τ.

After the description of the optimization approach, the next subsection focuses on the complexity of the presented problem.

7.3.3 *Problem Complexity*

Concerning the problem complexity, it can be stated, that if it is possible to reduce the optimization problem described in Model 1 to the MMKP (NP-hard [YL05]), the considered resource planning problem is NP-hard.

The MMKP is a derivative of the classical Knapsack Problem [MT87] in which the selected items have additional dimensions and restrictions. The difference between the resource planning problem and the MMKP is the amount of chosen items in each category. In the MMKP only one item per category is selected, whereas in the developed optimization model in some categories more than one item (service) has to be inserted into the knapsack (invocation plan) in order to be able to serve all incoming workflow execution requests.

The following provides the reduction of the resource planning problem to a MMKP and serves as a prove that the computational complexity of the analyzed resource planning problem is NP-hard.

In order to reduce the resource planning problem to the MMKP, it is necessary to expand the potential items in each category i to its power set in order to cover all possible service compositions. The QoS parameters as well as the costs for the services have to be aggregated when combining services by creating the power set of the services in one category, i.e., all possible combinations of services in one category. For the aggregation of the non-functional parameters and the costs, the specifications for the category level aggregation have to be applied, i.e., for response time aggregation the maximum operator and for costs and execution capacity the additive operator (see Section 7.3.1). Furthermore, Equation 7.6 has to be replaced by Equation 7.8, ensuring the selection of one service per category i.

$$\sum_{j=1}^{2^{m_i}} x'_{i,j} = 1 \quad \forall i = 1, ..., n \tag{7.8}$$

The power set $P(S)$ of a set S is defined as the set of all subsets of S. Assuming a category i containing a finite set S of services with a cardinality $|m_i|$, the power set of S has a cardinality of $|2^{m_i}|$. With the help of this reduction, the resource planning problem is proven to be NP-hard. The resource planning problem is a binary integer linear programming problem, as a special case of integer linear programming problems, where the discrete unknown variables are restricted to be 0 or 1. Because of discrete unknown variables, existing efficient algorithms in the field of linear programming (such as the simplex algorithm) cannot be applied in this case. For integer linear programming problems sophisticated search algorithms based on Branch & Bound, the cutting-plane method, or combinations of those are state of the art [KS04b]. However, these approaches cannot be applied for time-critical resource planning of service-based workflows due to an exponential increase in computation time, increasing with larger problem sizes and a growing complexity of restriction structures. Thus, heuristics are developed in order to solve these problems at short computation times and high solution qualities as discussed in the next sections.

7.4 HEURISTIC SOLUTION – MSS.KOM

This section introduces and describes the developed heuristic for the resource planning approach of service-based workflows. Concerning the high computation time of exact solutions, it is advantageous to determine feasible solutions with a high solution quality instead of optimal solutions for NP-hard problems. For this purpose, heuristics can be developed [DD98]. In contrast to the loss in solution quality by using a heuristic, the short computation time in contrast to exact solutions represents the most important advantage of heuristics. The solution quality of the heuristic is dependent on the properties and the procedure of the heuristic and the samples of the input data.

Due to the problem complexity (NP-hard) it is difficult to find an exact solution for the problem depicted in Section 7.2. Especially, with increasing

problem size (number of process steps and number of service candidates) and increasing complexity of restriction structure, it is not possible to find an exact solution that meets time-critical requirements. Furthermore, exponential computational complexity can be assumed for this problem in the worst-case.

Thus, a heuristic has been developed achieving a high computational performance and revealing a high solution quality. This heuristic, referred to as MSS.KOM, comprises three steps for solving the resource planning problem. The steps are as follows:

1. Valid solution: Determination of a feasible solution that meets constraint requirements.
2. Priority list: Sorting all service candidates in a priority list with a metric, measuring the relative utility to the non-functional properties.
3. Replacing algorithm: Approximation of a near-optimal feasible solution by a semi-random replacing algorithm.

The following sections describe the steps of the heuristic in detail.

7.4.1 *Valid Solution*

The developed heuristic aims at the determination of an invocation plan with near-optimal characteristics considering the objective function. This can be done by starting from a valid solution with the help of an iterative solution enhancement approach. A valid solution implies that primarily those service candidates are chosen for each category that provide the specific functionality and that meet the non-functional constraint requirements (see Equation 7.4 and 7.5). Thus, aggregate non-functional parameters have to be determined and compared with the constraints (response time and execution capacity).

In the system model in Section 7.2, we assume, the higher the execution capacity of a service, the higher the costs, and the higher the response time of a service, the lower the costs. Thus, it is supposed to be beneficial to prefer those service candidates with least consumption of resources in each category. Resource in this context is the response time which is constrained by a maximum overall response time τ.

In the first step of the heuristic, MSS.KOM prefers the selection of services with a low response time $t_{i,j}$ as depicted in Algorithm 2. Furthermore, we assume that the sum of service execution capacities per service category exceeds the workload. A valid solution beginning with an empty invocation plan ($Inv\!P$) is created as follows:

1. Sort all services in each category i ($cat_{resp\_list_i}[]$) in an ascending order regarding the response time starting with S_{i,α_0} which is the service with the smallest response time in category i ($t_{i,\alpha_0} = min\left\{t_{i,j} | j = 1, ..., m_i\right\}$).
2. Select stepwise the services for each category out of the sorted lists starting with service S_{i,α_0} ($S_{i,j} \in Inv\!P \Leftrightarrow x_{i,j} = 1$) until $cap_i \geqslant W$.

Algorithm 2 CREATE $InvP$

Require: W, τ
Output: $cat_{resp}\_list_i[]$, $InvP$

1: $InvP = \{\}$
2: $S_{i,j} \in InvP \rightarrow x_{i,j} = 1$
3: **for all** $i = 1, ..., n$ **do**
4: **for all** $j = 1, ..., m_i$ **do**
5: $cat_{resp}\_list_i[] \leftarrow S_{i,j}$
6: **end for**
7: **end for**
8: **for all** $i = 1, ..., n$ **do**
9: sort $cat_{resp}\_list_i[]$ ascending concerning $t_{i,j}$
10: $k = 0$
11: $cap_i = 0$
12: **while** $cap_i < W \wedge k < m_i$ **do**
13: $InvP = InvP \cup S_{i,j}$ at $cat_{resp}\_list_i[k]$
14: $t = \sum_{i=1}^{n} max\left\{t_{i,j} | x_{i,j} = 1\right\}$
15: $cap_i = \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}$
16: **if** $t > \tau$ **then**
17: return -1
18: **end if**
19: $k++$
20: **end while**
21: **end for**

3. Proceed with step 2 until services for all categories are chosen and check whether Equation 7.5 is still met. If the constraints are met, the selected services in $InvP$ represent a valid initial solution.

The results of step 1 are sorted lists ($cat_{resp}\_list_i[]$) in which services are sorted in an ascending order concerning their response time values, i.e., $t_{i,\alpha_k} \leqslant t_{i,\alpha_{k+1}}$ $\forall k = 0, ..., (m_i - 1)$. If step 3 concludes that the constraint 7.5 is not met, there exists no feasible solution for the considered resource planning problem.

Remembering, the considered non-functional parameters follow either an additive or a min-operator on workflow level. On workflow level the execution capacity follows a min-operator. This constraint can be checked locally in each category i. The response time on the other hand follows on the workflow level an additive operator. The response time for each category t_i is determined by the maximum response time of the chosen services in the category. Choosing services from each category with ascending response times until the demanded execution capacity is reached, leads to the response time minimal selection of services for a specific workload W (at high costs that can be minimized further). There exists no other set of services that has a smaller response time – this facilitates that both constraints, i.e., 7.4 and 7.5, are met, if a feasible solution exists. In case that due to this selection procedure one of the

constraints is violated, no feasible solution exists.

Adding more constraints and including more non-functional parameters to the optimization problem complicates the process of finding a valid initial solution, because if more than one parameter follows an additive operator on workflow level, this approach cannot guarantee to find a feasible initial solution. With the help of relaxations and a focus on the costs (including sorted lists by the costs of services) it may be possible to find a valid solution. Although, considering the analyzed optimization problem a valid initial solution can be found with the help of this approach.

7.4.2 *Priority List*

After the determination of an initial valid solution, step 2 provides an approach for a sorted priority list in which the services are listed with a help of a metric, measuring the relative utility to the non-functional properties of the services. As it is the objective, outgoing from a feasible solution, to reduce the overall costs for service invocation, the constraint space in accordance to the response time constraint has to be exhausted further. Therefore, a priority list is developed sorting the services in a way that preferably those services are selected that promise the most utility at least consumption of resources, i.e., the services have to be sorted with respect to their cost-effectiveness.

Thus, a metric $s_{i,j}$ for the cost-effectiveness measuring the resource consumption and the costs of service invocation in terms of the resources (constraints in the optimization model) is defined. The resource consumption in this case is denoted by $RC_{i,j}$ and measures to which extent the selected service "uses" a limited resource (corresponding to the response time in the considered model). The smaller the value of $s_{i,j}$ the higher is the cost-effectiveness of the selected service. The developed metric $s_{i,j}$ leads to $c_{i,j} * RC_{i,j}$, as lower costs and lower resource consumption is beneficial for the selection of a service in order to maximize the utility of the objective function (here minimize the costs) and to hold the constraints.

The measure for the resource consumption $RC_{i,j}$ can only be used for the response time due to the fact that a lower response time increases the utility. Concerning the execution capacity, another approach has to be followed because a higher execution capacity increases the overall utility and cost-effectiveness of a service. Therefore, the measure $s_{i,j}$ has to be refined and divided by the relative offered execution capacity of the specific service candidate (execution capacity offered by the service $cap_{i,j}$ at the ratio to the demanded workload W). This can be explained by less offered service execution capacity implies less overall utility in terms of service invocation, because a small offered execution capacity implies to invoke other services (at specific costs) in order to meet the demanded constraint requirement (Equation 7.4).

Algorithm 3 Creation of $global\_list[]$ and $cat\_list_i[]$

Require: $c_{i,j}$, $t_{i,j}$, $cap_{i,j}$, W
Output: $global\_list[]$, $cat\_list_i[]$

```
1: for all i = 1, ..., n do
2:     for all j = 1, ..., m_i do
3:         s_{i,j} = (c_{i,j} * t_{i,j}) / min{1, cap_{i,j}/W}
4:         global_list[] ← s_{i,j}
5:         cat_list_i[] ← s_{i,j}
6:     end for
7: end for
8: sort global_list[] ascending
9: for all i = 1, ..., n do
10:    sort cat_list_i[] ascending
11: end for
```

Furthermore, it will not be beneficial to invoke a service with a higher execution capacity as the demanded workload, because any execution capacity exceeding the demanded workload will not provide additional utility. Thus, the metric $s_{i,j}$ will be divided by $min\left\{1, \frac{cap_{i,j}}{W}\right\}$ ensuring that a higher execution capacity $cap_{i,j}$ than the demanded workload W will not unnecessarily increase the utility of a service. This leads to a refined metric for $s_{i,j}$ where $RC_{i,j} \hat{=} t_{i,j}$:

$$s_{i,j} = \frac{c_{i,j} * RC_{i,j}}{min\left\{1, \frac{cap_{i,j}}{W}\right\}} = \frac{c_{i,j} * t_{i,j}}{min\left\{1, \frac{cap_{i,j}}{W}\right\}} \tag{7.9}$$

Equation 7.9 describes the cost-effectiveness of a service candidate $S_{i,j}$ represented by $s_{i,j}$. Following this deliberation a smaller value of $s_{i,j}$ represents a higher cost-effectiveness. Thus, sorted lists arrange the services in an ascending order starting with the service $S_{i,j}$ with the smallest value $s_{i,j}$, i.e., the smaller a value $s_{i,j}$, the higher is its priority and its capability for a solution improvement.

The idea is to consecutively exchange the services, selected in the initial valid solution InvP, in order to achieve a smaller value for the objective function 7.3. This global priority list, listing all service candidates in ascending order, is denoted as $global\_list[]$ and has a length of $\sum_{i=1}^{n} m_i$. The replacement of chosen services by services from the $global\_list[]$ is accomplished by removing all previously selected services in category i and replacing them with services from the global priority list. As a single replacement of a service in one category does not necessarily satisfy the constraint requirements of the workload, further services for this specific category have to be assigned to this category and therefore to InvP. Besides the $global\_list[]$, it is necessary to create a further list for each category in order to be able to select a service with a high cost-effectiveness. Thus, by creating the $global\_list[]$, services will also be arranged in category lists for each category i, referred to as $cat\_list_i[]$. The

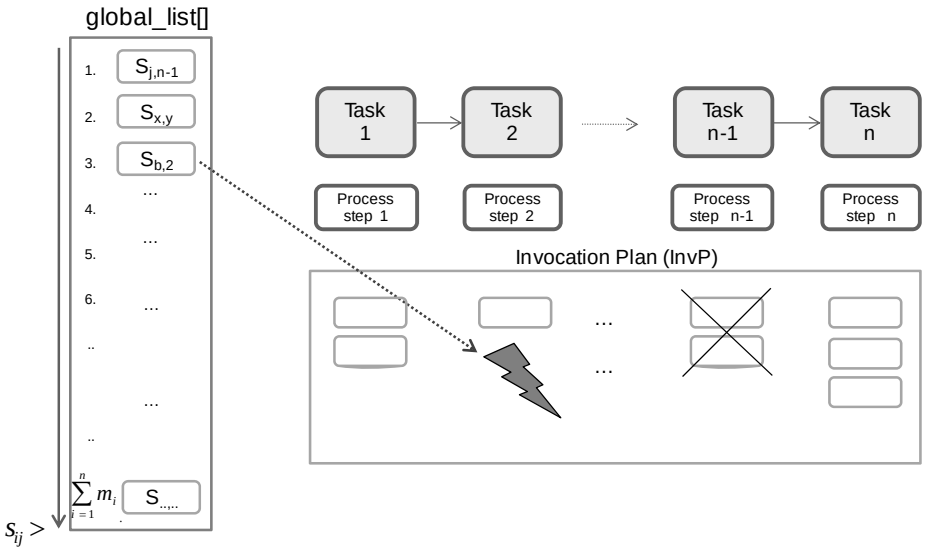


Figure 26: Processing of ordered priority list

creation of those lists is described in Algorithm 3. By creating $cat\_list_i[]$ it has to be noted that these lists change dynamically with the progress of service replacement. This occurs, because of placing a service $S_{i,j}$ into a process step i of the invocation plan reduces the remaining required execution capacity resulting from Equation 7.4 by $|cap_{i,j}|$. Hence, the priority of the remaining services in $cat\_list_i[]$ has to be recalculated with the help of the category list metric $cl_{i,j}$ incorporating the remaining workload demand.

$$cl_{i,j} = \frac{c_{i,j} * t_{i,j}}{\min\left\{1, \frac{cap_{i,j}}{W - \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}}\right\}} \tag{7.10}$$

MSS.KOM automatically creates and recalculates the category lists $cat\_list_i[]$ at relatively small additional storage requirements. The sorting operations in this heuristic for $global\_list[]$ and $cat\_list_i[]$ are implemented with a Quicksort algorithm.

7.4.3 *Replacing Algorithm*

In the third step of the developed heuristic MSS.KOM, a developed semi-stochastic replacing algorithm is applied to the sorted lists $global\_list[]$ and $cat\_list_i[]$. This algorithm consecutively passes through all items (services) in the $global\_list[]$ and replaces services from the InvP as presented in Figure 26. With each chosen item in the list, the solution's utility can be increased by saving costs and offering a high execution capacity of the services.

Assuming an initial valid solution, InvP is primarily attributed with a specific utility U representing a measure concerning the objective function. Starting with the selection of the first service candidate $S_{i,j}$ in $global\_list[]$ (service with the highest priority, i.e., lowest $s_{i,j}$ (at $global\_list[0]$)), this service will be added to category i as explained in Section 7.4.2 by removing all services of

Algorithm 4 Replacement

Require: $InvP$, U, $global\_list[]$, $cat\_list_i[]$
Output: $InvP_h$, U_h

1: **for all** $k = 0, k < \sum_{i=1}^{n} m_i$, k++ **do**
2: $S_{i,j}^k = S_{i,j}$ at $global\_list[k]$
3: $InvP' = InvP \mid$ all $x_{i,j} = 0$ in category i of selected $S_{i,j}^k$
4: $InvP'' = InvP' \cup S_{i,j}^k$
5: $cap = \min_{i=1}^{n} \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}$
6: $t = \sum_{i=1}^{n} max\{t_{i,j} | x_{i,j} = 1\}$
7: FILL $cat\_list_i[]$ at $S_{i,j}^k$
8: **if** $U'' > U \wedge cap \geqslant W \wedge t \leqslant \tau$ **then**
9: set $InvP = InvP'' \wedge U = U''$
10: **else**
11: select random $S_{i,j}^p$ at $global\_list[p]$, $p = k - \delta$, $\delta \in 1, ..., k \mid i'$ at $S_{i,j}^p \neq i$ at $S_{i,j}^k$
12: $InvP_{i'} = InvP \mid$ all $x_{i,j} = 0$ in category i of selected $S_{i,j}^k$ and in category i' of selected $S_{i,j}^p$
13: $cap = \min_{i=1}^{n} \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}$
14: $t = \sum_{i=1}^{n} max\{t_{i,j} | x_{i,j} = 1\}$
15: FILL $cat\_list_i[]$ at $S_{i,j}^k$
16: FILL $cat\_list_i[]$ at $S_{i,j}^p$
17: **if** $U_{i'} > U \wedge cap \geqslant W \wedge t \leqslant \tau$ **then**
18: set $InvP = InvP_{i'} \wedge U = U_{i'}$
19: **end if**
20: **end if**
21: **end for**
22: $InvP_h = InvP \wedge U_h = U$

this category that have been selected before in $InvP$, resulting in $InvP''$. If the demanded workflow execution capacity is not reached, further services out of $cat\_list_i[]$, sorted by the measure $cl_{i,j}$ in ascending order are integrated until constraint 7.4 is met. This yields to a new subset of services chosen for category i, satisfying the required workload demand resulting in a new invocation plan $InvP''$. If the resulting utility is larger than the previous one ($U''>U$) and the response time constraint 7.5 is not violated, the new invocation plan $InvP$ is set to $InvP''$.

Besides satisfying the constraint requirements and achieving a higher utility, the new invocation plan $InvP''$ may also induce constraint violations or achieve a lower utility. Under these circumstances $InvP''$ will be refused and the heuristic could proceed with the next item in the priority list ($global\_list[1]$). This approach could induce local optima that have to be avoided if possible. Therefore, MSS.KOM facilitates the revision of previously made decisions as described in the following.

Algorithm 5 FILL $cat\_list_i[]$

Require: $InvP_{current}$, W, τ, i
Output: $InvP''$

1: **for all** $l = 0, cap < W \wedge t \leqslant \tau$, l++ **do**
2: $InvP'' = InvP_{current}$
3: sort $cl_{i,j} = \frac{c_{i,j} * t_{i,j}}{min\left\{1, \frac{cap_{i,j}}{W - \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}}\right\}}$ for the specific category ascending
4: select $S_{i,j}^l$ at $cat\_list_i[l]$
5: $InvP'' = InvP'' \cup S_{i,j}^l$
6: $cap = min_{i=1}^{n} \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}$
7: $t = \sum_{i=1}^{n} max\left\{t_{i,j} | x_{i,j} = 1\right\}$
8: **if** $t > \tau$ **then**
9: $InvP'' = InvP'' \backslash S_{i,j}^l$
10: **end if**
11: **end for**

Assuming a service item $S_{i,j}$ in $global\_list[x]$ from category i and the added services of this category (until the execution capacity constraint is met) cannot be inserted to a new $InvP''$ due to constraint violations or a smaller utility U'', the algorithm selects a random item (service candidate) at position $x - \delta (\delta \in 1, ..., x)$ in $global\_list[]$, i.e., at $global\_list[x - \delta]$. This item has a higher priority and has already been considered for replacement. Furthermore, as a precondition this service has to belong to another category than i, i.e., $i' \neq i$. In the next step at category i' of $InvP$ all selected services have to be removed, resulting in additional space concerning the constraint equations (creation of $InvP_{i'}$). By achieving additional space, the heuristic tries to insert the service at $global\_list[x]$ in category i and fills the category with additional services out of $cat\_list_i[]$, sorted by $cl_{i,j}$ in ascending order. If this is not possible (i.e., the constraints are violated or the utility is not further increased), all changes will be discarded and the heuristic will proceed with the replacement approach at $global\_list[x + 1]$. In case the clearance of services in category i' and the resulting replacement of services in category i performs, category i' has to be filled with services from category i'. Assuming the replaced service at $global\_list[x - \delta]$ is at position y in its category list $cat\_list_{i'}[]$, MSS.KOM selects the services in ascending order out of $cat\_list_{i'}[]$ beginning at $cat\_list_{i'}[0]$ and tries to fill up category i'. If the created invocation plan $InvP_{i'}$ leads to an increased utility and does not violate any constraint, then $InvP = InvP_{i'}$.

The third step of the heuristic, the replacement approach, is formulated in Algorithm 4 using Algorithm 5. The heuristic finally determines an invocation plan $InvP_h$ offering the best subset of services providing a specific utility U_h. Before a detailed evaluation of the developed heuristic MSS.KOM is described in Chapter 8, the next section presents the developed single service selection heuristic 3S.KOM which is developed in order to achieve comparability to state of the art research.

7.5 SINGLE SERVICE SELECTION HEURISTIC – 3S.KOM

As mentioned in Section 7.1 also other approaches for solving service selection problems exist. As a comparison, existing heuristic solutions, adapted to the resource planning problem, are used in the following. In detail, the heuristics H1_RELAX_IP and H2_SWAP are chosen, because of the knowledge about the operating mode and the detailed parameter influencing the performance of the heuristic [Bero8].

In order to compare the developed heuristic MSS.KOM with an existing heuristic, an adapted heuristic solution has been developed: 3S.KOM. This heuristic uses the existing heuristic solutions H1_RELAX_IP and H2_SWAP and adapts these to the considered resource planning approach presented in Model 1 on page 73 with the difference that in Equation 7.6 only one service per category can be selected. Thus, Equation 7.6 has to be replaced by Equation 7.11. Furthermore, in order to facilitate the handling of multiple service selections in each category i, 3S.KOM prepares the input data before applying H1_RELAX_IP and H2_SWAP.

$$\sum_{j=1}^{2^{m_i}} x'_{i,j} = 1 \quad \forall i = 1, \ldots, n \tag{7.11}$$

As in the resource planning problem multiple services per category can be selected, it is needed to build up aggregated services out of the services m_i in each category. In order to create all possible service combinations in one category, the power set $P(S)$ of alternative services is determined. After building the power set, each category contains 2^{m_i} service candidates that is taken into account in Equation 7.11 and has to be adapted in Equations 7.3 and 7.4. The considered non-functional parameters of the services such as response time, execution capacity, and costs have to be aggregated with the help of the mathematical operators for the category level aggregation as described in Section 7.3.1. The aggregated services have to be listed in a new composed category list $comp\_cat\_list_i[]$ containing 2^{m_i} aggregated services. The non-functional aggregated parameter values of the aggregated services are represented as $t'_{i,j}$, $c'_{i,j}$, and $cap'_{i,j}$, whereas j ranges from 1 to 2^{m_i} in each category.

After this transformation, the problem is reduced to a Multi-choice Multi-dimensional Knapsack Problem and can be solved with existing heuristics. Due to the large computation time of those problems, it is beneficial to reduce the number of services per category further, taking into account that a large reduction may induce worse values of the objective function. Thus, the number of service candidates in $comp\_cat\_list_i[]$ is reduced with the help of the following algorithm in each category during the aggregation phase:

1. Check whether the aggregated service fulfills the capacity constraint, i.e., check whether $cap'_{i,j} \geqslant W$. If this is not the case, do not insert $S'_{i,j}$ in $comp\_cat\_list_i[]$.

2. Check whether the aggregated service has lower costs or takes less time than the aggregated service with the lowest *costs* so far, if not, do not insert $S'_{i,j}$ in $comp\_cat\_list_i[]$.

3. Check whether the aggregated service has lower costs or takes less time than the aggregated service with the lowest *time* so far, if not, do not insert $S'_{i,j}$ in $comp\_cat\_list_i[]$.

After this aggregation, the search space can be further reduced by applying the following algorithm:

1. Sort $comp\_cat\_list_i[]$ in ascending order concerning the costs $c'_{i,j}$.

2. Remove all services that have a higher or the same response time at a higher or the same cost level than any other service in category i, i.e., remove a service if $t'_{i,j} \geqslant t'_{i,k} \wedge c'_{i,j} \geqslant c'_{i,k}$ for $k \neq j$.

This algorithm reduces the search space further in order to increase the performance of the applied heuristics. The number of services in each category after the transformation in Equation 7.3 and 7.11 can be replaced by m'_i, whereas $m'_i \leqslant m_i$. Because of considering the characteristics of the aggregated values during the aggregation process, the following argumentation for $m'_i \leqslant m_i$ holds. As the response time is aggregated by a maximum operator on category level, after the creation of the power set only m different response time values exist in the set of services. Since costs follow an additive operator, the services with the same response time value solely differ in their costs. The service with the minimum cost value dominates the other services, leading to a deletion of the remaining services with identical response times. This proves, that the number of services after the reduction m'_i is smaller than m_i. As all aggregated services have to fulfill the capacity constraint, also less than m_i response time values can exist leading to a further decrease of m'_i. In addition, the application of step 1 in the aggregation phase further reduces the complexity of the optimization problem. Equation 7.4 is not needed any more and can be deleted. In order to adapt the overall response time constraint to the transformation and reduction process, Equation 7.5 has to be replaced by Equation 7.12.

$$\sum_{i=1}^{n} \left\{ t'_{i,j} | x'_{i,j} = 1 \right\} \leqslant \tau \tag{7.12}$$

After the search space is further reduced, the following heuristic solutions are applied on the resulting set of services [Ber08]:

1. Apply H1_RELAX_IP, i.e., solve the relaxed problem (integer variables are reduced to real variables) with *lp-solve*[1] (simplex algorithm) and determine a feasible solution with the help of a backtracking solution.

1 http://lpsolve.sourceforge.net/5.5/, Version 5.5.0.13

2. Apply H2_SWAP, i.e., swap services randomly in the solution with other services and try to improve the solution quality, i.e., the objective function.

As this heuristic solution can also be used for solving the resource planning problem, Chapter 8 provides a comparative analysis of this heuristic solution with MSS.KOM. After the description of the heuristic solutions for solving the resource planning problem of service-based workflows, the following section introduces a suggested generic architectural extension for QoS management systems for service-based workflows.

7.6 WORKFLOW PERFORMANCE EXTENSION – WPX.KOM

This section provides an overview on a suggested generic architectural Workflow Performance eXtension – WPX.KOM for QoS management systems for service-based workflows. In particular, WPX.KOM is introduced in order to extend the developed proxy architecture WSQoSX.KOM [Ber08], [Eck08]. This proxy architecture is an architecture for managing arbitrary Web service workflows that is able to compose workflows from multiple services while considering multiple QoS demands, such as response time or costs. The modular architecture consists of several components such as an Accounting, QoS Monitoring, and SLA Management component. From a QoS perspective, the most important component in this architecture is the service selection component. These architectures do not consider multiple parallel workflow execution requests and parallel service invocations.

The presented resource planning approach aims at the determination of invocation plans with a high solution quality for a large amount of workflow execution requests and the associated problem of limited service execution capacities. The question is which components are necessary to handle a large amount of incoming requests from different priority classes with several QoS requirements. To overcome these challenges, WPX.KOM is suggested in this context. The architectural extension WPX.KOM addresses the resource planning problem described in the previous sections, facilitating the realization of the resource planning approach in such an architecture. WPX.KOM enhances the architecture with two mandatory components in such a scenario: the resource planner and the workload predictor component (see Figure 27).

The workload predictor monitors incoming workflow execution requests and forecasts future workloads with the help of historic data or user behavior. The resource planner checks whether the execution capacity of any service is reached, replaces services by other services, and specifies which services have to be invoked in parallel in order to be able to serve all incoming requests. Thus, the resource planner has to create near-optimal service invocation plans with the help of the developed heuristics MSS.KOM and/or 3S.KOM. In detail, the tasks of the two architectural components in WPX.KOM are:

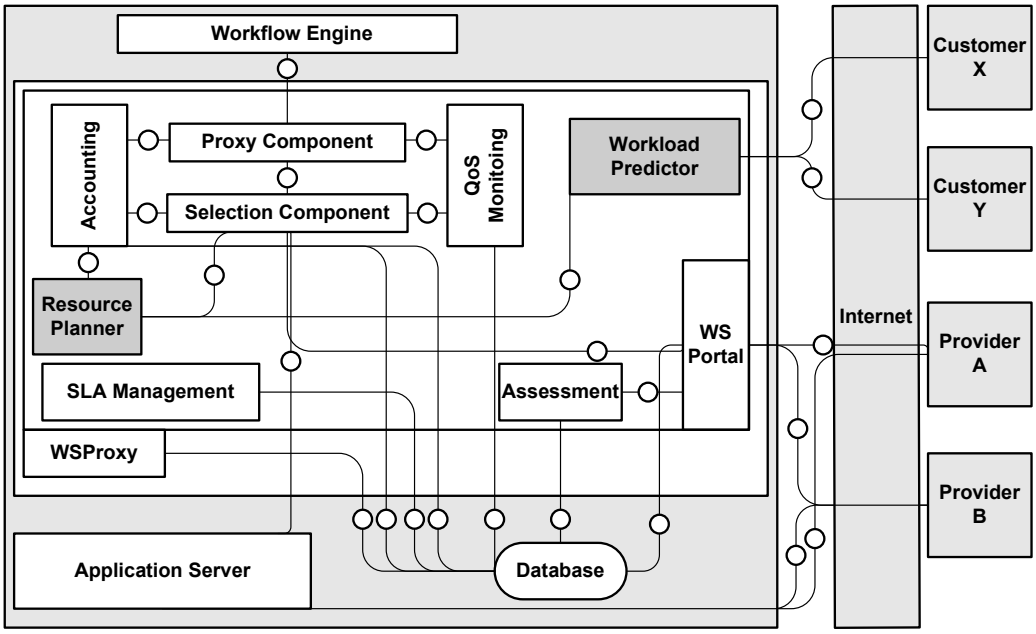


Figure 27: Workflow Performance eXtension – *WPX.KOM*

Workload Predictor:

- Monitoring of incoming workflow execution requests
- Quantitative forecasting
- Request Prioritization

Resource Planner:

- Checking of service execution capacity
- Replacing of services or determination of parallel service invocations
- Solving the resource planning problem with the developed heuristic solutions

The resource planner receives its information from the workload predictor and has to communicate with the accounting, the selection, and the QoS monitoring component. For resource planning purposes it is mandatory to have an efficient workload prediction due to the fact that the resource planning approach can only be efficient if the amount of incoming workflow execution requests is well predicted. For resource selection, the resource planner has to gather information about costs and QoS properties of invoked services to build up an efficient invocation plan for the next planning period. In this component, the developed heuristics can be implemented in order to realize an efficient solution for the resource planning problem of service-based workflows.

7.7 SUMMARY

This chapter describes the resource planning approach and develops an optimization model for solving the addressed resource planning problem for an

intermediary in order to provide a high level of QoS to his customers. This approach is described by giving an overview on related work and a detailed description of the system model including several assumptions. Furthermore, the resource planning problem is discussed in detail, providing methods for the non-functional service property aggregation as well as the formulation of the optimization problem. Based on the NP-hard complexity of this optimization problem, an efficient heuristic MSS.KOM is developed that is able to create invocation plans with multiple parallel service invocations taking into account several QoS parameters and costs. In addition, a single service selection heuristic 3S.KOM is developed that reduces the solution space of the optimization problem to a minimum and applies existing solution approaches from the operations research to the optimization problem. Furthermore, an architectural Workflow Performance eXtension – WPX.KOM for QoS management systems for service-based workflows is proposed, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows.

8 EVALUATION

After a detailed description of the proposed heuristic solution in Section 7.4, this chapter provides a detailed evaluation of MSS.KOM. The results of the developed heuristic MSS.KOM are compared to an exact solution (integer programming approach) and to the single service selection heuristic 3S.KOM. Especially the computation time and the solution quality are analyzed with respect to influencing factors. All experiments were run on an Intel Core 2 Duo (1.86 GHz) system with 2 GB of RAM, running Windows XP. The evaluation is conducted regarding the following influencing factors:

- Strength of resource constraints (overall response time τ and workload W).
- Problem size (number of process steps n and number of service candidates per process step m_i).
- QoS-CoS correlation of the services of the test cases.

The evaluation starts with the description of the setup and an investigation of the developed heuristic MSS.KOM. In detail, the analysis investigates the influence of the QoS-CoS correlation of the test cases (Section 8.2), the influence of the restriction strength (Section 8.3), and the influence of the problem size on the computation time and on the solution quality of MSS.KOM in contrast to the exact solution (Section 8.4). In addition, the heuristics MSS.KOM and 3S.KOM are analyzed and compared regarding the computation time and the solution quality (Section 8.5).

8.1 SETUP

As the current service market is not yet developed, the evaluation cannot be based on real test data. Instead, for evaluation purposes a large variety of test cases representing the attributes of the services such as response time and execution capacity are created. As there is a need for realistic bounds for the constraints of the optimization model, i.e., the overall response time and the workload (number of workflow execution requests), restriction strengths are considered measuring the constraint restriction in dependence of the test cases. Thus, the overall response time τ and the workload W are arbitrary set to several values investigating the influence of the response time or workload value on the performance of the developed heuristics.

Thus, a variety of test cases has been generated in order to realize a significant evaluation. These test cases have been used for solving the described resource planning problem in its entirety. Each of the generated services in the test cases is attributed with the following parameters:

- Costs (charged via a volume-oriented pricing model, cent per service execution capacity).
- Response time (ms).
- Service execution capacity (number of possible service invocations).

For the data generation and the optimization approach a prototype has been developed. This software is based on the Java JDK version 1.6.0.06. For the determination of an exact solution for the optimization approach the mixed integer linear programming solver *lp-solve* in Version 5.5.0.13[1] is used that applies the simplex algorithm for linear programming problems and the Branch & Bound method for integer linear programming problems. The data generator allows the creation of test cases as depicted in Figure 42 in the appendix on page 129. All test cases can be specified by determining the correlation between the parameters, the stochastic components, the variation of parameters as well as the operators on workflow and on category level that have to be used for QoS parameter aggregation. Furthermore, the number of test cases, the number of workflow steps, and the number of service candidates per category can be adjusted as shown in Figure 43 in the appendix on page 130.

As Section 7.4 describes the heuristic with the help of a utility value U, it is the objective to minimize the overall service invocation costs, i.e., the desired utility maximization equals overall cost minimization. In this consideration, the costs of a service measure the utility of a certain invocation plan. Furthermore, several providers offer services at several cost levels, trying to reflect qualitative characteristics. As mentioned previously, the remaining non-functional characteristics are correlated to the cost parameter. The response time is negatively correlated, as a shorter response time is more expensive than a higher response time of a service. The service execution capacity is positive correlated to the costs, as a provider will charge more if the offered service execution capacity is higher.

The cost parameter, reflecting the inverse utility of a service, is generated by a uniform distributed random variable within a specific interval and a certain mean value. The interval in which the costs for the services are generated is $[40;160]$ with a mean value of $\frac{40+160}{2} = 100$. A service offers a certain amount of service invocations $cap_{i,j}$ and 100 cents are charged on average, independent of the real usage of the service up to $cap_{i,j}$.

As it cannot be assumed that the relation between costs and response time or costs and service execution capacity is strong linear, a stochastic component in test case generation is included. Both the random variables for response time and for service execution capacity are defined as $P_z = (a * P_{cost} + b) * SP_z$. A linear correlation is assumed with a stochastic parameter SP_z representing a uniformly distributed random variable within the interval $SP_z \in$

1 http://lpsolve.sourceforge.net/5.5/

$[1 - x_z; 1 + x_z]$. The value x_z, as a real number between 0 and 1, can be adjusted in test case generation. This influences and determines the correlation of the QoS parameters with the cost parameter influencing the values for response time and execution capacity. The induced stochastic variation of P_z by the random variable SP_z can be adjusted by increasing or decreasing the value of x_z.

The following uniformly distributed random variables are defined for test case generation:

1. $P_{cost} \in [40; 160]$
2. $P_{resp} = (-3.\overline{3} * P_{cost} + 833.\overline{3}) * SP_{resp}$

   $\Rightarrow P_{resp} \in [(1 - x_{resp})300; 700(1 + x_{resp})]$
3. $P_{cap} = (3.\overline{3} * P_{cost} + 166.\overline{6}) * SP_{cap}$

   $\Rightarrow P_{cap} \in [(1 - x_{cap})300; 700(1 + x_{cap})]$

As depicted above, the mean value for the costs is 100 cent/service usage up to $cap_{i,j}$. The mean value of the execution capacity is 500 requests within a mean response time of 500 ms for all requests up to 500 service invocations. The cost parameter varies about 60% around the mean value of 100 up and down. The assumed linear correlations between response time and costs and service execution capacity and costs have a variation about 40% around the mean value of response time and service execution capacity. Solely the algebraic sign whether it is positive or negative denotes a positive or a negative correlation. As a stochastic variation has been preferred, this correlation is refined by a stochastic argument SP_{resp} (SP_{cap}) which is adjusted by the parameter x_{resp} (x_{cap}). By setting x_{resp}, respectively x_{cap} to a specific value, the QoS-CoS correlation of the generated parameters can be adjusted. This is used in Section 8.2 for generating several test cases with varying correlations between costs and the other non-functional parameters in order to analyze whether the QoS-CoS correlation has an impact on the solution quality or on the computation time of the heuristic.

For the QoS-CoS correlation, the Pearson product-moment correlation coefficient $r_{x,y}$ is introduced as a dimensionless measure for the degree of a linear correlation between two test series [BBK08]. It indicates the strength and the direction of a correlation with a value, ranging from -1 to 1. In case $r_{x,y} = 1$, a full positive linear relation between the test series x and y and in case of $r_{x,y} = -1$ a full negative linear relation can be assumed. In case $r_{x,y} = 0$, it can be assumed that there exists no linear relation between two test series x and y, although other non-linear relations may exist. As a linear relation between the costs $c_{i,j}$ and the non-functional characteristics $t_{i,j}$ and $cap_{i,j}$ in the test cases is assumed and also considered in the development of MSS.KOM, the correlation coefficient $r_{x,y}$ is a well-suited measure for the QoS-CoS correlation

of the test cases. The correlation coefficient $r_{x,y}$ with a series of n observations is defined in Equation 8.1.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 * \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{8.1}$$

With the help of $r_{x,y}$ the correlation between $c_{i,j}$ and $t_{i,j}$ and the correlation between $c_{i,j}$ and $cap_{i,j}$ can be calculated. As the correlation between cost and response time is negative and the correlation between cost and execution capacity is positive, the aggregated correlation $r_{QoS,CoS}$ only incorporates the absolute values of $r_{cap,cost}$ and $r_{resp,cost}$. The aggregated mean correlation $r_{QoS,CoS}$ for the QoS-CoS correlation is calculated by:

$$r_{QoS,CoS} = \frac{|r_{cap,cost}| + |r_{resp,cost}|}{2} \tag{8.2}$$

The aggregated QoS-CoS correlation coefficient $r_{QoS,CoS}$, varying by setting x_z, ranges from 0 to 1 and will be further indicated in %.

Besides focusing on the effects of varying QoS-CoS correlations of the test cases on the solution quality and on the computation time, the effects of the restriction strength are analyzed. Concerning the optimization approach depicted in Section 7.3.2, the larger the value W the more restrictive is the execution capacity restriction of constraint 7.4, as more services have to be invoked in parallel in order to serve all workflow execution requests. The smaller the parameter τ the more restrictive is the response time restriction of constraint 7.5. Thus, the restriction strength is further described as the ratio between the constraint restriction (W respectively τ) and the mean value of the considered non-functional parameter of the workflow ($\overline{w}$ respectively $\overline{t}$). The mean value for the workflow's execution capacity $\overline{w}$ is defined as:

$$\overline{w} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m_i} cap_{i,j} \tag{8.3}$$

The mean value for the workflow's response time $\overline{t}$ results to:

$$\overline{t} = \sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i} t_{i,j} \tag{8.4}$$

This leads to a restriction strength for the execution capacity of $W_{res} = \frac{W}{\overline{w}}$ and for the response time to $\tau_{res} = \frac{\tau}{\overline{t}}$. By varying those parameters, the impact of the restriction strength on the heuristics and the exact solution in terms of computation time and solution quality can be evaluated. The larger W_{res} the higher is the execution capacity restriction strength and the smaller τ_{res} the higher is the response time restriction strength.

In order to investigate the impact of the QoS-CoS correlation, the restriction strength, and the problem size on the solution quality and on the computation time, a measure for the solution quality is defined. By solving the resource planning problem with *lp-solve* and the heuristics MSS.KOM and 3S.KOM, several objective values F of the objective functions are achieved. As the optimization follows a minimization of the value of the objective function, the relative solution quality F_{rel} is defined by the ratio between the objective value of the solver F_{lp} and the objective value of the considered heuristic F_h. This measure F_{rel} ranges from 0 to 1 and yields to:

$$F_{rel} = \frac{F_{lp}}{F_h} \tag{8.5}$$

As the solution quality is described in %, the further sections describe the computation times as absolute values for the solution with the solver and the heuristics.

After the definition of the restriction strength, the solution quality, the computation time, and the description of the QoS-CoS correlation, the following sections discuss the evaluation results in detail. The evaluation in Section 8.2 is based on 16 different problem sizes, i.e., a varying number of process steps i and varying number of service candidates m_i whereas the test cases always reveal the same number of service candidates in each category i, i.e., $m_i = m$. The following problem sizes $n|m$ with n as the number of process steps and m as the number of service candidates per process step have been generated: 4|4, 4|6, 4|8, 4|10, 6|4, 6|6, 6|8, 6|10, 8|4, ... , 8|10, 10|4, ... , 10|10. For each problem size, varying test cases concerning the QoS-Cos correlation have been generated by setting the value of x_z. Thus, 10 test sets with 25 test cases each have been generated using the introduced random variables P_{cost}, P_{resp}, and P_{cap}. Starting with $x_z = 0.1$, the value is subsequently incremented by 0.1 up to $x_z = 1$. As the test cases are created with uniform distributed random variables, setting the value of $x_z = 0.1$ does not necessarily imply a QoS-CoS correlation of 10%, instead the QoS-CoS correlation has to be measured with the Pearson product-moment correlation coefficient $r_{QoS,CoS}$ for each created test case. Summing up, 4000 $(16 * 10 * 25)$ test cases have been generated and analyzed within this evaluation. All test cases are solved with the developed heuristics MSS.KOM and the exact solver *lp-solve*.

The evaluation starts with the analysis of the impact of the QoS-CoS correlation. This should help to decide which QoS-CoS correlation has to be set in order to analyze the impact of the restriction strength on the solution quality and on the computation time in order to analyze representative test cases. Section 8.2, 8.3, and 8.4 focus on the analysis of MSS.KOM whereas Section 8.5 compares the two heuristic solutions MSS.KOM and 3S.KOM.

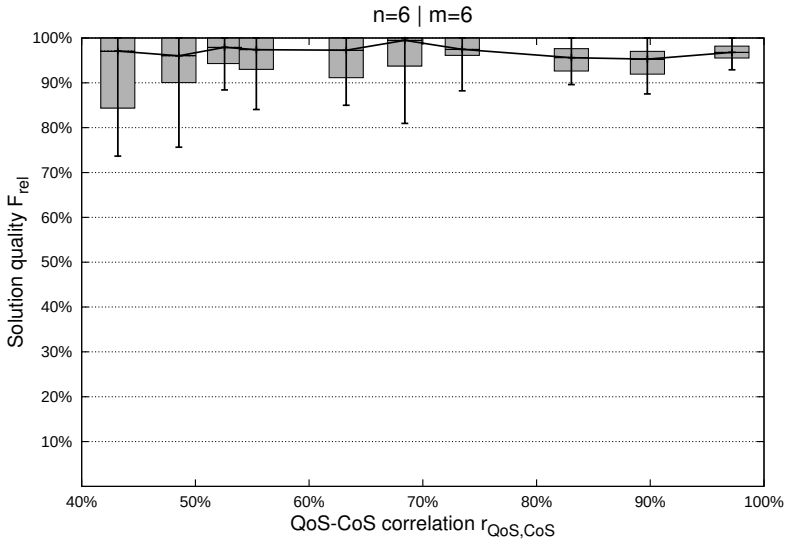


Figure 28: MSS.KOM – QoS-CoS correlation vs. solution quality $n = 6, m = 6$

8.2 IMPACT OF QOS-COS CORRELATION

This section focuses on the analysis of the impact of the QoS-CoS correlation of the test cases on the solution quality and on the computation time. Thus, several test cases with varying QoS-CoS correlations are generated by adjusting the value x_z.

The analysis starts with an investigation of the impact on the solution quality. For this purpose, all test cases have been solved with *lp-solve* and with MSS.KOM. Thus, 25 test cases for each value of x_z (10 values) have been generated for each problem size. As it might not be possible to solve the optimization problem with high restriction strengths, several restriction strengths have been analyzed regarding the tradeoff between restriction strength and the feasibility to solve the optimization problem in a maintainable computation time. Thus, for analyzing the QoS-CoS correlation, the restriction strengths are set to 90% of the mean value of the workflow's response time and to 25% of the mean value of the workflow's execution capacity, i.e., $\tau_{res} = 0.90$ and $W_{res} = 0.25$. Exemplarily for this analysis the problem sizes 6|6, 6|8, 6|10 are depicted in Figure 28, 29, and 30. In these figures, presenting the solution qualities, boxplots are used to describe the dispersion of the achieved solution qualities of the solved test cases. In detail, a boxplot presents the smallest observation (sample minimum), the lower quartile, the median, the upper quartile, and the largest observation (sample maximum).

As can be seen in these figures, although there are some outliers and there is no strict convergence from small correlation to high correlation, in an average consideration the heuristic shows better solution qualities for high QoS-CoS correlations. This effect occurs because in the creation of the measure for the cost-effectiveness $s_{i,j}$ a correlation between QoS properties and the costs was assumed and the measure was developed for this purpose in order to cover

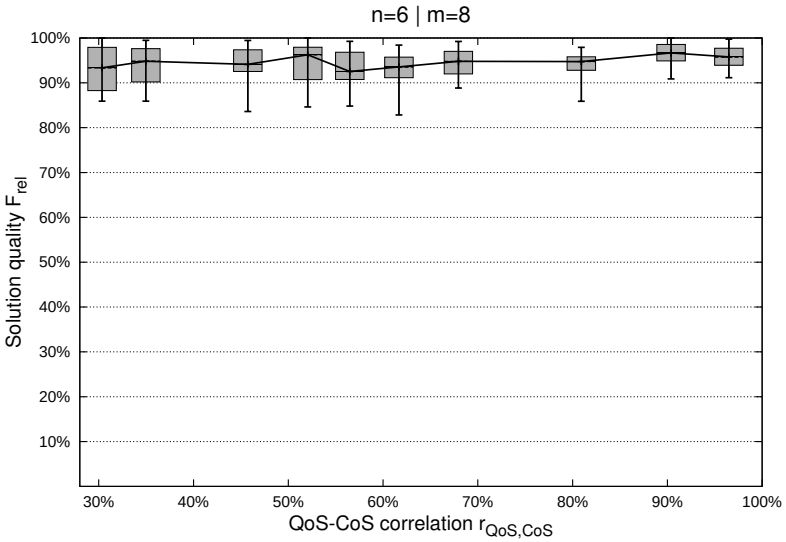


Figure 29: MSS.KOM – QoS-CoS correlation vs. solution quality $n = 6, m = 8$

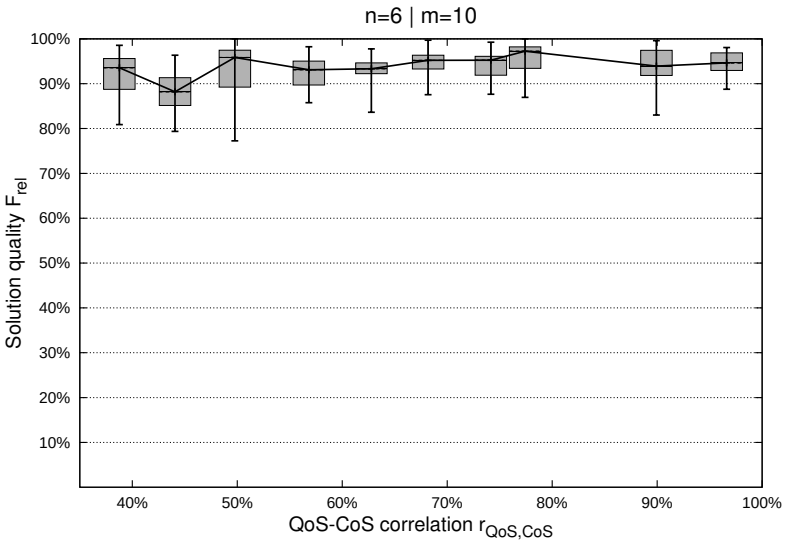


Figure 30: MSS.KOM – QoS-CoS correlation vs. solution quality $n = 6, m = 10$

this type of correlation. Even if the correlation between the QoS parameters and the costs is not high ($r_{QoS,CoS} = 31.34\%$) as shown in Figure 29, the heuristic shows an average solution quality of 93.43%.

The QoS-CoS correlations in Figure 28, 29, and 30 are not always the same as they are based on several generated test cases. Besides the problem size with $n = 6$, selected results of problem sizes with $n = 8$ and $n = 10$ are described in Figure 44 and 45 in the appendix on page 131 and 132. However, Figure 28, 29, and 30 can representatively serve as an example for the observations of the test cases. The main findings of the test cases solved with the heuristic and with *lp-solve* are as follows:

1. Over all test cases a mean solution quality of 95.55% is observed at several QoS-CoS correlation levels and problem sizes. The heuristic achieves

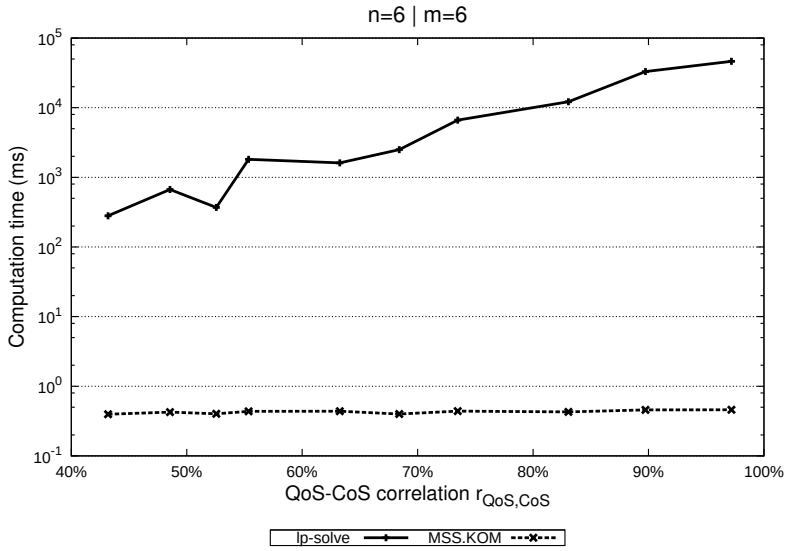


Figure 31: QoS-CoS correlation vs. computation time $n = 6, m = 6$

good solution qualities irrespective of the QoS-CoS correlation which serves as an indicator for the robustness of the heuristic concerning QoS-CoS correlation.

2. As already observed in Figure 28, 29, and 30, higher QoS-CoS correlation results in slightly better solution quality. Obviously, the higher the QoS-CoS correlation the smaller is the difference between the computed values $s_{i,j}$ and $cl_{i,j}$ in the $global\_list[]$ and the $cat\_list_i[]$. The service candidates are more equal concerning their cost-effectiveness measures $s_{i,j}$ and $cl_{i,j}$, although they differ in their absolute values. As a result, the objective values of the invocation plans do not disperse widely and result in better solution qualities.

After focusing on the impact on the solution quality, the impact on the computation time is analyzed in the following. The absolute observed mean computation times for the exact approach and the heuristic MSS.KOM are depicted for $n = 6$ workflow steps in Figure 31, 32, and 33.

Each measured point in the figures represents the mean value of 25 test cases. As it can be observed, the absolute computation times for the exact approach are very high and are not suitable for time-critical resource planning requirements. Concerning small problems sizes ($n = 6$, $m = 6$), the exact computation times of *lp-solve* range from 283 ms to 47,372 ms, whereas in case of $m = 10$ the computation times range from 38,487 ms to 32,162,482 ms. This indicates the high influence of the QoS-CoS correlation of the test cases on the computation time of the exact solution. Concerning test cases with low QoS-CoS correlation, the Branch & Bound method used by *lp-solve* can delimit the solution space faster than in case of a high QoS-CoS correlation where the test cases are very homogeneous. As it can already be seen at a mid-sized problem as depicted in Figure 32, i.e., a service-based workflow with 6 workflow steps

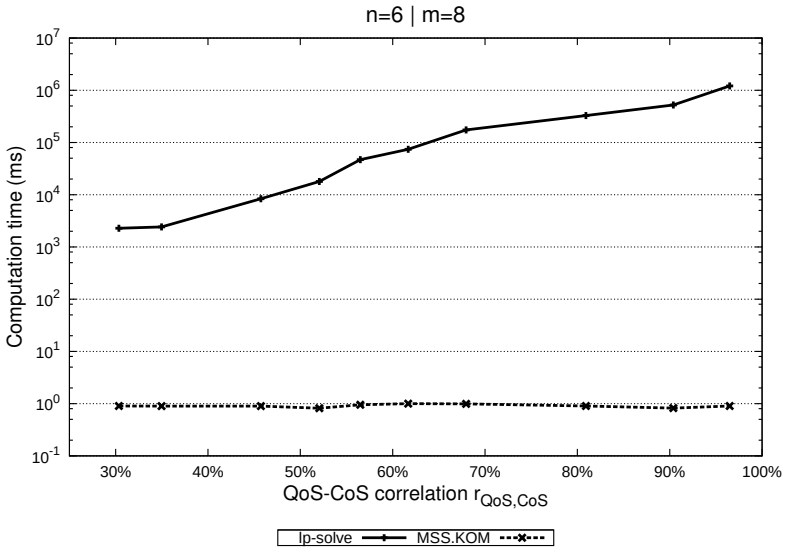


Figure 32: QoS-CoS correlation vs. computation time $n = 6, m = 8$

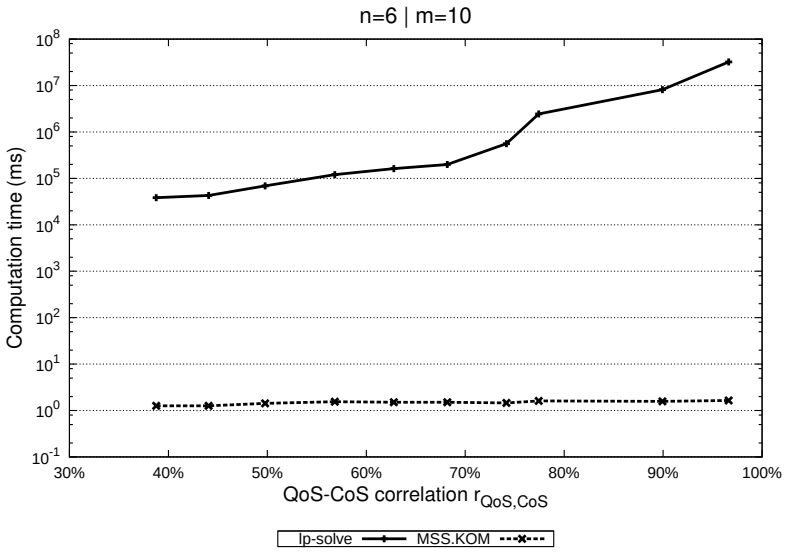


Figure 33: QoS-CoS correlation vs. computation time $n = 6, m = 10$

and 8 alternative services per category, an exact solution cannot be achieved in less than 47 seconds at an average QoS-CoS correlation level of 57%. At a high QoS-CoS correlation level, the exact solution needs up to 20 minutes for solving the resource planning problem. In case of the problem size presented in Figure 33, the calculation of the exact solution needs several hours. This indicates the high impact of the problem size and the QoS-CoS correlation on the computation time, resulting in an exponential increase in computation time for the exact solution.

In contrast to the exact solution, the developed heuristic MSS.KOM shows significant outperformance in terms of computation time. As shown in Figure 31, the computation time is 0.42 ms on average, representing 0.14% of the

shortest time needed for the exact solution at the smallest QoS-CoS correlation level. Concerning a larger problem size, Figure 33 shows computation times for the heuristic of about 1.61 ms on average. This complies with 0.038% of the computation time needed by the exact solution. In each of the analyzed scenarios with varying problem sizes, the outperformance of the heuristic in contrast to the exact solution can be observed. Furthermore, selected results with $n = 8$ and $n = 10$ are described in Figure 46 and 47 in the appendix on page 133 and 134. The mean observed relative computation time of the heuristic in this evaluation is 0.13%, vastly decreasing with increasing QoS-CoS correlation. Furthermore, the computation time of the heuristic is invariant of the QoS-CoS correlation. This finding affirms that the heuristic is applicable to a large variety of test cases independent on the QoS-CoS correlation of the test cases.

Due to the large increase in computation time of the exact solution with increasing QoS-CoS correlation, a QoS-CoS correlation level of about 55% is used for the further analysis in order to represent a realistic trade-off between the size of test samples and computation times of the solution approaches.

8.3 IMPACT OF RESTRICTION STRENGTH

After the evaluation of the impact of the QoS-CoS correlation of the test cases on the solution quality and on the computation time, this section focuses on the restriction strength of the side conditions of the optimization problem. Obviously, varying restriction strengths affect the selection process of the services. An increasing execution capacity restriction strength W_{res} results in a higher workload, i.e., more workflow execution requests. This yields to a larger number of parallel service invocations in order to meet workload requirements. Focusing on the response time restriction strength τ_{res}, an increase in this restriction strength represents a decrease in the value τ_{res} resulting in a smaller overall response time of the service-based workflow. If both restriction strengths increase, i.e., increasing value of W_{res} and decreasing value of τ_{res} it may be possible that no solution of the optimization problem exists. In order to analyze this behavior, the following deliberations observe the impact of the restriction strength on the solution quality and on the computation time.

In the findings of section 8.2 the QoS-CoS correlations of the test cases are set to about 55%, representing a realistic trade-off between test samples and computation time. In order to present the results of the impact of the restriction strength on computation time and solution quality, test samples with 8 workflow steps and 8 alternative services per workflow step are chosen.

In a first evaluation the impact of the execution capacity restriction strength is analyzed by generating test cases with varying restriction strengths for the execution capacity W_{res} while keeping the response time restriction strength $\tau_{res} = 100\%$ as presented in Figure 34 and 35. In this consideration, 25 test cases for each restriction strengths W_{res} varying in steps of 10% from 10% to

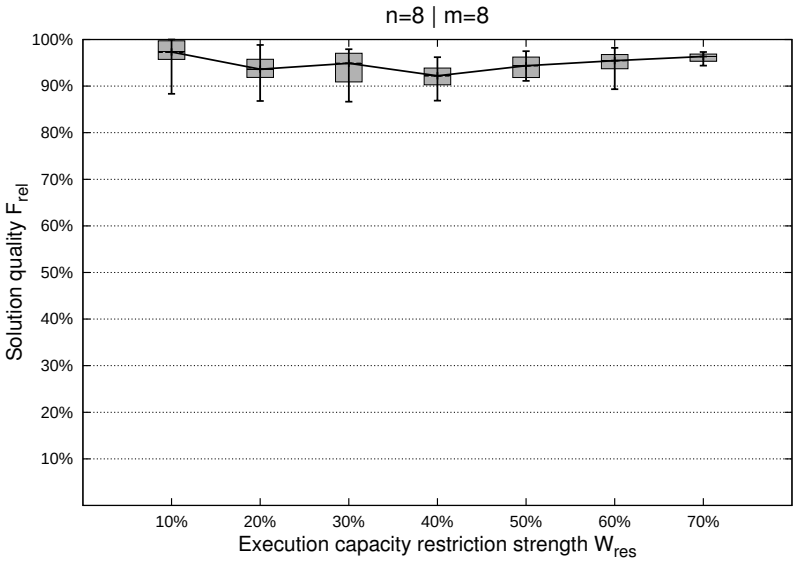


Figure 34: MSS.KOM – Execution capacity restriction strength W_{res} vs. solution quality $n = 8, m = 8$

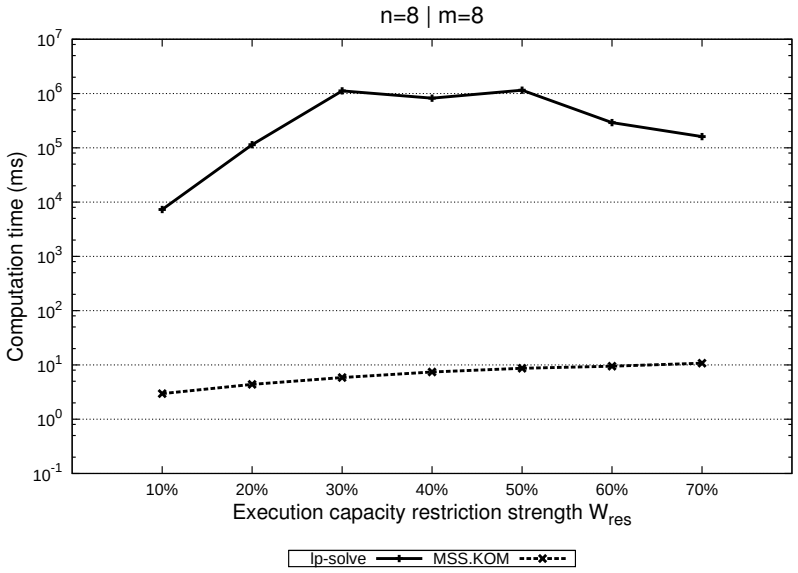


Figure 35: Execution capacity restriction strength W_{res} vs. computation time $n = 8, m = 8$

100% are analyzed.

The distribution of the solution quality F_{rel} for the 8|8 problem in Figure 34 presents a mean value of 93.5%. Concerning the impact of the restriction strength W_{res} on the solution quality, on average, the solution quality is relatively invariant to the restriction strength of the execution capacity. Focusing on the impact of the restriction strength W_{res} on the computation time of both, the heuristic and the exact solution, it can be observed that on average the heuristic took about 6.23 ms, whereas the exact approach took about 10 minutes. This leads to an average relative computation time of 0.06% slightly

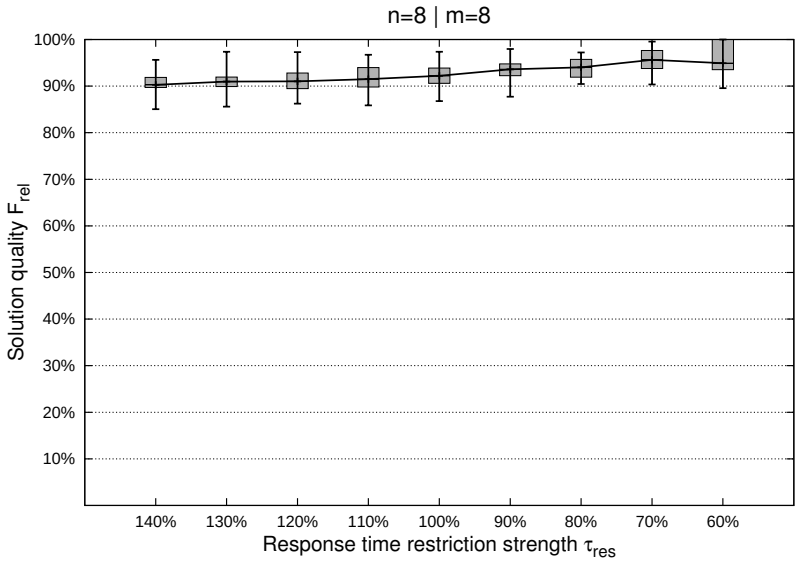


Figure 36: MSS.KOM – Response time restriction strength τ_{res} vs. solution quality $n = 8, m = 8$

increasing with the restriction strength. At low restriction strengths, few services are necessary in order to determine a feasible solution. Hence, the Branch & Bound method determines feasible solutions very early, resulting in a relatively small computation time. High restriction strength leads to a small solution space and many branches can be discarded due to constraint violations. Between these two restrictions strength the Branch & Bound method needs a high computation time as presented in Figure 35. In some cases it is not possible to determine a feasible solution due to the distribution of test cases as in case of restriction strengths above 70%. In these cases, the restriction strengths of both, the response time and the execution capacity are too high. No set of services in the test cases fulfills the requirements on response time and execution capacity in the side conditions.

The results of the analysis of the impact of the response time restriction strength on the computation time and on the solution quality is presented in Figure 36 and 37. For test case generation, the capacity restriction strength W_{res} is set to 40% and 25 test cases for the response time restriction strengths varying in steps of 10% are generated. The smaller the value of τ_{res} in %, the larger is the restriction strength. As it can be seen in Figure 36, the heuristic has a very high solution quality with a mean value of 92.71% slightly increasing with increasing restriction strength. As Figure 37 presents the computation time of the heuristic and the exact solution, it can be observed that the heuristic has a very small computation time of 4.11 ms on average. The decrease in computation time of the exact solution for a high response time restriction strength occurs because the Branch & Bound method can be executed faster, which means that the branching performs better. Again for higher restriction strengths ($\tau_{res} < 60\%$) no feasible solution exists. For some problem sizes it is not possible to determine all possible solutions because of computation

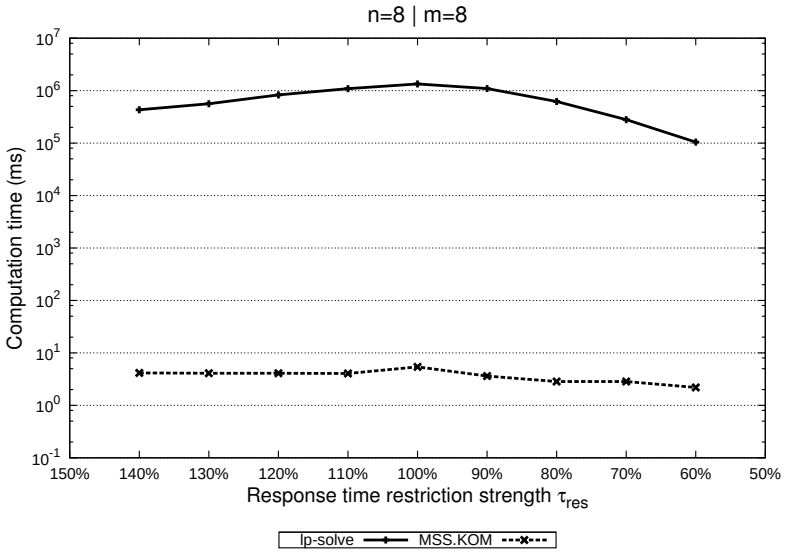


Figure 37: Response time restriction strength τ_{res} vs. computation time $n = 8, m = 8$

times beyond 18 hours whereas MSS.KOM solves the problem within 17 ms on average.

8.4 IMPACT OF PROBLEM SIZE

Focusing on the impact of the problem size on the solution quality and on the computation time, Figure 38 and 39 present significant results. Besides the analysis of problem sizes up to 10|10, also larger problems, i.e., 7|15 and 15|7, are generated and solved with the heuristic and with the exact solver *lp-solve*. Due to the vastly increase in computation time for large-sized problems, these two additional problems are generated in order to serve as a basis for the analysis of large-sized problems.

For the analysis in this section, $18 * 25 = 450$ test cases are generated with a QoS-CoS correlation mean value of about 55% (x_z ranges from 0.5 to 0.8 dependent on the test cases). Furthermore, the findings of the previous section are used in order to determine an appropriate restriction strength. In this section the response time restriction strength is set to 100%, i.e., the response time restriction τ is set to $\overline{t}$ and the execution capacity restriction strength is set to 30%, i.e., $W = 0.3 * \overline{W}$.

Focusing on the solution quality as presented in Figure 38, the evaluation presents an average solution quality for the heuristic MSS.KOM of about 95%. Furthermore, the solution quality is relatively independent of the problem size. As can be seen in Figure 39, the exact solution reveals an exponential increase in computation time with increasing problem sizes. In contrast, the computation time of the developed heuristic increases very small with increasing problem size. Concerning large problem sizes, e.g., a problem with 7 workflow steps and 15 service candidates per category, the computation time of the

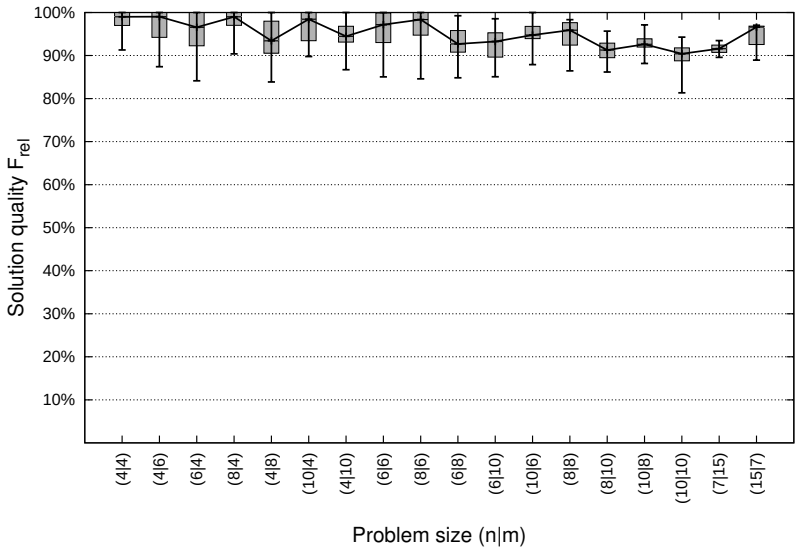


Figure 38: MSS.KOM – Problem size (n | m) vs. solution quality

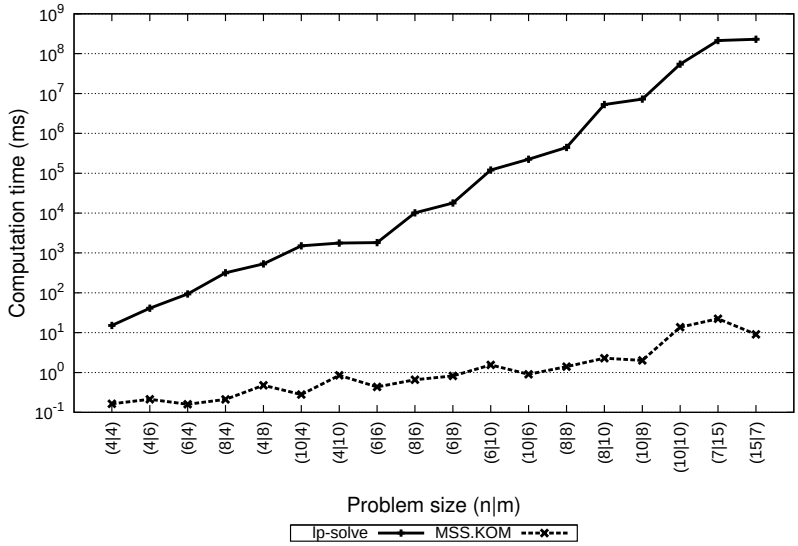


Figure 39: Problem size (n | m) vs. computation time

solver has been about 59 hours on average. This large computation time confirms the presumption that it is necessary to use efficient heuristics for solving the resource planning problem. Concerning this large problem size the heuristic performs very fast, i.e., it solves the optimization problem within 22 ms. Although this computation time is too large for time-critical requirements, it is a good result in contrast to the exact solution. Concerning small problem sizes, the evaluations presented in Figure 39 confirm computation times of a few milliseconds up to 1-2 seconds.

Analyzing the impact of the number of workflow steps and the number of alternative service candidates per category in detail, it can be stated that the

Initial Prob.	Trans.	Reduced	Reduction Ratio
4|4	4|16	4|3	10^{-3}
4|6	4|64	4|3	10^{-6}
4|8	4|256	4|3	10^{-8}
4|10	4|1024	4|4	10^{-10}
6|4	6|16	6|2	10^{-6}
6|6	6|64	6|3	10^{-8}
6|8	6|256	6|4	10^{-11}
6|10	6|1024	6|5	10^{-14}
8|4	8|16	8|3	10^{-6}
8|6	8|64	8|3	10^{-11}
8|8	8|256	8|4	10^{-15}
8|10	8|1024	8|5	10^{-19}
10|4	10|16	10|2	10^{-10}
10|6	10|64	10|4	10^{-13}
10|8	10|256	10|5	10^{-18}
10|10	10|1024	10|5	10^{-24}

Table 5: 3S.KOM - Reduction of solution space

influence of these two parameters have the same impacts on the solution quality and on the computation time. An increasing number of workflow steps n and an increasing number of service candidates m leads to a slight decrease in solution quality and also to an increase in computation time.

After the detailed evaluation of the developed heuristic MSS.KOM, the next section analyzes the developed single service selection heuristic 3S.KOM in comparison to MSS.KOM and presents selected results.

8.5 COMPARATIVE ANALYSIS

In this section, $13 * 25 = 325$ test cases with a QoS-CoS correlation mean value of about 55% are generated. The response time restriction strength τ is set to 100% and the execution capacity restriction strength W_{res} to 30%.

Before presenting the detailed comparison of the developed heuristics, the search space reduction of 3S.KOM is analyzed. Table 5 gives an overview on the degree of the search space reduction. The first column presents the analyzed initial problem size of the resource planning problem whereas the second column presents the problem size after the transformation and shows the drastic increase in alternative services per category. Column 3 shows the reduced number of services per category after the adoption of the described

Problem Size	4|4	4|6	6|4	8|4	4|8	6|6	4|10
MSS.KOM	0.4977	0.4259	0.1058	0.0358	0.0485	0.0271	0.0469
3S.KOM	8.5103	2.9622	1.4687	0.2648	0.5679	0.1121	0.5100

Table 6: Heuristics – Relative computation times in %, 1/2

Problem Size	10|4	8|6	6|8	6|10	10|6	8|8
MSS.KOM	0.0048	0.0016	0.0020	0.0009	0.0002	0.0001
3S.KOM	0.0310	0.0060	0.0086	0.0079	0.0003	0.0002

Table 7: Heuristics – Relative computation times in %, 2/2

algorithm and the search space reduction ratio is depicted in column 4.

As described in Section 7.5, the objective of the heuristic 3S.KOM is the reduction of the solution space, following a two-step approach. During the creation of the power set of the services in each category and after the aggregated services are created, the solution space is reduced. Without a reduction of a solution space, $(2^m)^n$ possible service combinations would exist. Out of those combinations, the optimization algorithm would determine the best invocation plan meeting all constraints of the optimization model, resulting in very high computation times [BSR+06b].

Assuming a problem size $n|m$, the proposed algorithm of 3S.KOM reduces the resulting aggregated services per category to less than m. Concerning a problem size 8|8, the power set in each category results to $2^8 = 256$ services. Thus, the new created problem is a 8|256 problem where it is allowed to select one aggregated service per category with a possible number of service combinations of 256^8. Even with high-performance heuristics, this solution space is too large in order to handle this problem in short computation times. Thus, 3S.KOM transforms the 8|256 problem into a 8|4 problem where only one aggregated service per category has to be chosen. This yields to a very high reduction of the solution space as presented in Table 5, influenced by the workflow length and the number of services per category. It can be observed, that dependent on the problem size, the reduction of the search space has a ration of 10^{-3} to 10^{-24}, computed as the ratio between the number of possible service compositions. This occurs because the number of workflow steps and the number of services per category both influence the initial problem size exponentially. Furthermore, the lower the QoS-CoS correlation, the more services can be deleted in the solution space.

After the description of the search space reduction, the following presents results in a comparative observation of the developed heuristics. Evaluations show, that the developed heuristic 3S.KOM is very fast also in comparison to MSS.KOM as presented in Table 6 and 7. At small problem sizes, 3S.KOM needs only 8% of the time of the exact solution. With increasing problem

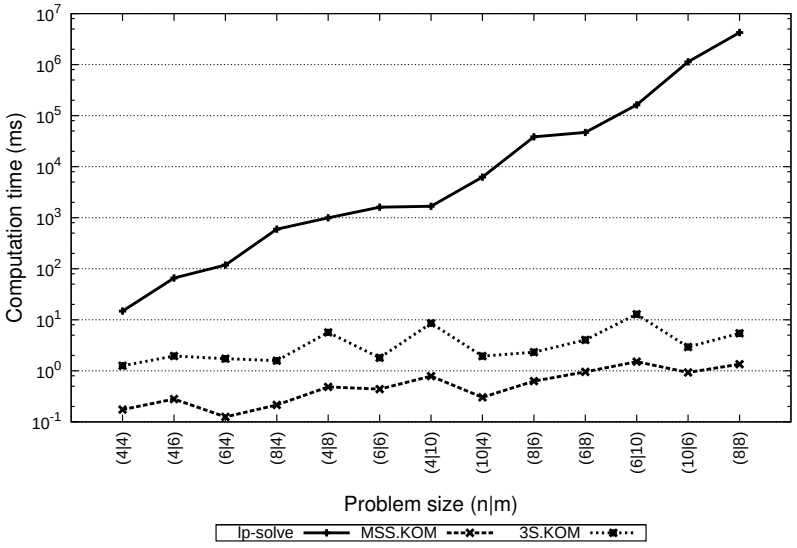


Figure 40: Heuristics – Problem size (n | m) vs. computation time

size the computation time of the heuristic 3S.KOM further decreases to a relative computation time of less than 0.0002% of the exact solution. In contrast, MSS.KOM shows relative computation times of 0.49% to less than 0.0001%. This highlights the extreme performance of both heuristics.

The robustness of both heuristics in terms of computation time is further presented in Figure 40, presenting the absolute values for the computation time of both heuristics and the exact solution. Since the computation time of the exact solution vastly increases with the problem size, the computation times of the heuristics increase slightly. In detail, the exact solution needs computation times from 14 ms up to 4,248,259 ms depending on the problem size. Obviously, these computation times cannot be used for time-critical computation of an exact solution for the analyzed resource planning problem. Comparing the relative and the absolute computation times of 3S.KOM and MSS.KOM, the latter performs better than 3S.KOM and is well suited for solving the resource planning problem offering short computation times.

Focusing on the solution quality an inverse observation can be noticed, as the solution quality of 3S.KOM, depicted in Figure 41, is always higher than in case of MSS.KOM. Thus, 3S.KOM has a higher solution quality but is slower in terms of computation time. Depending on the application scenario, the computation time and solution quality preferences, the intermediary has to select between those heuristic solutions. As both solutions have a high solution quality independent of the problem size, MSS.KOM reveals always shorter computation times. Thus, for the considered resource planning problem, the heuristic solution MSS.KOM is well suited, solving the problem at high solution qualities and short computation times.

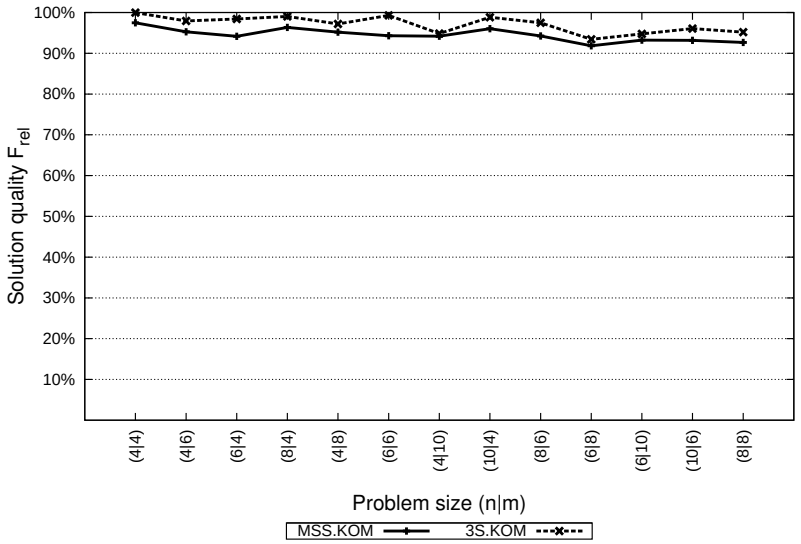


Figure 41: Heuristics – Problem size (n | m) vs. solution quality

8.6 SUMMARY

This chapter provides a detailed evaluation of the developed heuristic solutions and presents selected results. For the heuristic solution MSS.KOM, the impact of influencing factors as the QoS-CoS correlation, the restrictions strength, and the problem size on the solution quality as well as on the computation time are analyzed. Furthermore, the developed heuristic solutions MSS.KOM and 3S.KOM are compared in a detailed analysis. For the determination of the computation time and the solution quality the absolute values of the computation time and the values of the objective functions are compared to an exact solution of the optimization problem, solved with the mixed integer linear programming solver *lp-solve*[2].

Independent of the analyzed influencing factors, the developed heuristic solution MSS.KOM has a vast advantage in computation time in contrast to the exact solution. The relative computation time decreases vastly with increasing problem size and increasing QoS-CoS correlation of the test cases. Even at a large problem size with $n = 10$ and $m = 10$, the heuristic solves the analyzed resource planning problem at a solution quality above 92% about a factor 10^7 faster than the exact solution and at least about a factor 3 faster than 3S.KOM. The overall average solution quality over all test cases of MSS.KOM has been about 95% and of 3S.KOM about 97%. Both heuristic solutions reveal computations times of a few milliseconds whereas MSS.KOM performs faster than 3S.KOM. As both heuristic solutions achieve high solution qualities, the developed high-performance heuristic MSS.KOM is the preferred heuristic in terms of computation time for the resource planning problem discussed in this chapter.

2 http://lpsolve.sourceforge.net/5.5/

9 CONCLUSION AND OUTLOOK

This chapter concludes the main findings of this thesis and gives an outlook on future research challenges in the field of cross-organizational service-based workflows.

9.1 CONCLUSION

The application of the SOA paradigm on workflows introduces major challenges in recent years, especially in the field of workflow composition ensuring specific QoS and cost requirements of the users. Workflows can be decomposed into basic activities realized by reusable services offering specific business functionalities.

Concerning cross-organizational service-based workflows, services can be sourced from internal as well as from external providers. The success of the composed workflow is significantly dependent on the QoS in a scenario in which an intermediary aggregates workflow execution requests and composes workflows for his customers. Thus, ensuring QoS is mandatory in order to stay competitive. Efficient solution strategies for the QoS optimization of service-based workflows are necessary in order to provide a high level of QoS to the workflow consumers. Especially, considering specific QoS parameters such as response time, execution capacity, and several pricing models, the creation of invocation plans, meeting customer requirements, is of high importance. For workflows with a high repetition rate, inducing a large workload, multiple parallel service invocations may be necessary in order to serve all incoming workflow execution requests. The main objective of the intermediary is the creation of cost-efficient invocation plans that meet customer requirements. Furthermore, a resource planning is necessary for the determination of the number of services that have to be invoked in order to be able to serve all incoming workflow execution requests.

Thus, this thesis provides several contributions to the QoS optimization of service-based workflows. The main contribution is the development and the evaluation of efficient resource planning heuristics, facilitating the fast computation of cost-efficient invocation plans out of services with limited execution capacities and a specific pricing model. Furthermore, other constraints such as execution deadlines are taken into account and the problem is formulated as an optimization problem that is proven to be NP-hard. Thus, two efficient heuristic solutions are developed and implemented. For the evaluation, the optimization model is solved by mathematical optimization with an exact solution as well as by the application of the developed heuristics. The results show heuristic solutions achieving a solution quality of 95% (MSS.KOM) on aver-

age, revealing computation times significantly better than the exact solution. The relative computation times of the high-performance heuristic MSS.KOM are 0.49% to less than 0.0001% in comparison to the exact solution. Furthermore, the computation time of MSS.KOM is at least about a factor 3 faster than in case of 3S.KOM. Hence, the developed heuristic solution MSS.KOM is especially useful in time-critical applications in which the offering of services changes dynamically and the determination of cost-efficient invocations plans, meeting constraint requirements, have to be computed fast.

Further contributions address several challenges in the field of QoS optimization of service-based workflows. As a foundation, a classification of pricing models for services is developed, classifying these in three high level types of pricing models: variable fee, fixed fee, and hybrid fee pricing models. Furthermore, the impact on the service selection process of services for each of these models is discussed and analyzed in detail.

The developed QoS optimization solutions support the intermediary in the worst- and average-case performance analysis of service-based workflows. In the average-case performance analysis, key findings of queuing theory are adapted to the concept of service-based workflows. Furthermore, an optimization approach is developed which supports the intermediary in the optimization of the service utilization by providing a solution with low service invocation costs. Furthermore, constraints as the avoidance of a large increase of response time are taken into account. Concerning the worst-case performance analysis, major insights of network calculus are identified and adapted to the concept of service-based workflows. Consequently, optimization approaches, facilitating the optimization of QoS parameters such as the delay or the throughput in the worst-case, are developed. Finally, the architectural extension *WPX.KOM* for generic QoS management systems for service-based workflows, facilitating the realization and implementation of the developed solution strategies for the resource planning of service-based workflows, is proposed.

9.2 OUTLOOK

The presented QoS optimization approaches for service-based workflows are one aspect of the QoS management of service-based workflows. Thus, this section highlights several research challenges concerning service-based workflows in the future. Figure 1 in Section 1.1 gives an overview on related research topics in the field of service-based workflows whereas the presented thesis mainly focuses on the service composition and resource planning of service-based workflows.

One major research challenge in the field of service-based workflows is the monitoring of services in order to avoid SLA violations. Monitoring mechanisms have to detect SLA violations before they occur and have to propose countermeasures ensuring a reliable workflow execution. Besides centralized

monitoring mechanisms, decentralized monitoring mechanisms realize good monitoring results in terms of violation detection at decreasing monitoring costs [RES+08], [Rep09]. Furthermore, these approaches have to be adapted to large-scale systems with several stakeholders such as service providers and service consumers and extended to automated negotiation mechanisms for SLAs as a reaction of detected SLA violations.

As the Internet of Services comprises various roles and responsibilities, effective governance approaches become more and more important. Regarding the increasing number of services and the increasing system complexity, SOA Governance is an important issue in the future [NERS08]. A dedicated governance should define policies and recommendations for the management of requirements and for decisions such as buying a service on the market or developing a service. Furthermore, methods for the assessment of SOA maturity as well as governance policies have to be designed in the future.

Cross-organizational workflows induce a large set of security issues. Because of a large variety of service providers and service consumers and the changing offering of services over time, a large set of business relationships occur beyond enterprise boundaries. Large and complex distributed systems with different owners and services that have a high interoperability and are loosely coupled represent major challenges in the field of SOA security [MGK+09]. Thus, special purpose security mechanisms addressing these challenges have to be designed in order to support, e.g., intrusion detection and the assessment of SOA security risks.

Future work in the field of QoS optimization for cross-organizational service-based workflows is the adaptation of the developed solution strategies to other pricing models such as pricing models with dynamic schemes. Furthermore, the prediction of the incoming workflow execution requests has to be analyzed further in order to provide appropriate input parameters for the described optimization approaches.

[AC01] Jörn Altmann and Karyen Chu. How to Charge for Network Services – Flat-Rate or Usage-Based? *Computer Networks*, 36(5):519–531, 2001.

[ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer-Verlag, 2004.

[ACMS08] Vikas Aggarwal, Girish Chafle, Sumit Mittal, and Biplav Srivastava. Understanding Approaches for Web Service Composition and Execution. In *1st Bangalore Annual Compute Conference*, pages 1–8, 2008.

[Ark02] Assaf Arkin. Business Process Modeling Language, Version 1.0. November 2002. `http://www.bpmi.org/downloads/BPML1.0.zip`.

[ARK+06] Md. Mostofa Akbar, M. Sohel Rahman, Mohammad Kaykobad, Eric George Manning, and Gholamali C. Shoja. Solving the Multi-dimensional Multiple-choice Knapsack Problem by constructing convex hulls. *Computers and Operations Research*, 33(5):1259–1273, 2006.

[Art99] W. Brian Arthur. Increasing Returns and the New World of Business. *Harvard Business Review*, 4:147–170, 1999.

[AVMM04] Rohit Aggarwal, Kunal Verma, John Miller, and William Milnor. Constraint Driven Web Service Composition in METEOR-S. In *IEEE International Conference on Services Computing*, pages 23–30, 2004.

[Baloo] Simonetta Balsamo. Product Form Queueing Networks. In *Performance Evaluation: Origins and Directions*, pages 377–401, 2000.

[BBK08] Günter Bamberg, Franz Baur, and Michael Krapp. *Statistik*. Oldenbourg, 14th edition, 2008.

[BCOQ92] Francois Baccelli, Guy Cohen, Geert-Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons, 1992.

[Ber08] Rainer Berbner. *Dienstgüteunterstützung für Service-orientierte Workflows*. Books on Demand, 2008.

[BGdMT06] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Shridharbhai Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley & Sons, 2nd edition, 2006.

[BGR+07] Rainer Berbner, Tobias Grollius, Nicoals Repp, Julian Eckert, Oliver Heckmann, Erich Ortner, and Ralf Steinmetz. Management of Service-oriented Architecture (SOA)-based Application Systems. *Enterprise Modelling and Information Systems Architectures*, 1(2):14–26, 2007.

[BL06] Martin Bichler and Kwei-Jay Lin. Service-Oriented Computing. *IEEE Computer*, 39(3):99–101, 2006.

[BL07] David Booth and Canyang Kevin Liu. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C Recommendation. W3C Web Services Description Working Group, June 2007. http://www.w3.org/TR/wsdl20-primer/.

[BSR+06b] Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, and Ralf Steinmetz. Heuristics for QoS-aware Web Service Composition. In *4th IEEE International Conference on Web Services*, pages 72–79, 2006.

[BT01] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus*. Lecture Notes in Computer Science (LNCS). Springer-Verlag, 2001.

[Bun08] Bundesamt für Sicherheit in der Informationstechnik. *SOA-Security-Kompendium: Sicherheit in Service-orientierten Architekturen*. Bundesamt für Sicherheit in der Informationstechnik, 2008. https://www.bsi.bund.de/cln_136/ContentBSI/literat/studien/soa/index_htm.html.

[Buz73] Jeffrey Peter Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16(9):527–531, 1973.

[Car02] Jorge Cardoso. *Quality of Service and Semantic Composition of Workflows*. PhD thesis, Department of Computer Science, University of Georgia, 2002.

[CCDS03] Malu Castellanos, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Intelligent Management of SLAs for Composite Web Services. In *3rd International Workshop on Databases in Networked Information Systems*, pages 158–171, 2003.

[CHL+07] Roberto Chinnici, Hugo Haas, Amelia A. Lewis, Jean-Jacques Moreau, David Orchard, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts*. W3C Recommendation. W3C Web Services Description Working Group, June 2007. http://www.w3.org/TR/wsdl20-adjuncts/.

[CHT03a] Kishore Channabasavaiah, Kerrie Holley, and Edward Tuggle. Migrating to a service-oriented architecture, part 1. 2003. http://www.ibm.com/developerworks/library/ws-migratesoa/.

[CHT03b] Kishore Channabasavaiah, Kerrie Holley, and Edward Tuggle. Migrating to a service-oriented architecture, part 2. 2003. http://www.ibm.com/developerworks/library/ws-migratesoa2/.

[CHvRR04] Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers. UDDI Version 3.0.2. In *OASIS UDDI Spec Technical Committee*, 2004. http://www.uddi.org/pubs/uddi_v3.htm.

[CIJ+00] Fabio Casati, Ski Ilnicki, LiJie Jin, Vasudev Krishnamoorthy, and Ming-Chien Shan. Adaptive and Dynamic Service Composition in eFlow. Technical Report HPL-200039, Technology Laboratory, Palo Alto, CA, March 2000. http://www.hpl.hp.com/techreports/2000/HPL-2000-39.pdf.

[CMRW07] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Recommendation. W3C Web Services Description Working Group, June 2007. http://www.w3.org/TR/wsdl20/.

[CPEV05a] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Genetic and Evolutionary Computation Conference*, pages 1069 –1075, 2005.

[CPEV05b] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. QoS-Aware Replanning of Composite Web Services. In *IEEE International Conference on Web Services*, pages 121–129, 2005.

[CPEV08] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. A framework for QoS-aware binding and re-binding of composite Web services. *Journal of Systems and Software*, 81(10):1754–1769, 2008.

[Cru91a] Rene Cruz. A Calculus for Network Delay. I. Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[Cru91b] Rene Cruz. A Calculus for Network Delay. II. Network Analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[CSMA02] Jorge Cardoso, Amit Sheth, John Miller, and Jonathan Arnold. Modeling Quality of Service for Workflows and Web Service Processes. *Web Semantics Journal*, 1(3):281–308, 2002.

[DD98] Wolfgang Domschke and Andreas Drexl. *Einführung in Operations Research*. Springer-Verlag, 1998.

[DJMZ08] Wolfgang Dostal, Mario Jeckle, Ingo Melzer, and Barbara Zengler. *Service-orientierte Architekturen mit Web Services*. Elsevier, 3rd edition, 2008.

[DP02] Dan Davis and Manish Parashar. Latency Performance of SOAP Implementations. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 407–412, 2002.

[DS05] Wolfgang Domschke and Armin Scholl. *Grundlagen der Betriebswirtschaftslehre*. Springer-Verlag, 3rd edition, 2005.

[EBRS09] Julian Eckert, Marc Bachhuber, Nicolas Repp, and Ralf Steinmetz. The Implementation of Service-oriented Architectures in the German Banking Industry - A Case Study. In *15th Americas Conference on Information Systems*, 2009.

[Eck08] Julian Eckert. Resource Planning for Distributed Service-oriented Workflows. In *3rd International Conference on Software and Data Technologies*, pages 38–45, 2008.

[EEM+08] Julian Eckert, Deniz Ertogrul, André Miede, Nicoals Repp, and Ralf Steinmetz. Resource Planning Heuristics for Service-oriented Workflows. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 591–597, 2008.

[EEP+09] Julian Eckert, Deniz Ertogrul, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. The Impact of Service Pricing Models on Service Selection. In *4th International Conference on Internet and Web Applications and Services*, pages 316–321, 2009.

[EPR+07] Julian Eckert, Krishna Pandit, Nicolas Repp, Rainer Berbner, and Ralf Steinmetz. Worst-Case Performance Analysis of Web Service Workflows. In *9th International Conference on Information Integration and Web-based Application & Services*, pages 67–77, 2007.

[ERM09] Julian Eckert, Nicolas Repp, and Wolfgang Martin. *SOA Check 2009: Status Quo und Trends im Vergleich zum SOA Check 2008 und 2007*. IT-Verlag, 2009.

[ERN+08] Julian Eckert, Nicolas Repp, Michael Niemann, Marc Billeb, Achim Schäfer, and Ralf Steinmetz. IT Organization as a Limiting Factor for the Success of Service-oriented Architectures. *EFL Quarterly*, 2:4–5, 2008.

[ERS+09] Julian Eckert, Nicolas Repp, Dieter Schuller, Lars Kiewning, and Ralf Steinmetz. Implications of Service-oriented Architectures in the German Banking Industry - A Case Study. *EFL Quarterly*, 2:4–5, 2009.

[ESN+08] Julian Eckert, Stefan Schulte, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Worst-Case Workflow Performance Optimization. In *3rd International Conference on Internet and Web Applications and Services*, pages 632–637, 2008.

[ESR+08] Julian Eckert, Stefan Schulte, Nicolas Repp, Rainer Berbner, and Ralf Steinmetz. Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms. In *IEEE International Conference on Digital Ecosystems and Technologies*, pages 313–318, 2008.

[FR05a] Markus Fidler and Stephan Recker. A Dual Approach to Network Calculus Applying the Legendre Transform. In *3rd International Workshop on Quality of Service in Multiservice IP Networks*, 2005.

[FR05b] Markus Fidler and Stephan Recker. Transformation-based Network Calculus Applying Convex/Concave Conjugates. In *14. Fachtagung Kommunikation in Verteilten Systemen*, 2005.

[FR06] Markus Fidler and Stephan Recker. Conjugate Network Calculus: A Dual Approach Applying the Legendre Transform. *Computer Networks*, 50(8):1026–1039, 2006.

[GHM+07a] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C Recommendation. W3C XML Protocol Working Group, April 2007. `http://www.w3.org/TR/soap12-part1/`.

[GHM+07b] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. W3C Recommendation. W3C XML Protocol Working Group, April 2007. `http://www.w3.org/TR/soap12-part2/`.

[GKG03] Dimitris Gouscos, Manolis Kalikakis, and Panagiotis Georgiadis. An Approach to Modeling Web Service QoS and Provision Price. In *4th International Conference on Web Information Systems Engineering*, pages 121–130, 2003.

[GSW97] Alok Gupta, Dale O. Stahl, and Andrew B. Whinston. A Stochastic Equilibrium Model of Internet Pricing. *Journal of Economic Dynamics and Control*, 21:697–722, 1997.

[GWW02] Michael Gillmann, Gerhard Weikum, and Wolfgang Wonner. Workflow Management with Service Quality Guarantees. In *ACM SIGMOD International Conference on Management of Data*, pages 228–239, 2002.

[Hav98] Boudewijn R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach*. John Wiley & Sons, 1998.

[Hec06a] Oliver Heckmann. *The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design*. John Wiley & Sons, 2006.

[Hec06b] Oliver Heckmann. Efficiency and Quality of Service of IP Networks. *it - Information Technology Journal*, 3(6):177–180, 2006.

[HL03] Patrick C. K. Hung and Haifei Li. Web services discovery based on the trade-off between quality and cost of service: a token-based approach. *ACM SIGecom Exchanges*, 4(2):21–31, 2003.

[Hom00] Christian Homburg. *Quantitative Betriebswirtschaftslehre*. Gabler Verlag, 3rd edition, 2000.

[JBS97] Stefan Jablonski, Markus Böhm, and Wolfgang Schulze. *Workflow-Management*. dpunkt.verlag, 1997.

[JRS08] Christian Janiesch, Rainer Ruggaber, and York Sure. Eine Infrastruktur für das Internet der Dienste. *HMD – Praxis der Wirtschaftsinformatik*, 261(45):71–79, 2008.

[KBH94] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. The Zero/One Multiple Knapsack Problem and Genetic Algorithms. In *Symposium on Applied Computing*, pages 188–193, 1994.

[KBS04] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA - Service Oriented Architecture Best Practices*. Prentice Hall, 2004.

[KC08] Ramarao Kanneganti and Prasad Chodavarapu. *SOA Security*. Manning Publications, 2008.

[Kel02] Wolfgang Keller. *Enterprise Application Integration. Erfahrungen aus der Praxis*. dpunkt.verlag, 2002.

[KKL03] Sravanthi Kalepu, Shonali Krishnaswamy, and Seng Wai Loke. Verity: A QoS metric for selecting Web services and providers. In *4th International Conference on Web Information Systems Engineering*, pages 131–139, 2003.

[KL03] Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[Kle75] Leonard Kleinrock. *Queueing Systems. Volume 1: Theory*. John Wiley & Sons, 1975.

[Kle76] Leonard Kleinrock. *Queuing Systems. Volume 2: Computer Applications*. John Wiley & Sons, 1976.

[KLMA02] Shahadat Khan, Kin F. Li, Eric G. Manning, and Md. Mostofa Akbar. Solving the Knapsack Problem for Adaptive Multimedia Systems. *Studia Informatica Universalis*, 2(1):157–178, 2002.

[KNS92] Gerhard Keller, Markus Nüttgens, and August-Wilhelm Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). *Veröffentlichungen des Instituts*

für Wirtschaftsinformatik Saarbrücken, 89, 1992. http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf.

[Kov09] Aleksandra Kovacevic. *Peer-to-Peer Location-based Search: Engineering a novel Peer-to-Peer Overlay Network*. PhD thesis, Technische Universität Darmstadt, 2009.

[KS04b] Robert Klein and Armin Scholl. *Planung und Entscheidung*. Verlag Vahlen, 2004.

[Lie08] Nicolas Christopher Liebau. *Trusted Accounting in Peer-to-Peer Environments – A Novel Token-based Accounting Scheme for Autonomous Distributed Systems*. PhD thesis, Technische Universität Darmstadt, 2008.

[Lit04] Marin Litoiu. Migrating to Web services: a performance engineering approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 16:51–70, 2004.

[LJL+03] Kang Chan Lee, Jong Hong Jeon, Won Seok Lee, Seong-Ho Jeong, and Sang-Won Park. *QoS for Web Services: Requirements and Possible Approaches*. W3C Working Group, 2003. http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[LKD+03a] Heiko Ludwig, Alexander Keller, Asit Dan, Richard King, and Richard Franck. A Service Level Agreement Language for Dynamic Electronic Services. *Journal of Electronic Commerce Research*, 3(1 2):43–59, 2003.

[LKD+03b] Heiko Ludwig, Alexander Keller, Asit Dan, Richard King, and Richard Franck. *Web Service Level Agreement (WSLA) Language Specification*. IBM Cooperation, 2003. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[LLRS99] Chen Lee, John Lehoczky, Ragunathan Rajkumar, and Dan Siewiorek. On Quality of Service Optimization with Discrete QoS Options. In *5th IEEE Real-time Technology and Applications Symposium*, pages 276–286, 1999.

[LLSW01] Dahui Li, Zhangxi Lin, Dale O. Stahl, and Andrew B. Whinston. Bridging Agent–based Simulations and Direct Experiments - An Experimental System for Internet Traffic Pricing. In *7th Americas Conference on Information Systems*, pages 720–723, 2001.

[LOSW02] Zhangxi Lin, Peng Si Ow, Dale O. Stahl, and Andrew B. Whinston. Exploring Traffic Pricing for the Virtual Private Network. *Information Technology and Management*, 3(4):301–327, 2002.

[LR99] Frank Leymann and Dieter Roller. *Production Workflow, Concepts and Techniques*. Prentice Hall, 1999.

[LR02] Frank Leymann and Dieter Roller. Business processes in a Web services world – A quick overview of BPEL4WS. 2002. http://www.ibm.com/developerworks/webservices/library/ws-bpelwp/.

[LW04] Frank Lindert and Markus Wiedeler. Organisationsübergreifendes Geschäftsprozessmanagement. *it - Information Technology Journal*, 46(4):175–183, 2004.

[LZGS84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis using Queueing Network Models*. Prentice-Hall, 1984.

[LZR03] Zhangxi Lin, Huimin Zhao, and Sathya Ramanathan. Pricing Web Services for Optimizing Resource Allocation - An Implementation Scheme. *Second Workshop on e-Business*, pages 185–191, 2003.

[Mah07] Zaigham Mahmood. Service Oriented Architecture: Potential Benefits and Challenges. In *11th WSEAS International Conference on Computers*, pages 497–501, 2007.

[Man08] Michael Manitz. Queueing-model based analysis of assembly lines with finite buffers and general service times. *Computers and Operations Research*, 35(8):2520–2536, 2008.

[MBP+09] André Miede, Jean-Baptiste Behuet, Apostolos Papageorgiou, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures. In *3rd IEEE International Conference on Digital Ecosystems and Technologies*, 2009.

[MBR+09] André Miede, Jean-Baptiste Behuet, Nicolas Repp, Julian Eckert, and Ralf Steinmetz. Cooperation Mechanisms for Monitoring Agents in Service-oriented Architectures. In *9. Internationale Tagung Wirtschaftsinformatik*, pages 749–758, 2009.

[Men02] Daniel A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[MGK+09] André Miede, Christian Gottron, André König, Nicolas Repp, and Ralf Steinmetz. Cross-organizational Security in Distributed Systems. Technical Report KOM-TR-2009-01, Technische Universität Darmstadt, 2009.

[ML07] Nilo Mitra and Yves Lafon. *SOAP Version 1.2 Part 0: Primer (Second Edition)*. W3C Recommendation. W3C XML Protocol Working Group, 2007. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown, and Rebekah Metz. Reference Model for Service Oriented

Architecture 1.0. *OASIS (Organization for the Advancement of Structured Information Standards)*, 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[MS02a] E. Michael Maximilien and Munindar P. Singh. Conceptual Model of Web Service Reputation. *ACM SIGMOD Record*, 31(4):36–41, 2002.

[MS02b] E. Michael Maximilien and Munindar P. Singh. Reputation and Endorsement for Web Services. *ACM SIGecom Exchanges*, 3(1):24–31, 2002.

[MT87] Silvano Martello and Paolo Toth. Algorithms for Knapsack Problems. *Annals of Discrete Mathematics*, 31:70–79, 1987.

[MT04] Andreas U. Mauthe and Peter Thomas. *Professional Content Management Systems: Handling Digital Media Assets*. John Wiley & Sons, 2004.

[MW01] Kevin P. McCormack and Charles J. William. *Business Process Orientation: Gaining the E–business Competitive Advantage*. CRC Press, 2001.

[NERS08] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Towards a Generic Governance Model for Service-oriented Architectures. In *14th Americas Conference on Information Systems*, 2008.

[NM02] Srini Narayanan and Sheila A. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *11th International World Wide Web Conference*, pages 77–88, 2002.

[Odl01] Andrew M. Odlyzko. Internet Pricing and the History of Communications. *Computer Networks*, 36:493–517, 2001.

[OSS02] Giwon On, Jens Schmitt, and Ralf Steinmetz. On Availability QoS for Replicated Multimedia Service and Content. In *Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems*, pages 313–326, 2002.

[Pan06] Krishna Pandit. *Quality of Service Performance Analysis based on Network Calculus*. PhD thesis, Technische Universität Darmstadt, 2006.

[Pap03] Michael P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In *4th International Conference on Web Information Systems Engineering*, pages 3–12, 2003.

[PD06a] Yash Patel and John Darlington. A Novel Approach to Workload Allocation of QoS-Constrained Workflow-Based Jobs in a Utility Grid. In *2nd IEEE International Conference on e-Science and Grid Computing*, page 150, 2006.

[PD06b] Yash Patel and John Darlington. Average-Based Workload Allocation Strategy for QoS-Constrained Workflow-based Jobs in a Web Service-Oriented Grid. In *10th IEEE on International Enterprise Distributed Object Computing Conference Workshops*, pages 664–669, 2006.

[PD06c] Yash Patel and John Darlington. Queueing Theory Approach to Allocating QoS-Constrained Workflow-based Applications in a Web Service based Grid. In *Conference on Parallel and Distributed Computing and Systems*, 2006.

[PF02] Shankar R. Ponnekanti and Armando Fox. SWORD A Developer Toolkit for Web Service Composition. In *11th International World Wide Web Conference*, 2002.

[PHB93] H. T. Papadopoulos, Cathal Heavey, and Jimmie Browne. *Queueing Theory In Manufacturing Systems Analysis And Design*. Springer-Verlag, 1993.

[PHD05] Rafael Parra-Hernandez and Nikitas J. Dimopoulos. A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(5):708–717, 2005.

[PSKS06] Krishna Pandit, Jens Schmitt, Claus Kirchner, and Ralf Steinmetz. A Transform for Network Calculus and its Application to Multimedia Networking. In *SPIE Conference on Multimedia Computing and Networking*, 2006.

[PSL03] Chintan Patel, Kaustubh Supekar, and Yugyung Lee. A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows. In *4th International Conference on Database and Expert Systems Applications*, pages 826–835, 2003.

[PTDL06] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing Research Roadmap. In *Dagstuhl Seminar Proceedings on Service Oriented Computing*, 2006.

[PvdH07] Michael P. Papazoglou and Willem-Jan van den Heuvel. Service-Oriented Architectures: Approaches, Technologies and Research Issues. *The VLDB Journal –The International Journal on Very Large Data Bases*, 16(3):389–415, 2007.

[PYY+04] Dunlu Peng, Yang Yuan, Kun Yue, Xiaoling Wang, and Aoying Zhou. Capacity Planning for Composite Web Services Using Queueing Network-Based Models. In *Spinger LNCS*, volume 3129, pages 439–448, 2004.

[Ran03] Shuping Ran. A Model for Web Services Discovery with QoS. *Special Interest Group on Electronic Commerce Exchanges*, 4(1):1–10, 2003.

[RBHS07] Nicolas Repp, Rainer Berbner, Oliver Heckmann, and Ralf Steinmetz. A Cross-Layer Approach to Performance Monitoring of Web Services. In *Emerging Web Services Technology*, pages 21–32, 2007.

[Rep09] Nicolas Repp. *Überwachung und Steuerung dienstbasierter Architekturen - Verteilungsstrategien und deren Umsetzung*. PhD thesis, Technische Universität Darmstadt, 2009.

[RES+08] Nicolas Repp, Julian Eckert, Stefan Schulte, Michael Niemann, Rainer Berbner, and Ralf Steinmetz. Towards Automated Monitoring and Alignment of Service-based Workflows. In *IEEE International Conference on Digital Ecosystems and Technologies*, pages 235–240, 2008.

[Rie03] Rene Riedl. Begriffliche Grundlagen des Business Process Outsourcing. *Information Management & Consulting*, 18(3):6–10, 2003.

[RM08] Nicolas Repp and Wolfgang Martin. *SOA Check 2008: Status Quo und Trends im Vergleich zum SOA Check 2007*. IT-Verlag, 2008.

[RMNS08] Nicolas Repp, André Miede, Michael Niemann, and Ralf Steinmetz. WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In *3rd International Conference on Systems and Networks Communications*, pages 256–261, 2008.

[Row97] Jennifer Rowley. Principles of price and pricing policy for the information marketplace. *Library Review*, 46(3):179–189, 1997.

[Sch01] Jens Schmitt. *Heterogeneous Network Quality of Service Systems.* Kluwer Academic Publishers, 2001.

[Sch02] Jens Schmitt. On the Allocation of Network Service Curves for Bandwidth/Delay-Decoupled Scheduling Disciplines. In *IEEE Global Telecommunications Conference*, volume 2, pages 17–21, 2002.

[SdF03] Mithun Sheshagiri, Marie desJardins, and Timothy Finin. A Planner for Composing Services Described in DAML-S. In *AAMAS Workshop on Web Services and Agent-based Engineering*, 2003.

[SERS08] Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. An Approach to Evaluate and Enhance the Retrieval of Semantic Web Services. In *5th International Conference on Service Systems and Service Management*, pages 237–243, 2008.

[SN96] Roy W. Schulte and Yefim V. Natis. Service Oriented Architectures, Part 1. In *SPA-401-069. Gartner Group*, 1996.

[SN04] Ralf Steinmetz and Klara Nahrstedt. *Multimedia Systems*. Springer-Verlag, 2004.

[SRL01] Burkhard Stiller, Peter Reichl, and Simon Leinen. Pricing and Cost Recovery for Internet Services: Practical Review, Classification, and Application of Relevant Models. *Netnomics - Economic Research and Electronic Networking*, 3(2):149–171, 2001.

[SRSS09] Stefan Schulte, Nicolas Repp, Dieter Schuller, and Ralf Steinmetz. From Web Service Policies to Automatic Deviation Handling: Supporting Semantic Description of Countermeasures to Policy Violations. In *3rd IEEE International Conference on Semantic Computing*, 2009.

[Ste00] Ralf Steinmetz. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, 3rd edition, 2000.

[Sun03] Arun Sundararajan. Nonlinear pricing of information goods. *Management Science*, 50(12):1660–1673, 2003.

[SV99] Carl Shapiro and Hal R. Varian. *Information Rules. A strategic guide to the network economy*. Harvard Business School Press, 1999.

[TA06] Jean-Paul Thommen and Ann-Kristin Achleitner. *Allgemeine Betriebswirtschaftslehre*. Gabler Verlag, 5th edition, 2006.

[Tur86] Jonathan Turner. New Directions in Communications (or Which Way to the Information Age?). *IEEE Communications Magazine*, 24(10):8–15, 1986.

[ULMS08] Tobias Unger, Frank Leymann, Stephanie Mauchart, and Thorsten Scheibler. Aggregation of Service Level Agreements in the Context of Business Processes. In *12th International IEEE Enterprise Distributed Object Computing Conference*, pages 43–52, 2008.

[vdAH02] Wil van der Aalst and Kees Van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[Wes07] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, 2007.

[WM08] Stephen A. White and Derek Miers. *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies, 2008.

[WMC99] WFMC Workflow Management Coalition. TC-1011 Ver 3 Terminology and Glossary English. 1999.

[YCNCC06] Chantal Ykman-Couvreur, Vincent Nollet, Francky Catthoor, and Henk Corporaal. Fast Multi-Dimension Multi-Choice Knapsack Heuristic for MP-SoC Run-Time Management. In *International Symposium on System-on-Chip*, pages 1–4, 2006.

[YGL06] Inbal Yahav, Avigdor Gal, and Nathan Larson. Bid-Based Approach for Pricing Web Service. In *Springer LNCS*, volume 4275, pages 360–376, 2006.

[YL05] Tao Yu and Kwei-Jay Lin. Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. In *3rd International Conference on Service-oriented Computing*, pages 130–143, 2005.

[ZA04a] Li Zhang and Danilo Ardagna. SLA Based Profit Optimization in Autonomic Computing Systems. In *2nd International Conference on Service Oriented Computing*, pages 173–182, 2004.

[ZA04b] Li Zhang and Danilo Ardagna. SLA Based Profit Optimization in Web Systems. In *13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 462–463, 2004.

[Zac05] Roger Zacharias. Serviceorientierung: Der OO-König ist tot, es lebe der SOA-König. *Objektspektrum*, 2(2):43–52, 2005.

[ZBD+03] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality Driven Web Services Composition. In *12th International World Wide Web Conference*, pages 411–421, 2003.

[ZBN+04] Liangzhao Zeng, Boualem Benatallah, Anne H. H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30:311–327, 2004.

[Zin05] Michael Zink. *Scalable Video on Demand: Adaptive Internet-Based Distribution*. John Wiley & Sons, 2005.

[ZSS05] Michael Zink, Jens Schmitt, and Ralf Steinmetz. Layer encoded video in scalable adaptive streaming. *IEEE Transactions on Multimedia*, 7(1):75–84, 2005.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CoS	Costs of Service
FFQN	Feed-forward Queuing Network
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
InvP	Invocation Plan
LR	Latency-Rate
MMKP	Multi-dimensional Multi-choice Knapsack Problem
QoS	Quality of Service
S	Service
SLA	Service Level Agreement
SOA	Service-oriented Architecture
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WSLA	Web Service Level Agreement
XML	Extensible Markup Language

APPENDIX

A.1 DATA GENERATOR

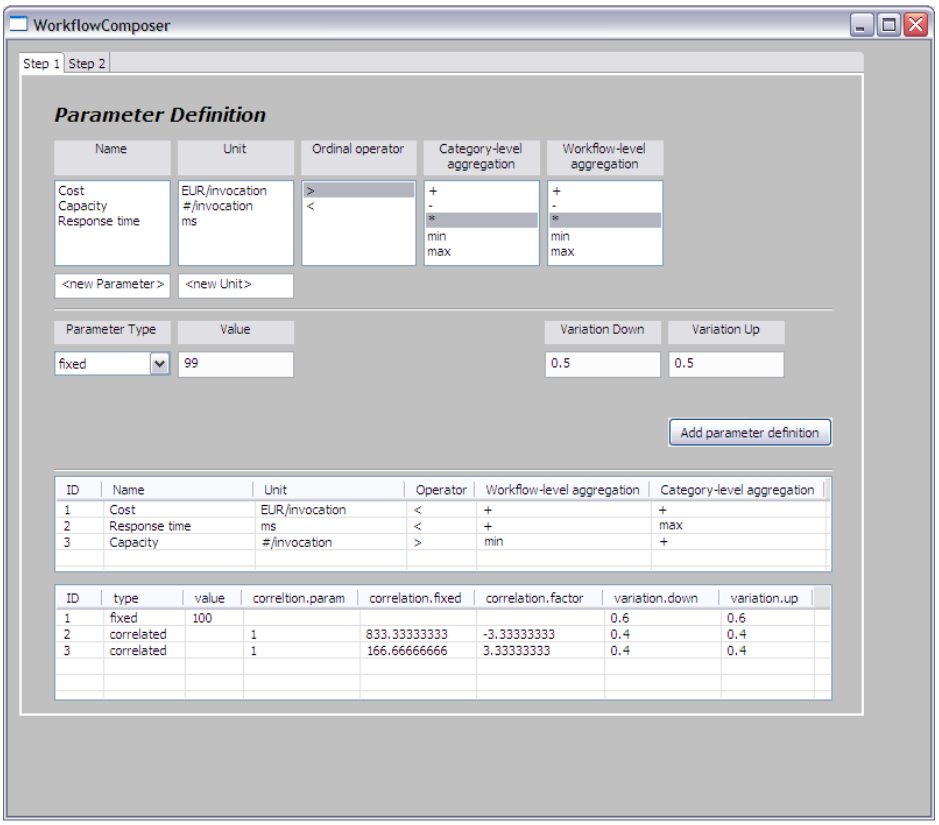


Figure 42: Parameter definition

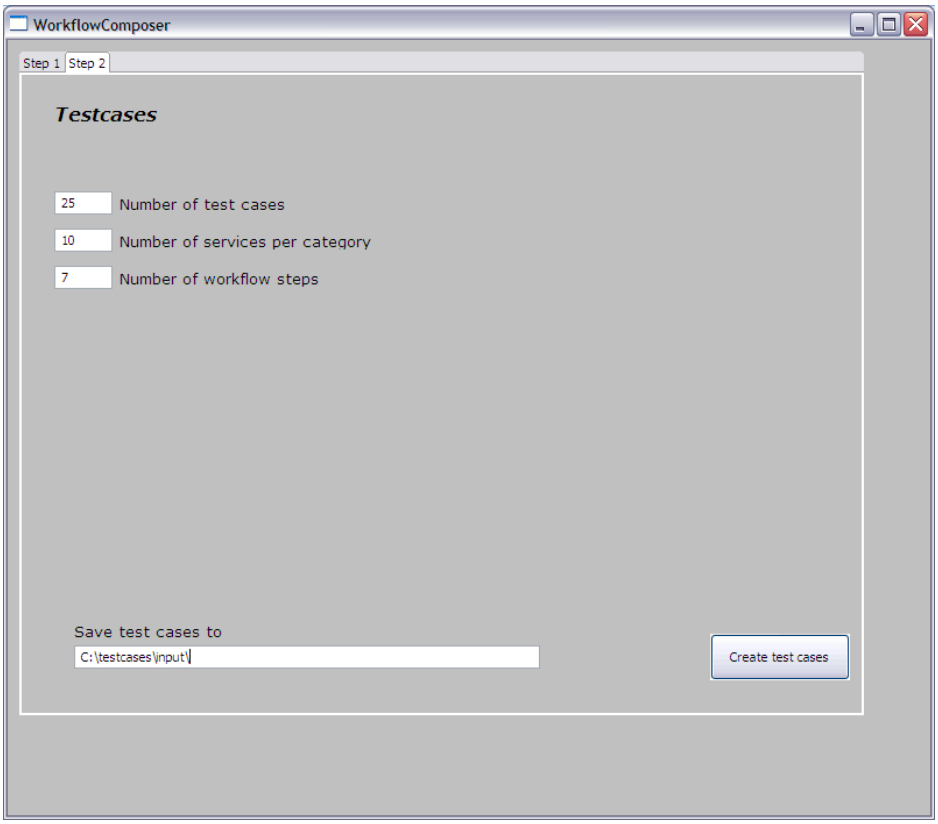
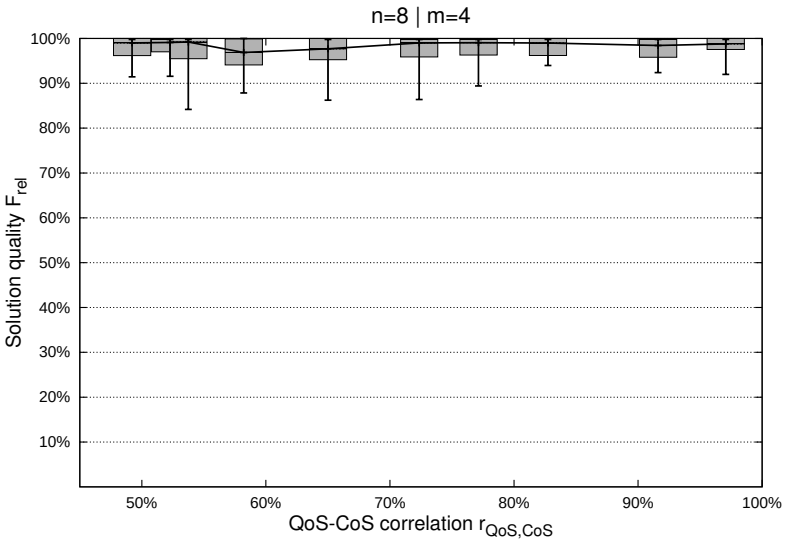
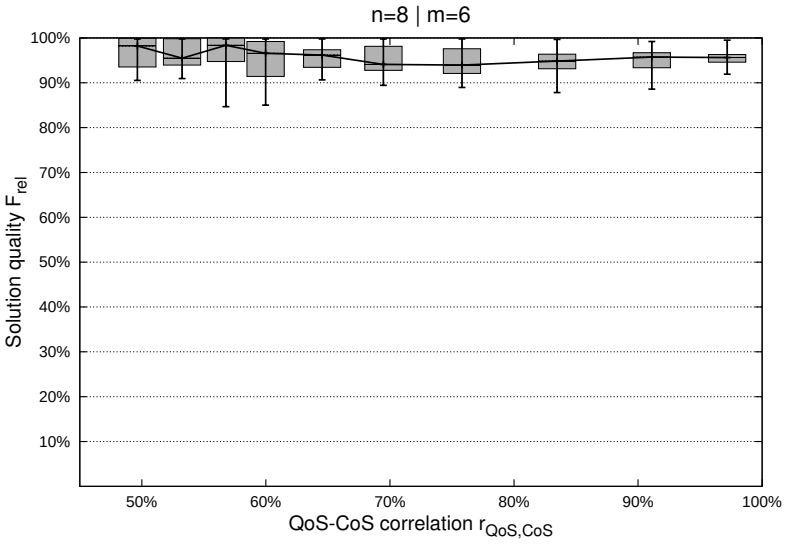


Figure 43: Problem generation

A.2 FURTHER EVALUATION RESULTS

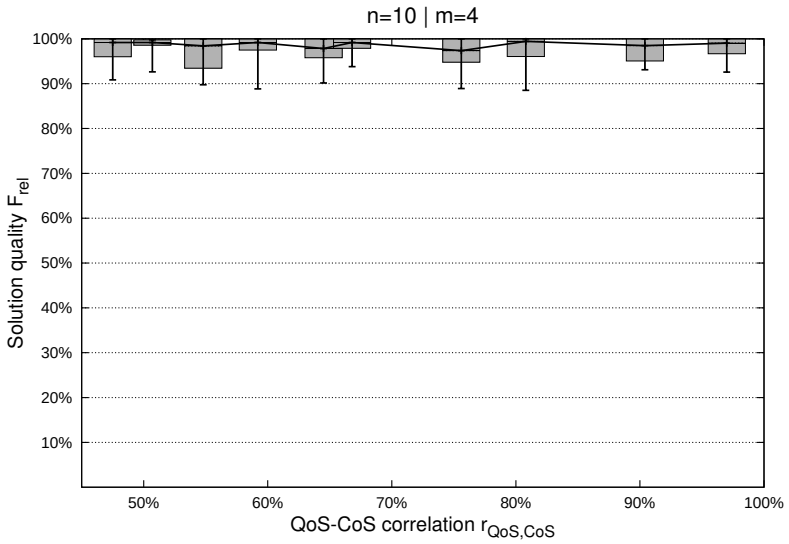


(a)

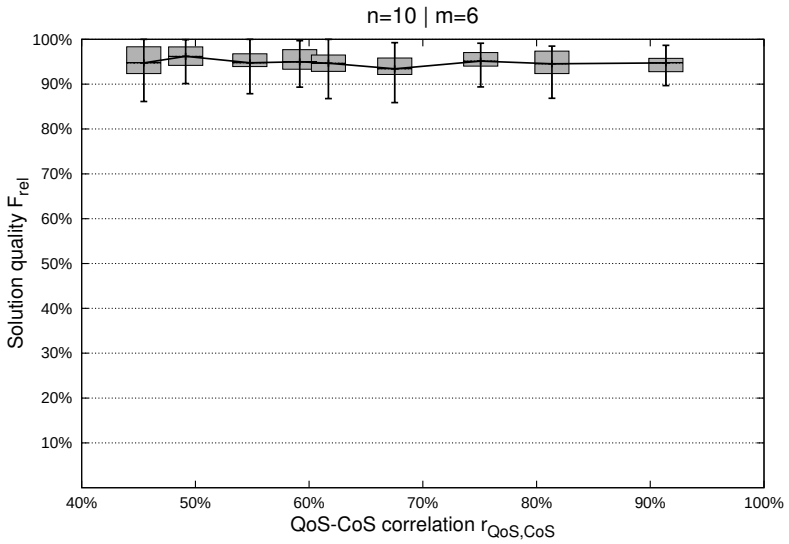


(b)

Figure 44: MSS.KOM – QoS-CoS correlation vs. solution quality n=8, m=4|6

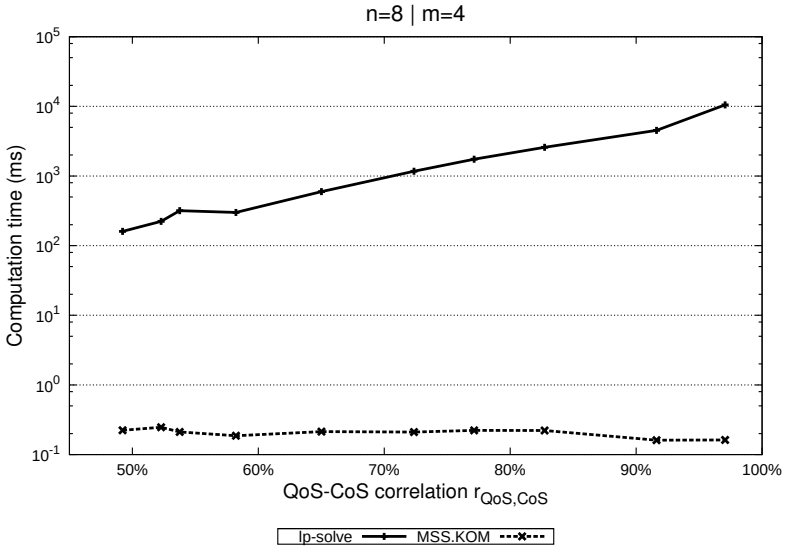


(a)

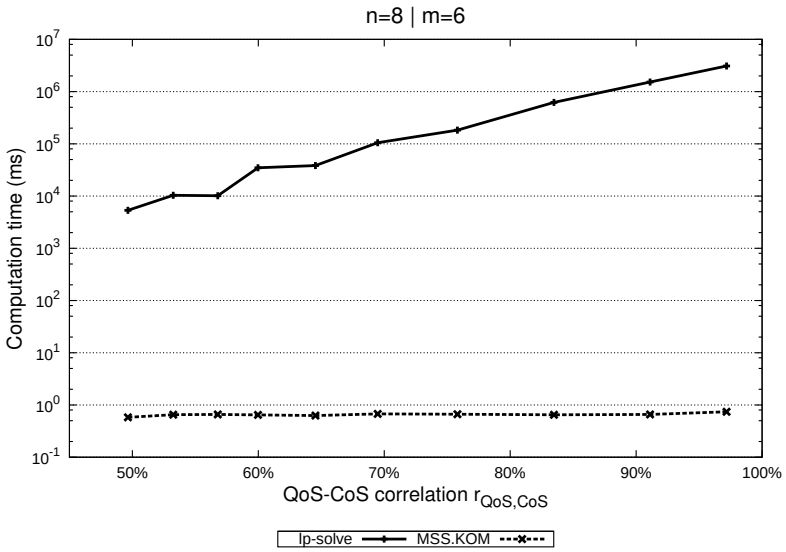


(b)

Figure 45: MSS.KOM – QoS-CoS correlation vs. solution quality n=10, m=4|6

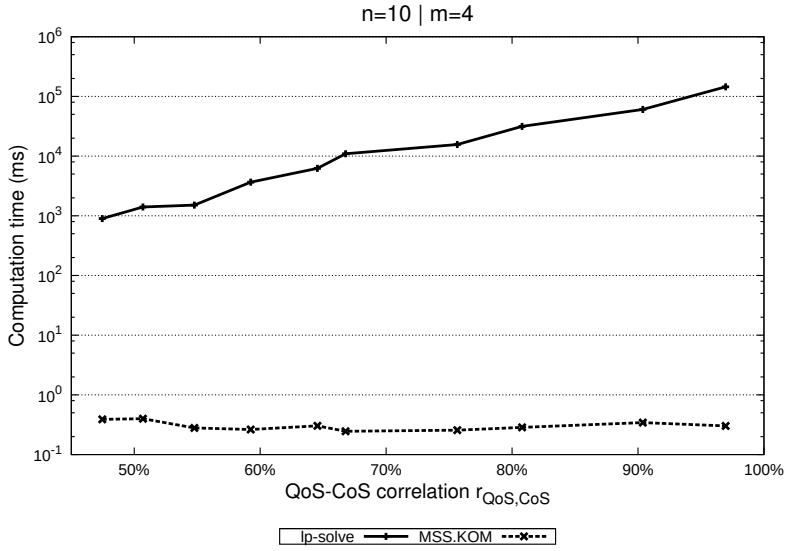


(a)

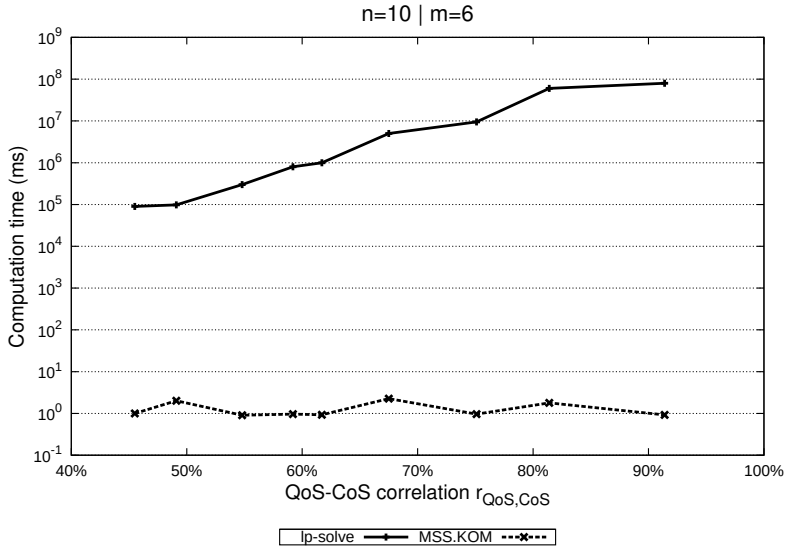


(b)

Figure 46: QoS-CoS correlation vs. computation time n=8, m=4|6



(a)



(b)

Figure 47: QoS-CoS correlation vs. computation time n=10, m=4|6

B

AUTHOR'S PUBLICATIONS

B.1 MAIN PUBLICATIONS

[1] Julian Eckert, Marc Bachhuber, Nicolas Repp, and Ralf Steinmetz. The Implementation of Service-oriented Architectures in the German Banking Industry - A Case Study. In *15th Americas Conference on Information Systems*. 2009.

[2] Julian Eckert, Deniz Ertogrul, Apostolos Papageorgiou, Nicolas Repp, and Ralf Steinmetz. The Impact of Service Pricing Models on Service Selection. In *4th International Conference on Internet and Web Applications and Services*, pages 316–321. 2009.

[3] Julian Eckert, Nicolas Repp, and Wolfgang Martin. *SOA Check 2009: Status Quo und Trends im Vergleich zum SOA Check 2008 und 2007*. IT-Verlag, 2009.

[4] Julian Eckert, Nicolas Repp, Dieter Schuller, Lars Kiewning, and Ralf Steinmetz. Implications of Service-oriented Architectures in the German Banking Industry - A Case Study. *EFL Quarterly*, 2:4–5, 2009.

[5] Julian Eckert. Resource Planning for Distributed Service-oriented Workflows. In *3rd International Conference on Software and Data Technologies*, pages 38–45. 2008.

[6] Julian Eckert, Deniz Ertogrul, André Miede, Nicolas Repp, and Ralf Steinmetz. Resource Planning Heuristics for Service-oriented Workflows. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 591–597. 2008.

[7] Julian Eckert, Nicolas Repp, Michael Niemann, Marc Billeb, Achim Schäfer, and Ralf Steinmetz. IT Organization as a Limiting Factor for the Success of Service-oriented Architectures. *EFL Quarterly*, 2:4–5, 2008.

[8] Julian Eckert, Stefan Schulte, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Worst-Case Workflow Performance Optimization. In *3rd International Conference on Internet and Web Applications and Services*, pages 632–637. 2008.

[9] Julian Eckert, Stefan Schulte, Nicolas Repp, Rainer Berbner, and Ralf Steinmetz. Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms. In *IEEE International Conference on Digital Ecosystems and Technologies*, pages 313–318. 2008.

[10] Julian Eckert, Krishna Pandit, Nicolas Repp, Rainer Berbner, and Ralf Steinmetz. Worst-Case Performance Analysis of Web Service Workflows.

In *9th International Conference on Information Integration and Web-based Application & Services*, pages 67–77. 2007.

[11] Julian Eckert, Nicolas Repp, Stefan Schulte, Rainer Berbner, and Ralf Steinmetz. An Approach for Capacity Planning for Web Service Workflows. In *13th Americas Conference on Information Systems*. 2007.

[12] Julian Eckert, Stefan Schulte, Oliver Heckmann, and Ralf Steinmetz. Decentralized Workflows. Technical Report KOM-TR-2006-05, Technische Universität Darmstadt, 2006.

B.2 OTHER PUBLICATIONS

[1] André Miede, Jean-Baptiste Behuet, Nicolas Repp, Julian Eckert, and Ralf Steinmetz. Cooperation Mechanisms for Monitoring Agents in Service-oriented Architectures. In *9. Internationale Tagung Wirtschaftsinformatik*, pages 749–758. 2009.

[2] André Miede, Steffen Braun, Julian Eckert, Dieter Schuller, Nicolas Repp, and Ralf Steinmetz. A Comparison of Self-Organization Mechanisms in Nature and Information Technology. In *15th Americas Conference on Information Systems*. 2009.

[3] Dieter Schuller, Apostolos Papageorgiou, Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Process Reliability in Service-oriented Architectures. In *3rd IEEE International Conference on Digital Ecosystems and Technologies*, pages 640–645. 2009.

[4] André Miede, Michael Niemann, Stefan Schulte, Julian Eckert, Aneta Kabzeva, Nicolas Repp, and Ralf Steinmetz. Selected Topics in Service Engineering and Management for Enterprise Systems. In *Pre-ICIS Workshop on Enterprise Systems Research in MIS*. 2008.

[5] André Miede, Nicolas Repp, Julian Eckert, and Ralf Steinmetz. Self-Organization Mechanisms for Information Systems – A Survey. In *14th Americas Conference on Information Systems*. 2008.

[6] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. SOA Governance: Survey and Model. In *Pre-ICIS SIG SRV Workshop*. 2008.

[7] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Towards a Generic Governance Model for Service-oriented Architectures. In *14th Americas Conference on Information Systems*. 2008.

[8] Michael Niemann, Melanie Siebenhaar, Julian Eckert, Stefan Schulte, Nicolas Repp, and Ralf Steinmetz. SOA in Practice. Technical Report KOM-TR-2008-04, Technische Universität Darmstadt, 2008.

[9] Nicolas Repp, Julian Eckert, Stefan Schulte, Michael Niemann, Rainer Berbner, and Ralf Steinmetz. Towards Automated Monitoring and Alignment of Service-based Workflows. In *3rd IEEE International Conference on Digital Ecosystems and Technologies*, pages 235–240. 2008.

[10] Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. An Approach to Evaluate and Enhance the Retrieval of Semantic Web Services. In *5th International Conference on Service Systems and Service Management*, pages 237–243. 2008.

[11] Stefan Schulte, Nicolas Repp, Julian Eckert, Rainer Berbner, Korbinian von Blanckenburg, Ralf Schaarschmidt, and Ralf Steinmetz. General Requirements of Banks on IT Architectures and the Service-Oriented Architecture Paradigm. In *Enterprise Applications and Services in the Finance Industry, Lecture Notes in Business Information Processing*, volume 4, pages 66–80. 2008.

[12] Stefan Schulte, Nicolas Repp, Julian Eckert, Rainer Berbner, Korbinian von Blanckenburg, Ralf Schaarschmidt, and Ralf Steinmetz. Potential Risks and Benefits of Service-oriented Collaboration – Basic Considerations and Results from an Empirical Study. In *3rd IEEE International Conference on Digital Ecosystems and Technologies*, pages 155–160. 2008.

[13] Stefan Schulte, Nicolas Repp, Julian Eckert, Ralf Steinmetz, Korbinian von Blanckenburg, and Ralf Schaarschmidt. Service-oriented Architectures – Status Quo and Prospects for the German Banking Industry. *Interoperability in Business Information Systems*, 3(2):9–31, 2008.

[14] Rainer Berbner, Tobias Grollius, Nicolas Repp, Julian Eckert, Oliver Heckmann, Erich Ortner, and Ralf Steinmetz. Management of Service-oriented Architecture (SoA)-based Application Systems. *Enterprise Modelling and Information Systems Architectures*, 1(2):14–26, 2007.

[15] Nicolas Repp, Rainer Berbner, Julian Eckert, Oliver Heckmann, Stefan Schulte, and Ralf Steinmetz. Der Einfluß von Transportschicht-Anomalien auf die Performanz von Web Services. In *5. Fachtagung Kommunikation in Verteilten Systemen*, pages 117–122. 2007.

[16] Nicolas Repp, Stefan Schulte, Julian Eckert, Rainer Berbner, and Ralf Steinmetz. An Approach to the Analysis and Evaluation of an Enterprise Service Ecosystem. In *Workshop on Architectures, Concepts and Technologies for Service Oriented Computing at the 2nd International Conference on Software and Data Technologies*, pages 42–51. 2007.

[17] Nicolas Repp, Stefan Schulte, Julian Eckert, Rainer Berbner, and Ralf Steinmetz. Service-Inventur: Aufnahme und Bewertung eines Services-Bestands. In *Workshop MDD, SOA und IT-Management 2007*, pages 13–22. 2007.

[18] Stefan Schulte, Nicolas Repp, Julian Eckert, Rainer Berbner, Ralf Steinmetz, and Ralf Schaarschmidt. Familiarity with and Usage of Service-oriented Architectures in German Banks. *EFL Quarterly*, 2:4–5, 2007.

[19] Stefan Schulte, Nicolas Repp, Ralf Schaarschmidt, Julian Eckert, Rainer Berbner, and Ralf Steinmetz. Potentielle Auswirkungen des Paradigmas

der Service-orientierten Architekturen auf die Softwarebranche – Ergebnisse einer Studie aus der deutschen Bankenindustrie. In *Science Meets Business*, pages 21–30. 2007.

[20] Stefan Schulte, Nicolas Repp, Ralf Schaarschmidt, Julian Eckert, Rainer Berbner, Ralf Steinmetz, and Korbinian von Blanckenburg. *Service-orientierte Architekturen – Status quo und Perspektive für die deutsche Bankenbranche*. Books on Demand, 2007.

C

CURRICULUM VITAE

PERSONAL

Name	Julian Eckert
Date of Birth	April 29th, 1979
Place of Birth	Darmstadt, Germany
Nationality	German

EDUCATION

since 4/2006	Technische Universität Darmstadt, Germany Doctoral candidate at the Department of Electrical Engineering and Information Technology
8/2005–1/2006	University of Massachusetts Amherst, USA Studies of Joint Degree of Business Administration and Electrical Engineering, research and preparation of the diploma thesis
10/1999–1/2006	Technische Universität Darmstadt, Germany Studies of Joint Degree of Business Administration and Electrical Engineering
9/1989–6/1998	Edith-Stein-Schule, Darmstadt, Germany Degree: Allgemeine Hochschulreife

TEACHING ACTIVITY

Since WS 2007	Technische Universität Darmstadt, Germany Seminar "Communication Systems and Multimedia I: Advanced Topics of Future Internet Research"
Since SS 2007	Technische Universität Darmstadt, Germany Project seminar "Communication Systems Lab III: Advanced Topics in Distributed Systems"
Since SS 2007	Technische Universität Darmstadt, Germany Lecture and Exercise "Introduction to Net Centric Systems"
Since SS 2006	Technische Universität Darmstadt, Germany Tutor for various Bachelor- and Master theses

HONORS

8/2008	Best Paper Award at the Americas Conference on Information Systems 2008 in Toronto, Canada, for the paper *Towards a Generic Governance Model for Service-oriented Architectures*.
2/2008	Best Presentation Award at the IEEE International Conference on Digital Ecosystems and Technologies 2008 in Phitsanoluk, Thailand, for the paper *Towards Automated Monitoring and Alignment of Service-based Workflows*.
2/2008	Best Presentation Award at the IEEE International Conference on Digital Ecosystems and Technologies 2008 in Phitsanoluk, Thailand, for the paper *Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms*.
2005	Scholarship of the German Academic Exchange Office

MEMBERSHIPS

AIS	Association for Information Systems

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt 2009