Optimising Quality-of-Service for the Composition of Electronic Services

vorgelegt von
Diplom-Ingenieur
Michael C. Jäger

Von der Fakultät IV – Elektrotechnik und Informatik –
der Technischen Universität Berlin

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Professor Dr. Hans-Ulrich Heiß
Gutachter:	Professor Dr. Bernd Mahr
	Professor Dr. Robert Tolksdorf

Tag der wissenschaftlichen Aussprache:
1. Dezember 2006

Berlin 2007
D 83

Zusammenfassung

Kompositionen aus elektronischen Diensten finden in Softwaresystemen Verwendung, die bei der Umsetzung von prozess-orientierten Vorgängen in Unternehmen, so-genannten Geschäftsprozessen, zum Einsatz kommen. Dies begründet sich darin, dass Kompositionen aus elektronischen Diensten und Geschäftsprozesse gemeinsame Charakteristiken besitzen. Daher sind Dienstkompositionen für die Realisierung von Geschäftsprozessen besonders geeignet.

Um eine Komposition aus individuellen Diensten zusammen zu stellen, werden Dienstvermittler herangezogen, die anhand von Anforderungsbeschreibungen geeignete Dienste identifizieren. Dabei können Optimierungskriterien berücksichtigt werden, um die Eigenschaften der Komposition zu verbessern. Als Optimierungskriterien dienen in dem berücksichtigten Anwendungsszenario quantifizierbare Dienstgüteeigenschaften. Sollen mehrere Kriterien gleichzeitig berücksichtigt werden, entsteht ein Optimierungsproblem, das in einem unpraktikablen Aufwand resultieren kann. Für dieses Problem soll die Anwendbarkeit von heuristischen Algorithmen untersucht werden. Der Ansatz heuristische Algorithmen auf dieses Problem anzuwenden ist neu und bedarf daher einer Untersuchung: Heuristische Verfahren können die optimale Lösung nicht garantieren. Eine zu beantwortende Frage ist daher, welche Eigenschaften Annäherungen im Vergleich zu einer optimalen Lösung aufweisen.

Zunächst wird ein Verfahren entwickelt, um die Dienstgüteeigenschaften einer Dienstkomposition zu berechnen. Für die Problemstellung ist dieses Verfahren notwendig, um bei der Auswahl eines einzelnen Dienstes die Auswirkungen auf die Komposition zu bestimmen. Basierend auf diesem Verfahren wird ein Modell für das Problem definiert. Anhand dieses Modells wird der Bezug zu verwandten Problemstellungen verdeutlicht und der resultierende Aufwand zur Lösung des Problems diskutiert. Darüber hinaus wird anhand des Problemmodells die Anwendung der heuristischen Algorithmen erklärt.

Mit der Implementierung einer Simulation wird die Leistungsfähigkeit der heuristischen Algorithmen untersucht. Der Begriff der Leistungsfähigkeit bezieht sich hierbei auf die Berechnungsdauer und auf die Dienstgüteeigenschaften der Komposition resultierend aus der jeweilig ermittelten Lösung bzw. Annährung. Die Ergebnisse durchgeführter Simulationen ermöglichen eine quantitative Bewertung der implementierten Algorithmen im Vergleich zueinander als auch den Vergleich zu einem Verfahren, das die optimale Lösung garantiert.

Summary

Electronic services and their composition gain a growing interest from businesses that intend to implement their processes with software systems. The general characteristics of electronic services resemble the idea of process-orientation as proposed by the business process re-engineering initiative introduced in the 90s. Thus, the software industry promotes developing service compositions in order to efficiently implement business processes.

The development of service compositions involves service brokers. These brokers implement a trading functionality in order to identify the suitable services based on requirement descriptions. The trading functionality can also consider different optimisation criteria in order to optimise the resulting composition. In the proposed application scenario, numerical *Quality-of-Service* (QoS) characteristics usually serve as optimisation criteria. When multiple criteria have to be considered at once, an optimisation problem arises that can result in an unfeasible computational effort. A novel approach for this problem is to apply heuristic algorithms. This approach requires a discussion, because heuristic algorithms do not guarantee to find the optimal solution. The question is how well the approximations compare with the optimal solution referring to the resulting QoS of the composition.

Based on the characteristics of the application scenario a method is developed for computing the QoS of compositions based on the QoS statements of the involved services. A QoS-based selection must use such a method in order to determine the QoS of the entire composition when selecting individual services. Based on this method, a model for the problem of QoS-based selection is defined. The model enables the understanding about the problem and it also serves as the reference for the discussion about the relations to other combinatorial problems. Moreover, the problem model is used for the explanation of the heuristic algorithms applied to the selection problem.

Based on the problem model and the relevant QoS concepts, the implementation of a simulation provides the evaluation of the performance of the heuristic algorithms. The simulation presents measures based on the resulting QoS of the composition and the computation time. The results from conducted simulation runs allow the comparison among the algorithms and with a method that always finds the optimal solution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The concept of a service covers many domains of application. In computer science, a couple of attempts to define a service exist in the literature. Often such definitions use assumed technical characteristics such as “defined interface” or “available in a network”. However, services exist that are not available in a computer network and, in general, software components also offer an interface. Such simple examples make clear that it is difficult to define services by referencing to characteristics of existing examples. The ISO 9004 standard offers a definition of services that is free from such references [127, section 3.5]:

> *[A service represents]* the results generated, by activities at the interface between the supplier and the customer and by supplier-internal activities to meet customer needs.

The title of this work contains electronic services (e-services) denoting a special kind of services. The term refers to a service provided by a computer system. Hull et al. also consider the idea that an e-service provides its operations referring to a common purpose, meaning that the different operations of an e-service form a set of interrelated functionality [48]. In addition to these basic characteristics, they emphasise also the following application of e-services:

> ... *[the goal of e-services is]* to have a collection of network-resident software services accessible via standardised protocols, whose functionality can be automatically discovered and integrated into applications or composed to form more complex services.

According to this view, the development and application of e-services also anticipate their composition. E-services and their compositions are supposed to provide their functionality in a computer network. The Internet represents such a network or, in other cases, the network remains in an organisational domain, such as a company. Regardless of organisational boundaries, e-services use protocols and software compatible with the Internet (cf. Huhn et al. [47]).

In the field of service compositions, two main applications are considered: The development of a component-oriented software system represents a first case. In this scenario, a software developer arranges individual e-services to create more complex software. Considering the provision over a network, this application presumes the ability to discover services in these networks or in the Internet. A software developer can consider software components that reside outside his organisational scope or local facilities. Software developing companies can provide their software in the Internet using an e-service infrastructure. Then, customers can integrate these e-services into their software systems. This setup establishes a new market of e-services offered by service providers and new profit mechanisms. In a broader sense, it provides customers with a wider range of products. This application case represents the motivation for using the Web services proposal by the World Wide Web Consortium (in short *W3C*) [11]. The W3C defines a Web service as follows:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards.

The Web services from the W3C represent a proposal to develop e-services using specific XML-based conventions, languages and using Internet protocols.

The second application case targets the software support of existing business processes or the development of new business processes entirely performing within computer systems. Business processes are often associated with workflows. In fact, the Workflow Reference Model published by the Workflow Management Coalition (WfMC) defines a workflow as the implementation of a business process with computer systems: The goal of establishing workflows is *the facilitation or automation of business processes by using computer systems* [45, section 2]. A workflow can include the participation of humans, usually actors in a business process. This work, however, will cover only workflows that do not include operations of users, i.e. do not require human interactions, because e-services represent primarily a technology aiming at the interoperation of software systems.

The use of e-services has already become reality in today's businesses: Last year, a Gartner Group study presented the average numbers of e-services in companies and enterprises [107]. According to this study, an average small company deploys about 25 e-services, while very large enterprises usually have more than 1000 of them. In such large enterprises, more than 100 clients access these e-services more than one million times a day.

As one of the first, Bolcer and Kaiser have recognised the rising potential and introduced a proposal to leverage e-services available in the Internet in order to build workflows [10]. Since then, many publications have mentioned the realisation of workflows as the motivating scenario to develop compositions of e-services. Ganesarajah and Lupu have developed a workflow management system based on compositions of Web services [36]. Hull et al. have discussed the theoretical foundation of e-service compositions including their application to develop workflows [48]. Patel et al. have recently introduced their SemWebQ framework which provides the automated discovery and composition of Web services based on a approach using metadata about e-services to form workflows [105]. In summary, existing research work and the application of e-services in today's businesses indicate the consensus that software developers can use e-service compositions as the technical foundation to implement workflows.

Considering the creation of business processes, the motivation arises to explore techniques to design and develop e-service compositions. For this purpose, different solutions already exist. In most cases, the products and development environments cover the currently most popular service architecture: the Web services architecture from the W3C. This thesis will consider e-services in general and will mention at certain points available technologies in the Web services domain. In order to build such e-service compositions, the following main steps are identified:

- **Design of the composition.** First, a software developer or workflow modeller defines and arranges abstract tasks that provide the desired functionality to form the composition. The outcome is an execution plan and a description of the required e-services. For this step, research work and industry consortia have already proposed *composition languages*.

- **Discovery and selection of e-service candidates.** Based on the description of the required e-services, a discovery process must evaluate available e-services in terms of their suitability for the composition. If more than one e-service suits a particular task, a selection should be based on preference criteria. The Reference Model for Open Distributed Processing (RM-ODP), published by the ISO, names the entire process – consisting of discovery and selection – *trading* [50, section 13]. The trading step results in an assignment of one or more candidates to the tasks in the composition.

- **Provision of the composition.** Based on the composition description and the assignment of available e-services to its tasks, an execution environment provides the composition by executing the individual e-services in this predefined way. In addition, the composition must be advertised to make future trading efforts consider the new composed e-service.

This work focusses on the trading of e-services to form compositions. More specifically, this work will discuss methods and algorithms to perform the selection of discovered e-service candidates based on preference criteria. The discovery process aims at identifying the functional suitability of candidates, whereas the selection assigns the optimal candidates to the tasks of the composition based on the preference criteria. As criteria the quality-of-service (QoS) is considered.

In general, the design of a system must be concerned with QoS in order to deliver dependable and consistent functionality. In a work about QoS in workflow management systems, Weikum has explained the importance of this issue [138]: After decades of development and research, database systems offer functionality with such characteristics. Today, database systems are known for almost never loosing data, offering uninterrupted availability and a sustained level of high performance. At the time when Weikum has written his text, workflow management systems did not offer this level of quality and neither do e-services today. This thesis aims at covering one aspect of QoS of e-services in order to improve their quality for future applications. A detailed discussion about what QoS represents in service compositions will be presented in a dedicated chapter. Until then, the ISO 9004 standard also offers a definition that is considered as a starting point [127, section 3.5]:

> *[The quality is]* the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.

Another more mundane definition is that the QoS denotes how well a service provides its functionality. In the remainder of this work e-services will simply be called *services*. The reference to electronic services will be taken for granted.

1.1 Service Trading

As for trading of services, this work considers the trading specification of the RM-ODP [53]. In general, the RM-ODP represents a model of distributed software systems that is independent of particular technologies. The Web services, mentioned in the previous section, can be seen as an implementation of such a system. Web services represent a popular and widely accepted technology for developing a distributed system. The W3C has introduced different standards and recommendations that cover the message exchange between the actors, the description of interfaces, behaviour of actors and many further specific aspects of distributed systems, such as security, QoS or monitoring. This thesis takes into consideration Web services in order to provide practical examples in addition to the theoretical foundations of the RM-ODP.

Although the RM-ODP and the RM-ODP trading specification cover many aspects of open distributed systems and of trading services, they do not cover the trading of different service types at once in order to build compositions of services. Consequently, trading software referencing the RM-ODP did not focus on this particular form of trading (cf. Kutvonen [76, chapter 3 and chapter 8]). As the previous section has explained, the idea to build service compositions came up later with the development of the Web services idea and the propagation of process-orientation in businesses.

However, regarding the basic concepts of an open distributed system, RM-ODP and Web services identify three main actors in a setup that research work describes as service-oriented architecture (SOA): (1) a *service exporter* is a party that provides a service, (2) a *service importer* invokes a service and (3) a *broker* trades services between ex- and importers. The basic procedures within this setup are as follows:

1. An exporter *exports* his service description to the broker. A broker is a component that implements a trading function and facilitates the matchmaking between requirements of importers and advertisements of services. The Web services corner uses the concept of a discovery service to provide a broker. The service description usually covers the interface and the location of the service.

2. An importer *queries* the broker whether a service is available by submitting a description about his requirements. The broker compares the requirements with his available service descriptions and – if available – returns the description and location of the matched services.

3. After receiving the interface and location of a service, the importer starts the interaction with the matched service. This step is mentioned in the RM-ODP as *importing*.

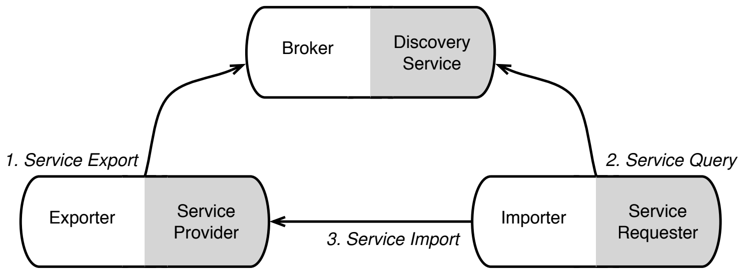


Figure 1.1: Service-Oriented Architecture - The RM-ODP view [50] and its Web service counterparts (shown in grey boxes, cf. WSA [11]).

Figure 1.1 illustrates this arrangement, with the three basic actors and their relations. To facilitate steps 1 and 2, the proposal of the Web services movement favours a specification called Universal Description Discovery and Integration published by the OASIS (UDDI, [135]). The UDDI proposal represents the idea of a centralised repository providing broker functionality. Although the UDDI specification has reached its third revision, software developers still do not use such facilities. So far, no products or development methodologies utilise the discovery of services over the Internet. The main problem lies in two aspects: The first aspect includes an organisational problem. An organisation is not likely to use services of other organisations revealed by an automated service discovery. Usually, business relations need a contract or similar explicit agreement. This problem relates to reputation, trust and contracting issues in this field.

The second problem is that current broker technologies cannot adequately process the requirements of service importers. Different research groups have already proposed extensions for supporting more sophisticated descriptions about the service functionality by using semantic service descriptions (cf. Trastour et al. [134], Paolucci et al. [102], Akkiraju et al. [1], or Srinivasan et al. [123]). Other work considers QoS requirements as the necessary criteria to identify the suitability of available services. In the field of Web services, several authors, for example Ran [112] or Benatallah et al. [6], have already proposed extending brokers with QoS support.

Regardless of currently evolving research to enhance the discovery of services, the ISO has published a technology-independent foundation for service trading, namely the trading specification as a part of the RM-ODP [53]. This specification defines the process of trading as a chain of subsequent isolation operations applied to the set of available services:

- At the beginning, the set $\mathbb{N}_1$ contains all services which the broker has listed. Speaking of Web services, $\mathbb{N}_1$ would consist of all services that some UDDI service contains.

- The first isolation is represented by the set of candidates $\mathbb{N}_2$. This set represents the result of a search for a keyword or similar search criteria among $\mathbb{N}_1$. In the Web service domain, a UDDI repository could provide this functionality.

- The second isolation is reached by matching functionality. This isolation results in the third set $\mathbb{N}_3$. In this context, matching means comparing descriptions from a service exporter with the requirements of an importer. This description can cover the interface as well as other metadata like organisational information. UDDI also covers this issue to some extent, and the above-mentioned research work on semantic descriptions has the goal to improve the efficiency of this step.

- In the third phase, the importer can apply preference criteria, not in order to form a new subset, but to give an order to $\mathbb{N}_3$. Usually, statements about the required QoS represent common preference criteria. Applying preference criteria results in a candidate ranking. The outcome of this step is a tuple T_4^O defined by the order O applied to $\mathbb{N}_3$: $T_4^O = (\mathbb{N}_3, O)$.

- The last isolation is the result of applying return policies. For example, a return policy can restrict the answer to return only one candidate (e.g. the best according to preference criteria). The result is a set $\mathbb{N}_5$ with an order, which is based on a subset of $\mathbb{N}_3$. This can be also seen as a tuple T_5^O with the same order O, applied to $\mathbb{N}_5$: $T_5^O = (\mathbb{N}_5, O)$.

In summary, the trading specification defines that $|\mathbb{N}_1| \geq |\mathbb{N}_2| \geq |\mathbb{N}_3| \geq |\mathbb{N}_5|$. This definition of trading explains that the QoS-based selection must be performed after a preceding matchmaking process has determined the functional suitability. The QoS-based selection results in the decision about which service represents the optimal choice according to the requirements of the service importer.

1.1.1 Trading to Form Compositions

So far, this introduction has discussed the trading of individual services. However, in the case of forming a composition of services, the process requires a new actor. This actor forms the composition by importing services as an importer while he is also offering the composed service as an exporter. The Telecommunications Information Networking Architecture (TINA) by the TINA consortium (TINA-C) provides an existing definition about the roles when composing services [27] and describes this setup as the TINA *Business Model*. This model contains the following roles:

- **A Consumer.** According to the TINA architecture, a consumer just imports services provided by the TINA system. As a consequence, a consumer also imports a composition of services.

- **A Broker.** A broker enables stakeholders of the TINA system to find other stakeholders. In the domain of Web services, the discovery is mainly focussed on finding service providers or retailers.

- **A Retailer.** In the TINA sense, a retailer provides consumers the access to services that are originally provided by 3rd party providers. Thus, a retailer represents the role that provides consumers also with compositions of individual services from 3rd party providers.

- **A 3rd Party Service Provider.** A 3rd party service provider offers his services to a retailer or other 3rd party service providers. The self-relating link from a 3rd party provider matches the idea of composing services: A composed service may also integrate other composed services as a part of its composition.

In addition, the TINA also defines relations among brokers. This relation implies that a broker can contact other brokers to enhance the discovery of services. Moreover, a retailer can access services provided by other retailers, which, in this sense, would represent 3rd party providers. Figure 1.2 shows the TINA business model and points out the analogies to an environment of service compositions.[1]

[1]The TINA business model mentions a *connectivity provider*. In the field of Web services, the Internet provider represents the connectivity provider. However, the use of services usually does not interfere with any aspects of the underlying network protocols in the Internet. Thus, the connectivity provider is neglected in this discussion.

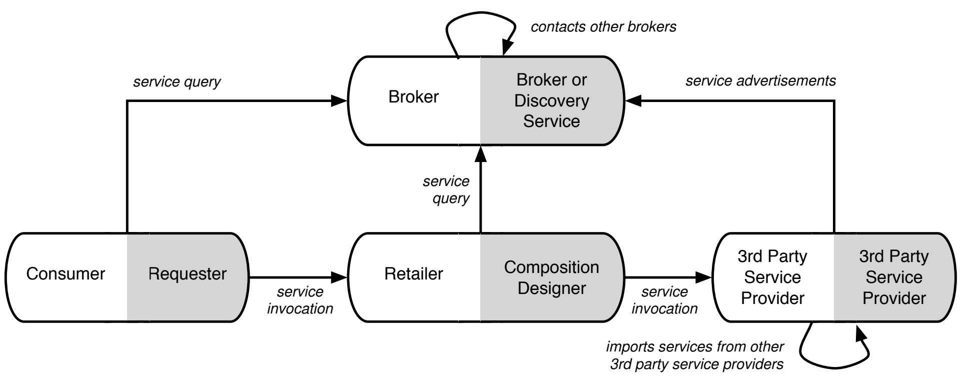


Figure 1.2: The business roles in the TINA [27] and roles in the domain of Web service compositions (shown in grey boxes).

The retailer plays the central role in developing and providing compositions. The retailer has the following characteristics:

- A retailer imports services from 3rd party service providers to integrate them in his composition.
- A retailer queries a broker for discovering services. As for his composed service, the retailer publishes his offerings of composed services to the broker as well.
- A retailer provides the composed service to consumers. According to the business model, the retailer also offers his composed service to other retailers as a third party provider from their perspective. Considering this distinction, a consumer represents the concept of an *end customer* in this setup.

The Special Role of the Retailer

The TINA business model does not mention any role that performs the entire creation process of a composition. Neither is it discussed which role takes which responsibility when forming compositions. In this thesis, the creation process is assigned to the retailer; the retailer represents the *developer* and *provider* of the composition. The previous section has introduced the different steps of trading. A broker could cover the complete trading functionality as described by these steps. However, when forming a composition, the trading involves an acting party with knowledge about the entire composition aiming at optimising the trading result. The TINA business model provides two actors, the retailer and the broker, for this trading process. It is clear that a broker is generally capable of trading *individual* services.

However, the trading of multiple services to form a composition requires special consideration. For example, a retailer can consider the requirement that the execution of the integrated services will not exceed a given duration. From the local perspective of the broker when trading individual services, such trading cannot meet global requirements. Thus, the retailer must perform the trading of individual services from a global perspective in order to ensure the desired result. Having looked at the trading specification and the TINA, the topic of this thesis can be narrowed down in the following way:

> *The QoS-based selection involves QoS as preference criteria as a part of a trading process. If a retailer builds a composition, he can perform a QoS-based selection on the service candidates in order to optimise the resulting QoS of the composition.*

One issue still remains open: *where precisely lies the difficulty of the QoS-based selection?* The before-mentioned argumentation implies that an issue exists with the trading of services and requires a global view on the composition. The following example clarifies the difference between trading from a local and from a global perspective: Figure 1.3 shows a model of an example composition.

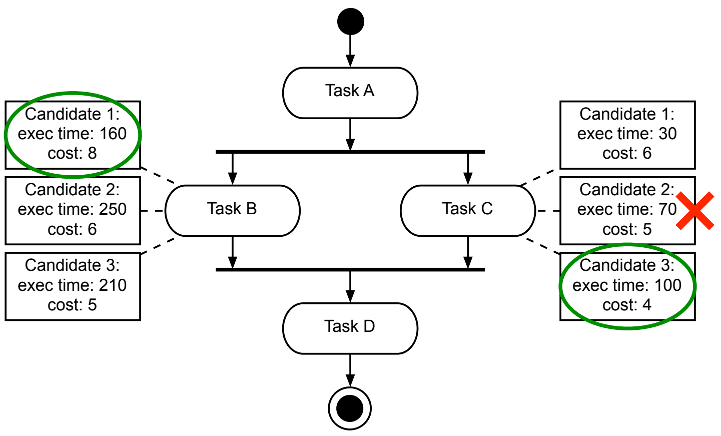

Task A
Task B
Task C
Task D
Candidate 1: exec time: 160 cost: 8
Candidate 2: exec time: 250 cost: 6
Candidate 3: exec time: 210 cost: 5
Candidate 1: exec time: 30 cost: 6
Candidate 2: exec time: 70 cost: 5
Candidate 3: exec time: 100 cost: 4

Figure 1.3: A simple example of the QoS-based selection problem.

This composition consists of four tasks. After executing the first task A, two subsequent tasks B and C are executed simultaneously. Then, after both tasks are finished, a fourth task D is executed. Considering the parallel arrangement of the two tasks B and C, it is assumed that a preceding matchmaking process has identified three service candidates for each task, each providing different QoS. In addition, it is assumed that the optimisation goal is to form the quickest composition with the lowest cost. In the given example, the quickest candidate for task C needs longer than any candidate for task B. Consequently, the optimal assignment for the task C is candidate 3. A selection from a local perspective, as performed by a broker without knowledge about the composition, would have identified candidate 2 for task C. Clearly, this would have resulted in a higher cost for the composition.

An approach from a global perspective would consider this potential optimisation. A naïve algorithm simply evaluates all possible combinations. This thesis explains that a combinatorial problem arises from this strategy. Using a naïve approach can result in unfeasible efforts. If the number of candidates increases by one, the number of combinations to evaluate is doubled. More tasks in the composition will result in a higher number of candidates as well. The quickly rising computational effort poses a problem for the retailer when performing the QoS-based selection for larger compositions. Thus, this thesis concentrates on discussing the problem and on evaluating the application of heuristic algorithms. Such approaches may identify the optimisation as outlined in the example and lead to adequate results while showing lower computational efforts in comparison to a naïve approach.

1.2 Problem Statement

The previous sections explained the motivation of this thesis, narrowed down its topic and explained the discussed problem. Based on these, a problem statement is formulated:

> *A retailer that builds a service composition can improve the provided QoS of the composition by means of a QoS-based selection that considers the QoS of the individual service candidates.*
> *The QoS-based selection results in a combinatorial problem of quickly rising computational efforts for a growing number of candidates. The development of heuristic methods for solving this combinatorial problem represents a novel approach. In order to assess the feasibility of this approach, the performance of the algorithms when applied to the selection problem must be evaluated.*

The foundation for the outlined discussion and assessment is a problem model of the selection problem. Based on this model, the application heuristic algorithms can be explained and their efforts can be discussed. The problem model can also serve as the foundation for the implementation of a simulation for performance evaluations. In conjunction with the definition of appropriate metrics, the performance penalty of the heuristic approaches can be assessed. This discussion is based on findings and techniques that require an explanation in advance. Thus, the argumentation is divided into four steps building on each other:

1. **Determining the QoS of Compositions.** In order to evaluate the QoS optimisation performance of the considered heuristics, a method is presented that determines the QoS of the composition based on the provided QoS of the individual services. To ensure its applicability, this method must not be

limited to specific QoS measures used in particular application cases. Moreover, it must be capable of processing different compositions, referring to their structure and size.

2. **Modelling the Problem of a QoS-based Selection of Services.** A problem model is presented that allows understanding the combinatorial issue with the QoS-based selection. Based on the problem model, the relation to related problems can be discussed in a more precise manner.

3. **Explanation of the Heuristic Algorithms.** Different heuristic algorithms are proposed and their implementation is discussed. Some heuristic approaches exist for other combinatorial problems such as the knapsack problem. The goal of this discussion is the potential evaluation of existing heuristics with regard to their suitability for the problem of QoS-based selection.

4. **Evaluation of Heuristic Algorithms.** An implementation of the simulation performs the algorithms for their evaluation. A first part covers the setup of the simulation and its set parameters. It is expected that the heuristics will perform differently, depending on, for example, the structural characteristics of the composition or on the variance of the provided QoS by the service candidates. Based on this, different simulation campaigns are defined. By varying parameters and setup, the simulation results indicate specific weaknesses or strengths of the heuristics.

1.2.1 Research Issues

The previously explained parts of the argumentation cover different open research issues. In order to provide the argumentation as outlined, the following crucial research questions must be covered:

- **Aggregation Method.** Different methods already exist for the aggregation of QoS in service compositions. However, these methods either discuss specific QoS characteristics such as response time or cost (cf. Yu and Lin [158]) or they focus on compositions that consist of a sequential execution of services (cf. Lee [78]).

  Apart from the composition of services, aggregation methods exist that cover the execution time of software in real-time environments (cf. Puschner and Schedl [111]) or the aggregation of the QoS in workflows (cf. Cardoso [15]). Based on the existing research work, the presented aggregation method supports different QoS characteristics and is tailored to the possible structures as found in service compositions.

- **Problem Model.** Different research work also discusses the combinatorial problem that arises when a QoS-based selection is performed to form compositions (cf. Lee [78], Zeng et al. [159], Yu and Lin [158]). But, similar to the aggregation method, existing discussions consider only specific QoS categories, i.e. response time and cost (cf. Lee [78], Yu and Lin [158]), or they reduce the problem to a sequential execution of services (cf. Lee [78], Yu and Lin [158]).

All three mentioned research efforts in this area define the problem of the QoS-based selection as a variant of a knapsack problem, namely the multiple-choice knapsack problem (MCKP, cf. Lee [78], Zeng et al. [159], Yu and Lin [158]). Consequently, the approach is to apply existing solutions to the MCKP in order to perform the QoS-based selection. This thesis explains, why this represents a simplification that does not cover all problem cases: very briefly, the MCKP-approach is tied to QoS characteristics that result in an integer linear optimisation statements and also covers mainly a sequential execution of services. In contrast to this, a problem model is defined that is independent of particular QoS categories, covers different structural elements found in compositions and, thus, is not equivalent to the MCKP. Then, existing solutions cannot be applied. As a consequence, the application of heuristics is discussed as an approach to deliver near-optimal solutions while requiring reduced efforts as compared to methods that guarantee optimal solutions.

- **Simulation Setup.** Results from previous research about a simulation environment for the QoS-based selection have shown that the efficiency of the applied heuristic algorithms depends on different simulation parameters (cf. Jaeger and Goldmann [62]). Thus, the goal is to deduce from existing studies the nature of the needed parameters in order to design realistic simulation conditions.

  Furthermore, this thesis identifies the impact of the simulation parameters on the efficiency of the heuristic algorithms. For this, different simulation campaigns will examine the particular impact of different parameters. Then, the evaluation of the heuristic algorithms provides more precise findings about their strengths and weaknesses.

1.3 Structure of the Thesis

The structure of this thesis follows the argumentation of the previous section. Before the argumentation starts, Chapter 2 clarifies in detail the origin of service compositions and their main application, which is the implementation of business processes. It also discusses the relation between workflow and business process management. Then, different approaches of modelling business processes and compositions are presented.

Chapter 3 discusses the QoS of service compositions in detail. It begins with clarifications on the different concepts and on the used terms in this field. Then, it explains how QoS needs to be processed in order to perform the QoS-based selection. Based on these clarifications, Chapter 4 presents the aggregation of QoS. The chapter is divided into two parts: Its first part covers a structural model independent of particular technologies. Its second part deals with the aggregation of QoS based on this model.

Chapter 5 explains the problem of QoS-based selection and defines a problem model. Furthermore, this chapter discusses the characteristics of the problem and continues with the introduction of the heuristic approaches. Chapter 6 explains the simulations, the different campaigns, and how the simulations are performed. Then, the chapter also presents the results and discusses their interpretation. The chapter ends with an analysis of the efficiency of the applied heuristic approaches.

After the discussion of the selection problem, Chapter 7 explains how the QoS-based selection is integrated into the process of developing service compositions. The chapter introduces different research works that discuss designing and creating compositions. Based on the presented contributions, a basic development process is presented that provides a description about its required activities and facilities. Then, the chapter identifies at which points the QoS-based selection is performed. In addition, it clarifies which information the process must provide in order to perform the QoS-based selection as well as how the output of the selection is used for subsequent tasks in the process.

Chapter 8 provides a summary of the research contributions provided by this thesis and presents the conclusions. It also discusses possible improvements and envisages future directions of this work.

Chapter 2

Workflows, Business Processes and Service Compositions

This section will introduce background information of the application scenario that is considered in this work: As briefly mentioned in the introduction, the goal is to develop business processes and workflows by means of compositions of services. In such a scenario, software components available as services perform individual tasks of a business process or a workflow. Such a setup is often embedded into an IT infrastructure that is tailored to the provision and utilisation of services, the SOA. This chapter intends to clarify the terms *business process* and *workflow*, and their relation to each other. Because the structure of processes will become important for the subsequent chapters of this work, this chapter will also introduce different modelling languages for business processes, workflows and service compositions.

2.1 Business Processes

In the mid-90s, the term "business process (re)engineering" drew attention to a number of opportunities for optimising the efficiency of enterprises, companies and other organisations. The work of Hammer and Champy, who promoted the reengineering of business processes [43], drew the attention of the IT industry to developing software systems that facilitate the creation and management of business processes [86] [34, p. 230].

The basic idea of this initiative is to implement business processes in an existing organisation in the most modern and optimised way: The business process reengineering should result in a new and optimal process without any legacy artefacts. The basic approach is to start with an evaluation and analysis of the activities within an organisation. Based on the analysis, the goal is to redesign the activities and to group them into defined processes. A business process should provide a clear and unambiguous definition about what it does, what its output is and what it needs as an input. This represents a clear analogy to computer programs: In general, they also feature unambiguously defined in- and outputs, as well as a clear definition about what they do, which is represented by the source code of a program. In addition, every business process should have a dedicated customer – either internal or external of the organisation – and thus, have a clear purpose. By this way, a process can be clearly oriented to the needs of a customer. And it will be the client of the process who will pay for the results of its execution. Besides the consumer, a process should also have an owner, who is in charge and responsible for a particular process, in order to provide customers with a defined point of contact.

When the business process paradigm was introduced, just the opposite situation was the reality in companies: Different organisational units where divided by their functional responsibilities. As a consequence, a process typically crossed many organisational parts and involved a number of responsible persons. In such constellations, the average process duration slowed down. Thus, in case of problems or inquiries it was hard to identify a responsible person. Among different motivations and anticipated benefits from applying the business process reengineering idea, the main drivers were (cf. Krallmann et al. [34, p. 230]):

- **Optimisation of existing activities.** The business process reengineering gives the opportunity to re-evaluate the advantages and disadvantages of what the actors in the business do and how they do it. The obvious goal is to optimise existing activities. For example, it is possible to evaluate whether sequentially performed tasks can be performed in parallel in order to save time.

- **Improved controlling.** An organisation might perform many different individual activities. As a result, common metrics and comparison values are hard to apply. This makes monitoring and benchmarking efforts more difficult. Based on the standardised processes, monitoring and benchmarking the ongoing processes will results in values and findings that are more suitable for comparison and analysis. Moreover, controlling efforts to prevent unintended activities can be reduced.

- **Reducing overhead.** What applies to the controlling, also applies to when the process is performed. Based on a exact definition of the processes, mistaken activities or misunderstandings between involved actors are reduced and thus the productivity is improved.

Apart from the evident advantages, analyses of the performed business process reengineering efforts have also revealed problems that may occur. Most noticeable is that applying too radical changes in order to establish more efficient processes will lead to social problems in the organisation [34, p. 239]. Moreover, a strong focus on the process optimisation also carries the risk of poor improvements on the quality of the individual activities.

2.1.1 Definition of Business Processes

Like with many terms in the field of IT, there are many definitions available for the term business process. The general definition of a process provided by the ISO 9000 standard is that a process "is a transformation that adds value" [128, section 4.6]. Hammer and Champy [43] define a business process as "a set of activities that, taken together, produces a result of value to a customer". Davenport defines a business process as "an ordering of work activities across and place, with a beginning, an end, and clearly identified inputs and output" [21]. There are many more definitions available in various books. When considering the basic attributes of business processes as discussed in the mentioned literature, they show some similarities with service compositions:

- **Input and output.** A process has a defined input and output. This idea has clear analogies to the compositions of services that provide a defined input and output as well. A composed service starts and ends each with an individual service. A service represents a software operation which has input and output parameters as well.

- **Purpose.** A process should address at least one goal. This aspect is also inherent to the composition of services, which must follow a goal as well.

- **Responsibility.** A process has one responsible person or unit. In the domain of service compositions, the role of the retailer who provides the composed service represents this function.

- **Recipient.** A process has at least one consumer. In this respect it is analogous to the service composition, because the retailer would not provide the composed service if there were not consumers using the service.

- **Activities.** A process consists of activities. The idea of activities that together form a process also resembles the nature of a service composition, which consists of individual services.

Consumers or clients, i.e. responsible and acting individuals, play defined roles, which are attached to a process. Besides the general attributes of a process, constraints can be also applied. For example, a constraint exists when a process must generate a positive value or that the goal of the process is to serve a consumer with a high performance. Furthermore, a business process can be divided among organisations, which increases the efforts to establish and to run the process. The aspect of covering business processes between organisations also matches the nature of service compositions: The used techniques and technologies allow involving individual services from different organisations as well.

2.1.2 Modelling Business Processes

A model of a business process models a set of activities and its relations. This set can be described with inputs, outputs and a definition of the involved roles. The basic idea of the modelling step for the software engineering side, as well as for the business process engineering mentioned before, is to achieve a clear and common understanding of what a business process should do and what its benefits are. A business process model thus can serve as the common point of understanding between the management and the software developers that are supposed to develop the service composition. Then, the realisation of the process can be planned and the technical feasibility can be assessed. Business process models are also the foundation of business process analysis aimed at identifying the potential for improvements. Possible improvements are automation, elimination of unnecessary media changes or the reduction of delays. A model can also help to verify processes in order to prevent live- or deadlocks which prevent the process from terminating properly. Figure 2.1 shows the basic two roles of a business process model with respect to the business process and a service composition.

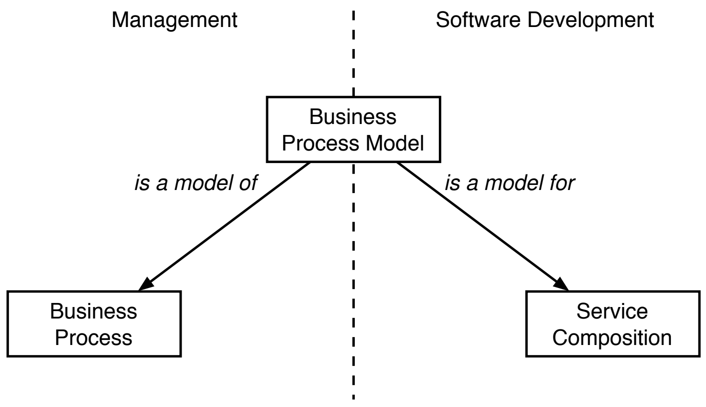


Figure 2.1: The roles of a business process model.

Since this topic has gained reasonable attention from the industry, the number of companies that offer products and services for business process (re)engineering activities has increased. In addition, various languages and methodologies for modelling, managing or performing business processes were introduced. For the modelling, different graphical and textual languages and conventions exist, which can be used to create diagrams or a description of a business process. Then these descriptions can be interpreted by software systems. Graphical representations can be flow diagrams, block diagrams, graphs or listings. Considering a basic graph, a node represents an activity, an event or an entity where directed edges represent the relations between the elements.

One early graphical language for the modelling of processes is the Event-

Driven Process Chain (EPC) [71]. As the name suggests, the basic element of the event-driven process chain is the event, which is a defined condition and thus can be the result of a process, a function or an external event. In addition, events can also trigger a function. Because the events are not routing the flow, events are regarded to be passive. Contrary to events, a function is an active element which describe state changes. The events or functions can be combined with routing operators. With EPCs, conjunctive (AND), disjunctive (XOR) and adjunctive (AND-OR) operators are supported. EPCs are suitable for the modelling of control flows that define the order of occurring events and executed functions. To model the data flow of a business process (or also the flow of goods) extensions are proposed that appear in literature as extended EPCs [34, p. 221]. However, modelling the data-flow is not the focus of this work and therefore not subject to further investigations.

The Business Process Modelling Language (BPML, [29]) is a specification of the Business Process Management Initiative (BPMI) and is a textual language for describing business processes. The BPMI represents a non-profit organisation, with the goal to support and coordinate the advances in business processes among its members. The BPML is intended to serve as a comprehensive description of a business process. It consists of different constructs to describe the control flow of a business process as well as the data flows in it. The standard representation used for BPML documents is XML. The BPMI has also released a graphical notation called Business Process Modelling Notation (BPMN, [140]) to provide a set of graphical symbols and layout conventions for drawing business process models. In addition to the BPML, the BPMI has also discussed differences of business processes management when compared to the workflow management (cf. Smith at al. [121, 122] and van der Aalst [141]). Very briefly, this discussion has revealed that differences are hard to specify. Rather, business processes and workflow management show many similarities, as the next section about workflows will point out as well.

Considering another modelling proposal in the field of business processes, the Business Process Execution Language (BPEL), also named BPEL for Web Services (BPEL4WS) and now being renamed to WS-BPEL, represents a special proposal because it specifically covers Web services. Although its name, mentioning execution, indicates a different scope than BPML, which carries the name modelling in it, both proposals compete with each other. At the moment, a committee at the Organisation for the Advancement of Structured Information Standards (OASIS) coordinates the development of BPEL. Before, BPEL was carried out by a joint venture of mainly IBM, Microsoft and BEA.[1] Originally BPEL was the result of a merger of the Web Services Flow Language by IBM (WSFL, [80]), which shows influences from IBM's MQ Series workflow software [143], and XLANG [129], which was intended to serve as the process modelling language in Microsoft's BizTalk middleware. All the three languages are designed for realising the activities or tasks of a process by using Web services. Following the Web

[1]Today, additional main contributors to the development of the BPEL are the software companies Siebel and SAP.

services paradigm, all three proposals define an XML-based notation as well as cover elements that a process modeller can use to directly refer to WSDL interface descriptions of involved Web services.

The main difference between the two languages WSFL and XLANG is that XLANG provides a block-structured flow description whilst a process description using WSFL is oriented to an unrestricted graph. The main process element found in XLANG can represent basic structural arrangements like a sequence or a conditional branch. Contrary to that, WSFL covers activities that represent the nodes in a process flow graph. Then, WSFL provides elements to define the edges in the graph which denote the invocation order among the activities. Consequently, the combination of both, BPEL4WS, features both ways to model a business process, using block-structures as well as a flow-graph. For example, looking at the elements provided to describe the control flow structure in BPEL4WS: It can be seen that the element all from XLANG meaning that all activities are supposed to be executed in parallel has been dropped. Contrary to that, the other parallel statements (pick, while, switch) have survived in BPEL4WS. In addition, the element flow replaces the element all from XLANG and also introduces the concept of control links as originally found in WSFL.

Besides the mentioned EPC, BPML, BPEL, WSFL and XLANG, there are many other approaches to model business processes. In addition, the literature mentions WSFL, XLANG and BPEL4WS as languages for modelling compositions of Web services, which also indicate that the border between business processes and service compositions is becoming blurred. Another proposal for modelling business processes is to use the Unified Modelling Language (UML) from the OMG, for example by using activity diagrams [100, section 2.13.2.1]. Originally intended for "software-intensive" systems, as the foreword of the UML specification says, this approach is used by some software products. This is also proposed in related research works, especially when it comes to the realisation of business processes. Section 7.1.1 will introduce some efforts in more detail.

2.2 Workflow Management

The field of workflows has got a different origin and thus also a different history than the field of business processes. Not following an application-independent approach as a general strategy for organisations, first workflow management systems were applied for specific application cases. One of the systems mentioned as the first steps in the workflow area is the OfficeTalk software, which came as a part of the Xerox Star system [68]. The Xerox Palo Alto Research Center (PARC) developed this system in the 70s. It represented a computer system for working with electronic documents of different kinds, like texts, memos, messages etc. in a typical office environment. Such a system does not represent a workflow management system as it is understood today: Today, many works consider the definition of the Workflow Management Coalition (WfMC) [45, section 2.1] which says that

[A workflow represents] the computerised facilitation or automation of a business process, ...

The way OfficeTalk worked was trying to reflect the way humans would work on documents without computers and thus it was covering a specific process. This represented a workflow by realising real-world processes with a computer system. Clearly, workflow has in this sense a strong relation to support collaboration and document management.

Other systems that also focussed on the workflow around electronic documents followed. The primary purpose was to support the creation of documents, to send them around and to control the evolutions of these. The workflow management controlled the order in which the users would work on a document. Also the systems tried to offer a seamless integration with e-mail systems, word processing applications and input forms in order to standardise the user input. In summary, workflow management systems optimised the handling of documents in the following ways:

- **Prevention of media breaks.** When dealing with documents and data, it happens that the media, which transports the information, changes. E.g. a document is printed out on paper from an electronic document and needs to be entered in manually into another system. This is named media break. It was the hope to reduce such media breaks with workflow management systems and to come one step closer to the paperless office.
- **Accelerated forwarding between tasks.** Moving a document from one's desk to the next took time and relied on the motivation of the employees. Workflow management systems can forward documents right after one party has finished its tasks.
- **Controlling.** When workflow management systems facilitate or automate a workflow, data can be derived that allows performance measurements. Moreover, such data can be used to predict future performance for controlling purposes.
- **Automation of tasks.** Tasks that do not require an interactive handling can be automatically executed by a workflow management system and thus accelerate the workflow.

These early workflow management systems developed further. The systems became more sophisticated and more compatible to external systems. An article by Mahling et al. explains the evolution of workflow management systems by referring to the Poise system which covers all the industry's developments since its beginning in the 70s. The first version of the Poise development represented an office information system supporting tasks that occurred in the handling of documents, such as entering information or realising static workflows [86]. In the mid-80s, a subsequent development named Polymer took advantage of the lessons learned with Poise: Its main innovation was a more sophisticated concept of modelling workflows that resulted in more flexible workflows as well as in better coverage of different application scenarios. Based on the previous products, Polyflow was introduced in 1995 as an application- and domain-independent workflow management system.

Besides the early orientation of workflow management systems to document management and collaboration, these systems became also more sophisticated in other aspects. In the beginning, such products were installed in an office environment. Their architecture followed the classical client-server principle presuming that clients and server reside within a local network. Today, workflow management systems provide different versions of client software as well. And, they support different communication protocols to communicate beyond the local network. Moreover, different workflow management systems can interoperate with each other to support federations of workflow management systems. This scenario becomes useful, if different workflow management systems serve different organisational or functional needs but still need to interoperate. The result of these developments is that workflow management software today represent versatile systems that facilitate processes in various application cases.

Among the different products and developments, the WfMC has standardised the characteristics of workflow management systems in a reference model [45]. The WfMC represents a non-profit organisation that tried to coordinate advances and developments in the workflow area. Apart from the architectural advances, the development made progress in the area of modelling workflows, which is the focus of the next section.

2.2.1 Modelling Workflows

Since the development of workflow management systems begun, most vendors have provided their own workflow modelling language. And up to today, the community has the choice between many proposals for workflow modelling by different (industry) organisations, software vendors and research groups.

Van der Aalst et al. have introduced a set of product- and vendor-independent patterns that discusses and compares the structural characteristics of the different workflow modelling proposals [146]. Examples of such patterns are different fork-conditions if the workflow splits into two sub-flows. Another example is the capability of a workflow management system to process multiple instances of a workflow at once. As for orientation, the analysis of workflow management systems conducted by van der Aalst et al. based on the workflow patterns covers about 15 different workflow management systems each with different workflow modelling capabilities [146]. There are more products available, but those 15 can be considered as the group of popular ones.

Besides workflow management software, vendor-independent approaches exist as well. Such an effort is represented by the XML Process Definition Language (XPDL, [87]) published by the WfMC. The XPDL did not convince many software vendors to be used [143]. However, contrary to other commercial proposals, the XPDL represents an effort independent of a particular vendor.

The XPDL serves as a part of the workflow reference model of the WfMC, because it represents a reference implementation for the process definitions. Based on this interface, the authors of the XPDL have created a model process definition on a meta-level. The authors admit that this meta-model will likely not cover every concept that is found in all the available workflow software products, modelling methodologies and modelling languages. However, their hope is that all the other players in the field of modelling workflows accept their model as the minimum consensus. Based on the meta-model of XPDL, an XML representation exists. The authors have chosen XML because of its wide support among different computing platforms. As a consequence, XPDL was intended to serve as a platform-independent description language that allows sharing a workflow description between workflow modelling tools and workflow execution environments as well as the interoperation of different workflow execution environments.

XPDL supports both workflows that require the interaction between human and machines, as well as entirely automated processes, by supporting concepts for invoking external services and execute local software applications. As for the second version of XPDL, which was released three years later in 2005, the authors propose XPDL also as the XML notation format for serialising graphical models using the BPMN [118]. Moreover, the WfMC has changed the used terminology and explains that XPDL serves as language for modelling business processes whilst the term workflow slips into the background. These signs indicate that WfMC and BPMI try to merge their efforts into one common consensus. Looking at the structural modelling elements, XPDL offers both elements to define block-like structures, a block-activity, as well as elements to define a graph of activities, using the concept of an activity-map [118].

Besides these efforts, research work covers also the modelling and specification of workflows. Most approaches in the research area consider the application of event-driven process chains, or formal calculus or (high-level) Petri nets as a foundation for modelling workflows. A Petri net, named after its inventor Petri, is a convention for modelling and specifying discrete events of dynamic systems. Petri nets, also named Place-/Transition-Nets (P/T-Nets) have been applied for different scenarios such as the specification of telecommunication protocols, or in business applications such as description of a logistic chain. Petri nets were introduced by Petri in the year 1962. Since then, many extensions and applications were introduced in literature. Today, the ISO covers Petri nets as an industry standard [56].

Janssens et al. have introduced an analysis of existing workflow modelling efforts that use Petri nets [67]. Their analysis covers twelve different major contributions that have covered research issues of modelling workflows with Petri nets or Petri net variants. The main reason for using Petri nets for modelling workflows is that they offer, besides a graphical notation, a formally defined semantic description of the elements. This allows the application of formally proven analysis techniques (cf. Janssen et al. [67] and van der Aalst [142]). Among these contributions, van der Aalst et al. have defined a popular Petri net variant that they name *Workflow Net*. The workflow net represents a convention for modelling workflows with Petri nets [148]. Based on this work and the workflow patterns analysis mentioned at the beginning of this section, van der Aalst et al. have also introduced a workflow modelling language named Yet Another Workflow Language (YAWL, [149]). YAWL represents a language proposal which extends the concepts of Petri nets in order to support the workflow patterns to their full extend, while keeping a formal foundation that allows the anticipated verification on modelled workflows.

2.3 Workflows versus Business Processes

The previous sections have introduced two concepts, namely business processes and workflows, which seem to have many issues in common. Apart from their origin and their purpose, which showed different roots in the beginning, both have many research questions in common. For example, in order to specify either processes or workflows, common approaches exist that use notation techniques based on Petri nets or event-process-chains. Considering the execution side, both share common challenges regarding distributed executions, fault-tolerance or optimisations. The resulting question is: In what aspects are these fields different? The reference model of the WfMC defines a workflow as a business process facilitated by computer systems. This definition, which has been stated in many research works since 1995, says that any computer system processing business processes represents in fact a workflow management system.

The two terms are used synonymously sometimes, and many publications do not mention any difference between the two, suggesting that using either the one or other is based on historic reasons. However, two main communities exist, one representing the workflow side and the other representing the business process community: the WfMC and the BPMI. In a retroperspective of the workflow reference model published by the WfMC, the authors acknowledge the growing momentum of business process management [46]. The WfMC explains that the evolution of the involved technologies and techniques of workflow management systems meet today the concept of business process management: Business process management is supposed to cover accounting issues and the management of resources. Business processes include both machine and human activities. Consequently, the WfMC proposes using their original reference model as the foundation for a future reference model that covers business process management.

Members from the BPMI promote a different view on the relation between the management of workflows and business processes. Smith and Fingar have initiated a discussion by publishing the statement that a workflow is purely concerned with process description [122]. According to their view, a comparison between workflow management systems on the one hand, and business process management systems on the other hand, reveals that workflow management systems show a number of disadvantages that make them less suitable for the new demands of today's enterprises. They explain that workflows are more static (i.e. application-dependent) and do not support modifications of the business process. Common workflow management systems do not cover the concepts needed for the majority of business processes and workflow management systems do not refer to a common model of workflow. This approach of the BMPI has received a response, which has motivated the authors of the original article to publish clarification [121]. What remains is that, according to the BPMI, workflow represents just one aspect of what is covered by business process management. In addition to workflows, business process management covers the integration of different computer systems as well the non-computerised parts of business processes.

When it comes to the application of the terminology and the referring languages, the main disadvantage is that a clarification must be made between what is different and what is only claimed to be different. Considering the presented overview, the conclusion is that workflow and business processes move together to become the same. Apparently, the difference between both as they have evolved until today results from their different origins – on the one side stands a vision about automating the paperwork in an office environment and on the other side there is the optimisation of what happens in companies from a business perspective. The underlying problems, such as the expressiveness of the modelling languages, how verifications can be applied or which graphical modelling language is the most efficient, appear to be similar for both fields. Considering the workflow definition of the WfMC, Figure 2.1 is modified by considering the relation between workflows and business processes as shown in Figure 2.2.

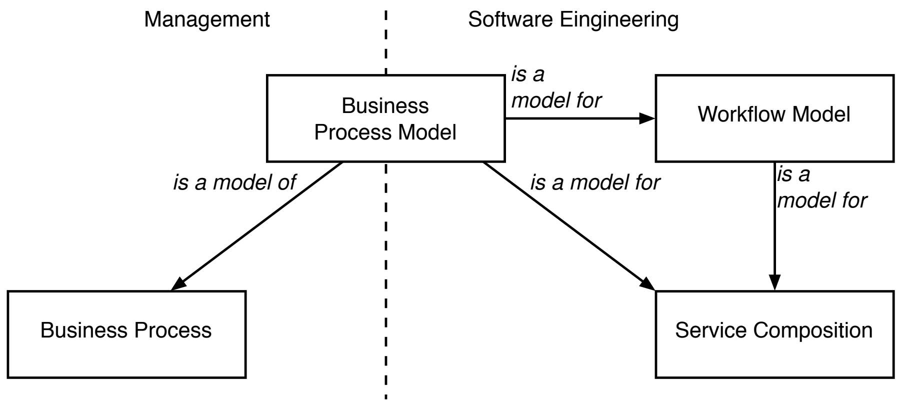


Figure 2.2: The roles of business process and workflow models.

2.4 Realising Business Processes and Workflows

After the clarification of the relation between business processes and workflows, this section summarises how these two fields can benefit from an SOA and the composition of services. Different research work has analysed the characteristics of business processes and workflows with regard to service compositions. These research works also provide several proposals on how to create business process (and workflows) in an SOA. The main points in favour of using services and compositions of them are:

- **Technology Independence.** (cf. Hunhs and Singh [47], Papazoglou [104], and Yang [154]) The basic idea behind promoting the SOA architecture is to establish a middleware that ties together functionality offered by different systems, regardless of their hard- and software. Services should use common interaction protocols as well as common interface descriptions and platform-independent types. With Web services as an implementation of an SOA, it appears that is becoming reality: Both, service consumer and service provider can run on different hardware as well as be implemented in different programming languages. However, to a certain extent all the different players can interoperate.

  The role of Web services as the middleware for integration goes even further: Solutions exist that encapsulate other middleware platforms, which were originally designed to provide independence from language and hardware as well. Thus, software products exist that use Web service technologies to provide a common interface technology for existing technologies, e.g. CORBA, mainframes, Java component frameworks etc. [109]. Heterogeneous IT systems are standard in large businesses, because these businesses have usually started the integration of computer systems at early stages, when the interoperation of computer systems was not a major concern. Thus, these different systems must be integrated into one common platform in a potential business process reengineering effort.

- **Implementation Neutrality.** (cf. Dijkmann and Dumas [22], Hunhs and Singh [47], and Yang [154]) One of the main differences between service-oriented computing and distributed object computing [4] is that services usually provide one aggregated interface that might use objects. However, services hide the structure of the software that provides the functionality behind the interfaces. This leads to an encapsulation of underlying objects. As a consequence, changes applied to the particular implementation do not necessarily result in a change to the service interface and therefore do not require changes when establishing the interoperation between service requester and provider. The previous point and this point are covered by the concept of *access transparency* defined by the RM-ODP (cf. [50, section 8.1.2]).

- **Location Transparency.** (cf. Bolcer and Kaiser [10], and Papazoglou [104]) Location transparency provides an abstraction of the physical location where the service is provided in the sense of the RM-ODP (cf. [50, section 8.1.2]). Today's SOA implementations support Internet protocols and consumers in the Internet can invoke services across the local computer, the local network or the local organisation. Although an organisation might not consider invoking just any service that is available somewhere in the world, using Internet protocols offers a greater level of flexibility than using runtime environments that run on specific computers or a federation of computers.

- **Loose Coupling.** (cf. Hunhs and Singh [47], and Papazoglou [104]) A common definition for loose coupling does not exist. Usually, in an SOA, loose coupling means that a service consumer knows what kind of service is required during design time. However, the binding ("coupling") to a real available service takes place during run-time. This does not necessarily imply that a service consumer will bind services only during run-time. However, loose coupling enables service consumers to revise existing bindings during run-time when necessary.

- **Process Orientation.** (cf. Bolcer and Kaiser [10], and Dijkmann and Dumas [22]) Modelling business processes also leads to a description of required tasks and a specification about the execution order of the tasks. A composition of services can provide a business process using an SOA environment when every task can be provided as a service. This presumes that, for example, human interaction is not required. The orientation to the service interfaces stands in contrast to what a set of objects would provide in a distributed object computing environment. The resulting implementation neutrality conforms to the business process paradigms: Like services, business processes should have a defined input and output, whilst the implementation of each task in the process becomes secondary.

This list of points represents the motivation that already a couple of products have been introduced already be software vendors to develop business processes with forming service compositions. Examples from the Software industry are the Oracle BPEL Process Manager [117], or the WebSphere Integration Developer by IBM [85].

2.4.1 Modelling Service Compositions

Based on the modelling languages for workflows and business processes that have been used so far, different languages have been proposed (and also already been mentioned) that directly model compositions of services. These languages do not focus on user interactions and consider mainly automated processes. Most languages provide direct support for Web services as this represents the major SOA implementation used today. In this context, support means that languages refer to

WSDL or SOAP or other specifications from the field of Web services. Currently, many different proposals are available to describe compositions of Web services; similar to what can be observed in the field of workflow or business process modelling. To provide a clear view of the different languages, the following three groups for these languages are proposed:

- **Abstract level languages, Level 1.** Languages that are primarily intended to describe an abstract process with activities. In this case, services might provide these activities but the reference to concrete services is not mandatory to make the description complete. If available particular services are not mentioned, such a description describes the composition on an abstract level and handles the activities or involved services as black boxes with technology-independent interface descriptions.[2]

- **Concrete level languages, Level 3.** On the concrete level, the description involves particular services and a description of each particular service. The main issue when talking about services is that a service does not represent only an atomic, stateless operation, but provides a set of different operations. Depending on the complexity of the service and the composition, a specification of a composition must involve this aspect. These languages do not focus on service compositions, because they can be also applied for describing the interaction between individual parties on a technical level.

- **Languages covering both levels, Level 2.** Some proposals clearly focus on defining the interoperation between service exporter and importer and other proposals focus more on modelling the process and its activities. As a third group, some languages are right in the middle of both, not showing a clear process modelling focus and also not a clear interoperation focus.

It must be noted that this categorisation does not provide a formal basis nor an argumentation like: If concept x is found in a language then it belongs to level y. This categorisation only has the purpose of giving a rough orientation. The two main levels 1 and 3 have their analogies to the items mentioned in Figure 2.2 which has shown the relation between the business process model and the service composition. In accordance with this figure, an updated version 2.3 shows that level 1 modelling languages refer to the business process model, while level 3 languages are used to express service composition models. The work of Ouyang et al. discusses the transformation between process models and models of service compositions by considering BPMN and UML activity diagrams on the process side, and BPEL for modelling the service compositions [101].

Table 2.4.1 lists a selection of proposals that were mentioned in the literature as *Web service composition languages*, with their acronyms and their proposed

[2]The term *abstract* is used in the same sense as in the ISO RM-ODP: *The process of suppressing irrelevant detail to establish a simplified model, or the result of that process* [51, section 6.2]

categorisation. In addition, Figure 2.4 shows a chronological overview of their introduction dates. The two BPEL proposals, WSFL and XLANG, were mentioned already in the Section 2.1.2 about business process modelling. The focus of these languages lies on the specification of a business process by using available Web services. A specification using one of these languages forms a composition of services that is ready for execution. Thus, these candidates can clearly be named Web service composition languages and would fit into the second group.

Regarding the languages at the first level, Table 2.4.1 mentions XPDL, BPML, and the Business Process Specification Schema (BPSS). All three offer language elements to directly support the invocation of Web services. However, the invocation of a Web service is not required in the XPDL for the realisation of an activity [118, section 7.1.4.1]. The BPSS clearly has the smallest focus on supporting compo-

L.	**Acronym**	**Full Name**, Reference **Supporting Parties**, Remarks
1	XPDL	XML Process Definition Language [118] WfMC, contributing authors were from Global 360, FileNet, Staffware/TIBCO, Prozone and Fujitsu Software
1	BPML	Business Process Modelling Language [29] BPMI, the specification mentions only one contributing author from Intalio
1	BPSS	Business Process Specification Schema [29] *(Part of the ebXML Suite)* UN/CEFACT, an United Nations Body for Electronic Trade and an OASIS Technical Committee, including members from Cyclone Commerce, Fujitsu, SAP AG and Sun Microsystems
2	WSFL	Web Services Flow Language [80] IBM, moved into the BPEL4WS proposal
2	XLANG	subtitled ”Web Services for Business Process Design” [129] Microsoft, merged with the BPEL4WS proposal
2	BPEL4WS	Business Process Execution Language for Web Services [31] IBM, Microsoft and BEA, merged with the WS-BPEL proposal
2	WS-BPEL	Web Services Business Process Execution Language [126] An OASIS Technical Committee involving 18 industry parties, among them BEA Systems, IBM, Microsoft, Oracle, Sun Microsystems, SAP AG
3	WSCI	Web Service Choreography Interface [30] W3C Note submitted by BEA Systems, Intalio, SAP AG und Sun Microsystems
3	WS-C	Web Service Choreography [14] W3C Working Group, continuing with the WSCI proposal

Table 2.1: Overview: Web service composition languages.

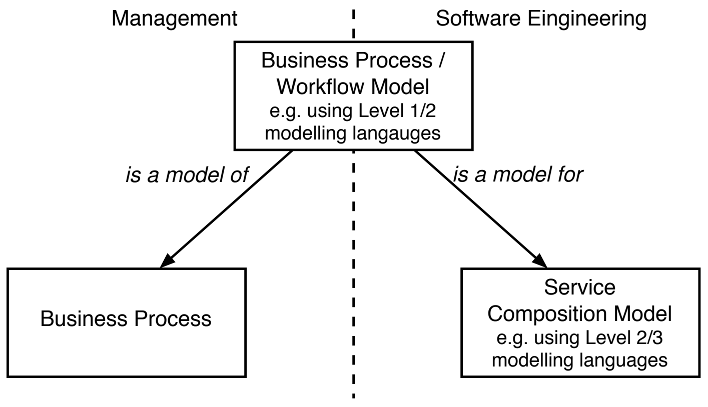

Management
Software Eingineering
Business Process / Workflow Model
e.g. using Level 1/2 modelling langauges
is a model of
is a model for
Business Process
Service Composition Model
e.g. using Level 2/3 modelling languages

Figure 2.3: The relation between business process, workflow and service composition models.

sitions of services. The BPSS is a part of the ebXML suite, which supports establishing agreements to facilitate electronic businesses on an inter-organisational level. The motivation for this effort is to provide a specification for helping developing countries to participate in electronic commerce without being dependent on technologies by particular vendors. The ebXML is maintained by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and a Technical Committee at OASIS. The first release was submitted in 2001. At that time, compositions of services or an SOA environment was not mentioned at all.

There are two reasons why this work considers BPSS as member in the circle of the composition languages: At first, it has been mentioned in some publications as a language for expressing compositions of services (cf. [120, 143, 152]), thus it is useful to clarify its relation to service composition and the other languages. However, considering the BPSS as a composition language must take into account that the specification is independent from particular SOA technologies. To provide the ties to concrete SOA implementations like Web services, separate specifications named Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) are proposed. A BPSS document may refer to a CPP or CPA to provide a description of the involved services. For example, a BPSS document could refer to a WSDL interface description of a Web service. However, since the BPSS has clear a focus on enabling international electronic trade, it is not the preferred candidate to develop business processes in an average company.

The Web Service Choreography Interface (WSCI) represents the foundation for the third group. Considering a Web service, it usually does not provide atomic single operations to be invoked but rather a set of operations in a specific order while capturing the state of the invocation. The WSCI proposal directly focusses on specifying the needed message flow between Web services resulting from their offered operations [30]. Consequently, the authors of BPML mention explicitly

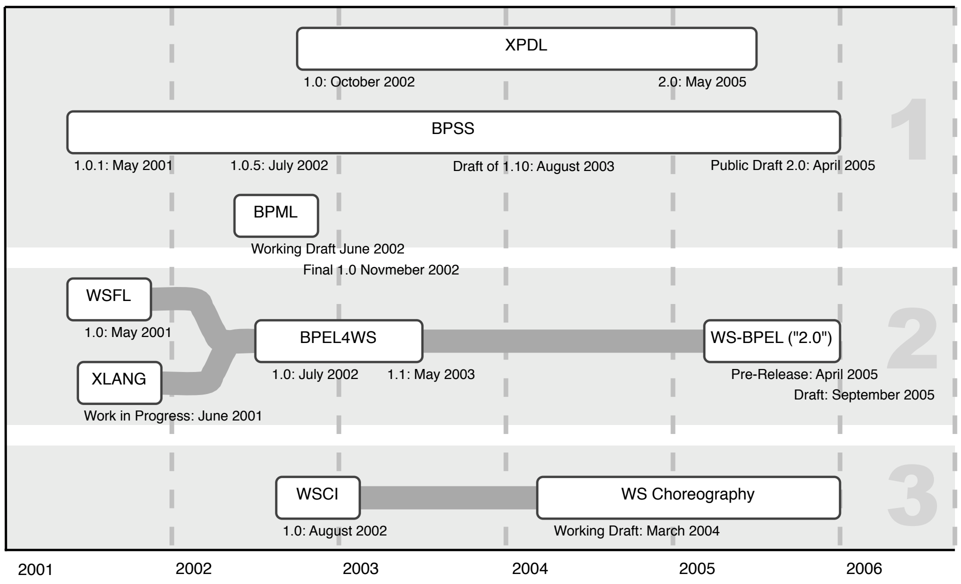


Figure 2.4: Release dates of service composition languages.

that WSCI and WSDL specification are complementary [29, section 1.2]. The WSCI has been submitted to the W3C as a technical note in 2002. Since then, this proposal has not received any further updates. In beginning of 2003, a W3C working group named Web Services Choreography has begun its work to take the WSCI proposal as input. Based on the WSCI, the working group continues to work on this topic [14]. By choreography, the working group refers to the "characteristic of describing linkages and usage patterns between Web services". The working group uses the term choreography in a similar manner as other proposals use collaboration, conversation, coordination or orchestration. Currently, the working group has released several documents which are under further development.

A criterion for comparing the different languages is not only their application but also their expressiveness. As expressiveness, not the expressive power but the convenience to express particular aspects is considered. Such a comparison has already been conducted by Wohed, van der Aalst et al. [151]. In their work, they have presented a comprehensive analysis on the structural capabilities of languages to specify Web service compositions for BPEL4WS, XLANG, and WSFL. Based on this research work, further work has been published covering BPML and WSCI [150] and XPDL [143]. The general result of this work is an analysis on which composition language directly supports which (structural) workflow pattern. By using the patterns, Wohed, van der Aalst et al. give a detailed explanation about the languages' capabilities and problems and – more importantly – they also explain how previously introduced workflow modelling languages have coped with expressing specific aspects. Thus, their summarising conclusion is that Web service composition languages do not offer a greater level of expressiveness when compared to workflow modelling languages. However, it must be noted that their primary purpose is to facilitate the development of service compositions. Consequently, Figure 2.3 can be extended as shown in Figure 2.5: The business process model represents the interface between the organisational side and the software engineering side. On the software engineering side, the model of a service composition is used as a model for the required development tasks. The development of compositions will be the topic of Chapter 7.

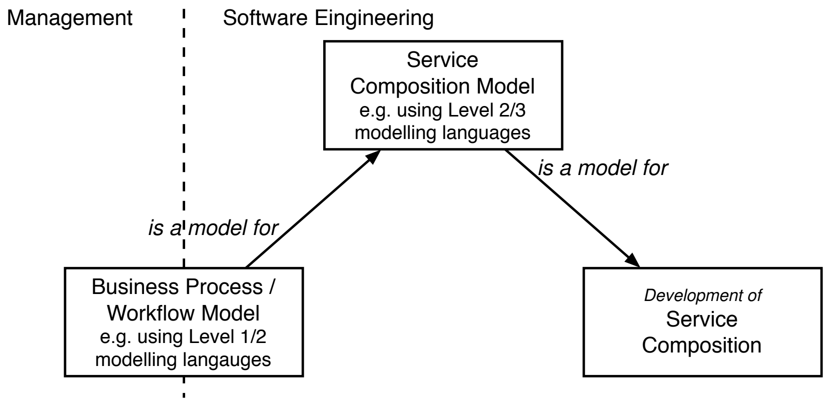

Management
Software Eingineering
Service Composition Model
e.g. using Level 2/3 modelling languages
is a model for
is a model for
Business Process / Workflow Model
e.g. using Level 1/2 modelling langauges
Development of
Service Composition

Figure 2.5: The role of the service composition model.

Despite of the clarifying statement of this particular research group, the development of the different composition languages is ongoing. The OASIS-driven WS-BPEL effort has currently the largest attention in the area of service compositions. It considers the widely accepted Web services standards and gathers many industry players providing SOA-based products, such as IBM, Microsoft, Oracle, SAP AG and Sun Microsystems, together. Besides, the WS Coordination working group shows ongoing activities. Taking into account that existing documents published so far have only draft status, they indicate that there is more to expect. Apart from these languages, which directly focus on Web services, this chapter has introduced other proposals that focus on different purposes but can be considered for describing service compositions as well.

Chapter 3

Quality-of-Service in Service Compositions

The introduction has already presented a general definition of the QoS which cites the ISO 9004 standard [127, section 3.5]. Every system is designed to provide a certain functionality which represents the functional behaviour of the system. For example, if a service that performs a mathematical calculation on a given number is considered, the description and definition of this calculation would cover the functional behaviour of the system. Nevertheless, a system shows also a non-functional behaviour. If the example of the calculation service is considered, its non-functional behaviour can cover the consumption of resources, the time needed for an operation, or the precision or the delivered result. Such nun-functional issues show a relation to the concept of QoS, which will be elaborated further in Section 7.2.1 of the chapter that discusses the application of the QoS-based selection when developing service compositions.

In literature, the two, QoS and non-functional behaviour, sometimes appear as synonyms (e.g. in the WSA from the W3C [11]), whereas other publications mention them with clearly separated meanings. In this work, the concept of QoS is subsumed by the concept of non-functional aspects, which is also the view of the UML Profile for *Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms* (the “UML QoS-Profile”, [99]). In accordance with this recommendation, the QoS describes quantifiable non-functional characteristics of a system.

The concept of QoS covers a wide area of aspects. Today, the QoS refers mostly to the field of telecommunication networks. Besides the functionality that a telecommunication network provides – the transmission of data – QoS characteristics are inherent properties. Such characteristics can be the delay that occurs when data is transmitted, or the bandwidth representing the amount of data that can be transmitted within a given time. A telecommunication network can be seen as a service that transports data from one point to the other. The concept of QoS can be easily transferred to the field of the SOA. Looking at the RM-ODP, the standardisation efforts to define what QoS means in a open distributed processing system have been published as a designated proposal [54], which represents the intended sixth part of the RM-ODP. The currently available normative reference about QoS by the ISO is the *Quality of Service Framework* published in the Information Technology section [55]. From this reference, this work considers the following selection of definitions:

> **QoS characteristic:** A quantifiable aspect of QoS, which is defined independent of the means by which it is represented or controlled.
>
> **QoS establishment:** The use of QoS mechanisms to create the conditions for some system activity, before that activity occurs, so that a desired set of QoS characteristics is attained.
>
> **QoS mechanism:** A specific mechanism that may use protocol elements, QoS parameters or QoS context, possibly in conjunction with other QoS mechanisms, in order to support establishment, monitoring, maintenance, control, or enquiry of QoS.
>
> **QoS data:** QoS information other than QoS requirements, e.g. warnings, QoS measures and information used in QoS enquiries.
>
> **QoS requirement:** QoS information that expresses part or all of a requirement to manage one or more QoS characteristics, e.g. a maximum value, a target, or a threshold; when conveyed between entities, a QoS requirement is expressed in terms of QoS parameters.
>
> **QoS parameter:** QoS information that is conveyed between entities as part of a QoS mechanism; parameters are classified into requirement parameters and data parameters; the information conveyed may relate to one or more QoS characteristics.
>
> **QoS information:** Information related to QoS: [...] It is classified into QoS requirements (if it expresses a requirement for QoS) and QoS data (if it does not).

There are other recommendations and specifications from industry bodies and non-profit organisations, such as the UML QoS-Profile [99], which was mentioned already in this work. The authors have taken both documents from the ISO [55, 54] in account for the specification of this recommendation. In addition to the documents of the ISO, the UML QoS-Profile provides also definitions for the terms "QoS characteristic" and "QoS category", which are considered as the basis for this work [99, page 10]:

> **QoS characteristic:** A QoS characteristic represents a quantifiable characteristic of services. A QoS characteristics is specified independently of the elements that they qualify. A QoS characteristic is the constructor for the description of non-functional aspects like: latency,

throughput, capacity, scalability, availability, reliability, safety, confidentiality, integrity, error probability, (...)

QoS dimension: QoS dimensions are dimensions for the quantification of QoS characteristics. QoS characteristics can be quantified in different ways (e.g., absolute values, maximum and minimum values, statistical values). For example, the latency of a system can represent an end-to-end delay of the invocation, the mean time of all invocations, or the variance of time delay. (...)

QoS category: When the number of QoS characteristics is large, or they are especially complex, some mechanisms for grouping are required. Some examples of general groupings of quality attributes are: i) performance: performance references to the timeliness aspects of how software systems behave. ii) dependability: (...)

3.1 Exchange of Quality-of-Service Information

Based on these definitions, a model is designed that covers the flow of QoS information when trading services. The first chapter has defined that the retailer processes the QoS information covering compositions. However, it remained unclear how a retailer will obtain such information and which role is represented by the broker in such a setup. Different methods exist to exchange QoS information as introduced by the RM-ODP, the trading specification and the Business Model of the TINA-C architecture [156]. The trading specification defines that a service offer includes so-called service properties which subsume the concept of QoS in this context [53, section 7.2]. The TINA-C Business Model defines a broker that can involve (QoS) attributes about a service as well [156, section 2.3]. However, the architecture description of the TINA defines that the QoS is negotiated between the retailer and the 3rd party service [156, section 3.5]. Telecommunications services are regarded as volatile and subject to change to every interoperation between importer and exporters. Thus, a repeating re-negotiation between retailer and 3rd party provider is preferred to better cope with this volatility.

The currently available proposals for processing QoS in open distributed processing propose a separate, dedicated broker that only processes QoS information. This is a QoS broker component that is added to the standard setup of an SOA. A dedicated QoS broker has the advantage that such a software can also perform QoS monitoring and dynamically trigger QoS improvements if necessary. Research work in the mid 90s proposed broker architectures to provide a QoS-aware service interoperation in the telecommunications domain. The work of Nahrstedt and Smith represents an example of such an approach [94]. In their work, the QoS broker is arranged as an “application subsystem”, a tier between the service importer and the underlying communication tiers, to implement a QoS mechanism and co-ordinate the flow of QoS information.

Frøhlund and Koistinen have discussed the QoS in open distributed processing systems [35]. Their contribution is focussed on the specification of QoS information and its representation during runtime. Then, a so-called QoS-based trader establishes the interoperation between service exporter and importer. Apart from their abstract specification, their work also explains the application of their findings in a CORBA environment. A couple of other research groups have worked on the QoS-based trading to the CORBA environment. Among them is the work of Cardoso et al. [16] which covers the implementation of QoS-aware middleware facilities to provide the optimisation of QoS in workflows.

Besides the workflow domain, the research in the mid-90s discusses multimedia applications in telecommunication networks. In this scenario, there is a strong demand to provide real-time characteristics for the transmission of audio and video data. Their transmission requires a constant data rate with minimum delays. Otherwise, the service can become useless. For example a telephony service is useless when the delay resulting from the transmission is too long. Aurrecoechea et al. [3] have discussed the issues of such applications and their underlying QoS-aware infrastructure in a survey that summarises the different approaches in this field.

3.1.1 Quality-of-Service in a Service-Oriented Architecture

A software infrastructure that provides telecommunication services differs from an SOA. Contrary to the telecommunication domain, the scenario is as such that existing software and components are integrated into one common invocation middleware that allows to the interoperation of the different applications involved in this middleware tier. The telecommunication domain is influenced by the ISO Open System Interconnection Model (ISO-OSI, [57]) that denotes a tiered architecture in which one tier offers its functionality as a service to the next upper tier.

The functionality of telecommunication services is concerned with the transmission of data, while services in an SOA can provide different functionality that include telecommunication aspects (e.g. a service that offers to send an e-mail). The QoS that covers real-time characteristics is not as often considered in an SOA as for telecommunication services. The main reason is that two of the SOA characteristics imply just the opposite: *a)* the loose coupling and *b)* the use of Internet protocols (cf. Jaeger and Mühl [60]). Because an SOA usually provides a loose coupling characteristic, the binding operation requires an unpredictable amount of time. Internet protocols are known for their robustness. However, they do not guarantee that a connection can be established and can be hold with a constant QoS level. Thus, for some soft real-time applications such as telephony and video-broadcast, extensions to the Internet protocols are used which provide reservation and signalling functionality. In summary, the support for QoS is not as present in an SOA as it is for telecommunication systems.

Quality-of-Service with Web Services

Different research groups have worked on how to cover the QoS in an SOA or in the field of Web service infrastructures. A topic in this field is the discussion about QoS-based brokers specifically for trading Web services [88, 115, 116, 131, 136]. All works share the idea that a separate dedicated broker processes the QoS information. Either service providers must submit their QoS parameters as a part of their advertisement, or the QoS broker obtains the QoS information from a monitoring process. The difference between the mentioned works lies in how the QoS information is modelled and represented during run-time.

The mentioned research works have in common that a shared agreement about the QoS information is required. This agreement is necessary to enable all involved parties to exchange this information. Wang et al. and Tian et al. propose each a proprietary XML-based language [136] for the specification of QoS information that should be used when communicating with a QoS broker. The work of Maximilien and Singh puts the standardisation of used QoS concepts into an abstract taxonomy that appears independent from notation formats such as XML or an XML Schema [88].

In addition, some approaches exist that extend the common Web service repository specification UDDI with the capability to process the QoS rather than establishing separate facilities [2, 78, 112]. Besides the research work, the UDDI implementation as a part of Microsoft's server platform allows extensions in order to support the processing of QoS information [93]. The common approach regarding UDDI repositories is to extend the data model with custom definitions of certain QoS characteristics. The specification allows extensions to the standard data structure by supporting the addition of data models, so-called tModels, using the standard API of a UDDI repository. When querying a service, the query can include QoS requirements that refer to the added QoS concepts. In this scenario, a UDDI repository represents the broker role that also processes QoS information. Then, the broker is involved when establishing the interaction between importer and exporter.

Besides the broker-oriented approaches, proposals exist for Web services to negotiate the QoS directly between the importer and exporter. Such proposals consist of a negotiation protocol and a QoS specification language. Menasce has discussed common issues of this setup in conjunction with Web services [90]; the basic idea is that the importer negotiates a Service Level Agreement (SLA) with the exporter. Both parties define in this SLA the QoS that the service exporter must provide. Examples for SLA languages are the Web Service Level Agreement Language by IBM (WSLA, [84]), the Web Service Offerings Language by Tosic et al. (WSOL, [133]) or a similar proposal to express SLAs by Sahai et al. [113]. The authors of the WSOL have also published a Web Service Offerings Infrastructure (WSOI, [132]) which explains the use of the WSOL. Generally, the WSOI is not concerned with the concept of trading services, but with monitoring and managing services during the run-time. Consequently, the WSOI provides an extended server infrastructure to support QoS parameters mainly residing at the service exporter. A separate component for the QoS-based trading of services using the WSOL is not offered by the WSOI.

Table 3.1 summarises the different main approaches which are grouped into three main architectural characteristics: *1)* the integration into existing service brokers, *2)* the provision of a separate broker architecture and *3)* the direct and individual negotiation. Among these works, the problem exists that the service importer must have the same understanding about the used QoS characteristics and as the service importer and the broker.

Research Group	**Remarks**
Integrated Broker, extending a UDDI repository	
Ali et al., 2003 [2]	Discussion about a general extension to UDDI that supports properties of service advertisements. Such properties can be used to denote QoS parameters.
Ran, 2003 [112]	Discussion about a model for the integration of QoS information into a UDDI repository.
Lee, 2003 [78]	Discussion about the trading of Web services to form service compositions; covers the QoS information by extending a UDDI repository.
Dedicated Broker, separate from a UDDI repository	
Wang et al., 2004 [136]	Introduction of a framework that adds the support of QoS in an SOA, as used in a large enterprise. Involves also QoS-based trading.
Degwekar et al., 2004 [115]	Discussion about expression of (QoS) constraints using the WSDL. Proposes to use a separate broker for querying QoS parameters of services.
Tian et al., 2004 [131]	Discussion about a framework that provides a QoS-based broker for the selection of Web services.
Maximilien, Singh, 2004 [88]	Discussion about issues of service selection performed by autonomous agents.
Yu, Lin, 2005 [157]	Broker for dynamic integration of Web services and adaptation to ensure the QoS during runtime.
Direct Negotiation	
Tosic et al., 2002 [133]	Language to specify QoS requirements and parameters; later work covers also management issues.
IBM, WSLA, 2003 [84]	Language to specify QoS requirements and parameters; specification covers also negotiation scenarios and strategies.
Menasce, 2004 [90]	Discussion about general QoS and Web services topics while promoting the SLA-oriented approach.

Table 3.1: QoS-based trading in the domain of Web services.

As mentioned above, Maximilien and Singh [88] propose covering this issue with a QoS ontology to provide a common agreement when performing a QoS-based trading. Their approach discusses two major ontologies: One is called “QoS upper ontology” which covers general terms such as “Measurement” or “Attribute”. A definition for such terms is also provided by the QoS-framework published by the ISO [55] or the UML QoS-Profile from the OMG [99, section 8]. For a more detailed view, a “QoS middle ontology” defines concrete QoS characteristics and a hierarchy of QoS categories. A similar hierarchy is also covered by the work of the ISO and the OMG [99, section 10]. A “QoS lower ontology” is foreseen to include all application and domain specific details. Section 3.2 will discuss the QoS characteristics that are relevant for such an ontology.

3.1.2 The Role of the Retailer

The previous section has introduced three basic patterns of processing QoS in the field of Web services. These can be abstracted for general application in an SOA: *a)* using a combined broker, *b)* using separate brokers or *c)* direct negotiation between service importer and exporter. Based on these three main patterns, the processing of QoS information from the view of the retailer can be discussed. The role of the retailer is special because he acts as a service importer when importing the services to form a composition and he acts as a service exporter when he provides the composition to the consumer. All these patterns have in common that a service exporter provides his QoS parameters while the submission of QoS requirements by the service importer is regarded as optional. The three main patterns are as follows:

- **Direct negotiation.** The first pattern reflects the proposal of the TINA, where the QoS information is processed individually and decentrally. In this scenario, a service importer agrees with the exporter on a service level agreement. In this agreement a requirement from the importer is optional. Regarding the QoS requirements, two different cases are possible:

  - The retailer receives the requirements and then forms the composition, i.e. he selects the set of services that result in a composition that meets the given requirements.
  - The retailer has formed a composition that meets internal QoS requirements. Then the retailer advertises the composition for export including the corresponding QoS statements. Then a consumer can express his requirements to see whether these requirements are met by retailer or not.

  This setup covers the optimisation of the QoS of the composition with respect to the QoS requirements of the consumer. The retailer performs the QoS-based selection which is a part of the trading process. Figure 3.1 summarises this setup.

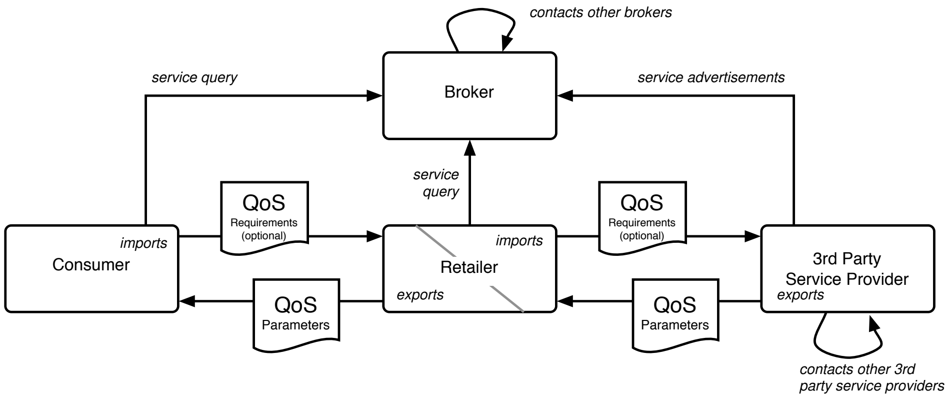


Figure 3.1: Flow of QoS information without involving a broker.

- **Integrated broker.** In the second pattern, a broker component is extended to also process QoS information when trading services. In the field of Web services, this setup represents the approaches which provide an extension to existing UDDI repositories. Contrary to the first pattern, the broker processes the QoS parameters provided by a service exporter who advertises his service. When a trading process takes place, the broker can optionally process QoS requirements by the service importer.

  Since the retailer represents a service importer and exporter simultaneously, he first queries the broker when forming compositions, optionally with given QoS requirements. At this point, the QoS-based trading can be performed by two parties:

  - **Retailer-sided.** The retailer queries the broker for the individual services and receives a set of candidate services from which he must perform the selection based on QoS criteria. In this case the trading process is split between the broker and the retailer.
  - **Broker-sided.** The retailer submits a description of the composition to the broker. Then, the broker can perform the trading process and the QoS-based selection of services. Then the broker would return a list of selected services.

  After the retailer has formed the composition, he can advertise the composition as a new service to export to the broker with the resulting QoS parameters. Then a consumer can query the broker to find appropriate services that comply with given QoS requirements. This setup is represented in Figure 3.1.

- **Separate QoS-broker.** In the third pattern, a dedicated broker performs the

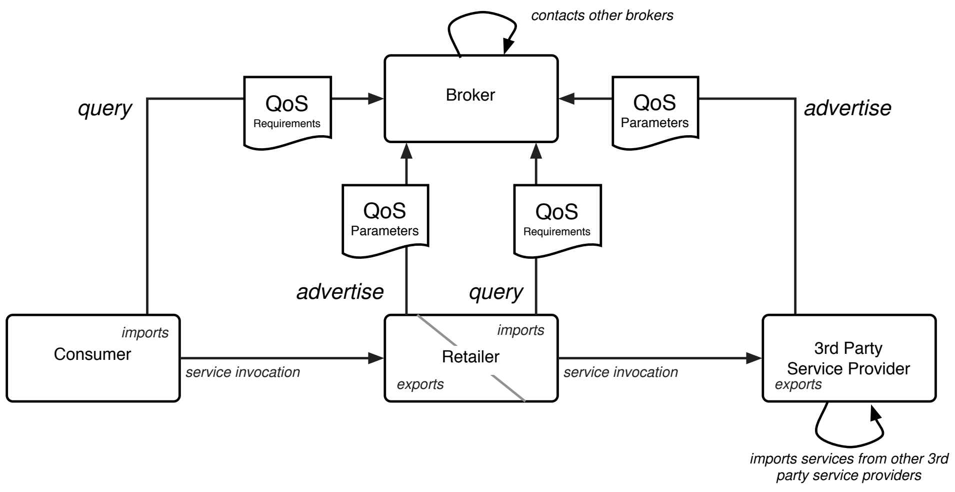


Figure 3.2: Flow of QoS information involving a broker.

QoS-based selection of the trading process. From the research presented in the previous section, there is no rule on when to extend an existing broker facility and when to establish dedicated structures to process QoS information for trading services. A motivation for establishing a separate broker infrastructure exists, if monitoring and QoS-based optimisation during the run-time are also required. Then, a service repository could focus on the discovery part of the trading process while a QoS-broker can also monitor the provided QoS to apply dynamic adaptations.

Figure 3.3 outlines such a setup with a dedicated broker. In this setup, the service importer queries the general broker for the desired services and receives a set of candidates. As a second step, the service importer queries the QoS of the candidates at the dedicated QoS-broker. Like in the previous case, either the retailer performs the QoS-based selection or he submits a description of the composition to the QoS-broker. In the latter case, the QoS-broker can entirely perform the trading process. After the retailer has formed the composition, he can advertise the composition as a new service at the general broker. Then he submits the referring QoS parameters to the QoS-broker.

These three patterns outline the basic arrangements in a QoS-aware SOA. Of course, other arrangements are possible. For example in a mixed configuration, the retailer accepts the direct negotiation with a consumer but queries a dedicated QoS-repository to search for the QoS parameters of the service candidates. These mixed-variations were ignored because they would not lead to additional conclusions besides those that are drawn in the remainder of this section. The following

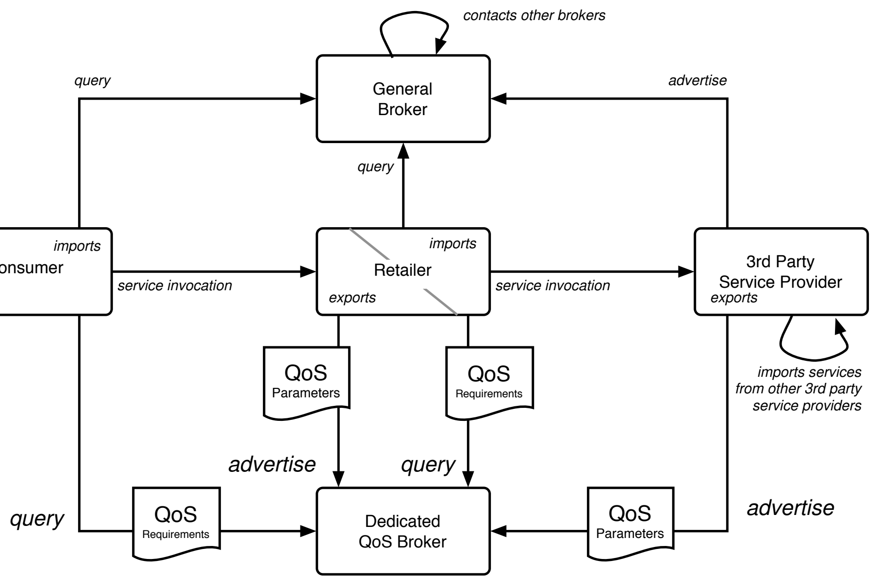


Figure 3.3: Flow of QoS information involving dedicated brokers.

two characteristics are variable in these three processing patterns when performing the QoS-based trading:

- **Consumer-driven QoS-based trading.** The first scenario, which describes the direct negotiation, describes the case that the consumer submits QoS requirements. Then, the retailer performs the selection that also involves the requirements of the consumer. In the opposite case, which is the only case supported when involving a broker, the retailer considers only own requirements and advertises the composition as a service with the resulting QoS parameters.

  In the case that QoS parameters are advertised at some party representing a broker – regardless of whether dedicated for processing QoS information or not – and no direct negotiation is supported, then the consumer-driven QoS-based trading cannot be supported: The QoS requirements will be processed at the broker and will not arrive at the retailer.

- **Retailer-driven trading.** In the first scenario, which describes the direct negotiation, the retailer entirely performs the QoS-based trading. In the two other processing patterns, either the broker or the retailer can perform the trading. The problem was outlined in the first chapter: The broker would require information about the composition in order to perform the trading. From a technical viewpoint, the trading would perform in both cases in the same way. However, from an organisational point of view two reasons could prevent this setup:

1. Performing QoS-based selection to form compositions requires special logic that is able of finding solutions for the resulting combinatorial problem. Because it is unclear whether existing and future broker technologies will actually support this functionality, the retailer would minimise his possibilities to find appropriate services.

2. If the retailer submits information about the business process to the broker, he shares his intellectual property or information of some business value. If broker and retailer do not belong to the same organisation, organisational constraints might prevent such a disclosure of information.

Based on this clarification, statements can be derived about how the given QoS concepts relate to the retailer and the other roles. This model represents an extended version of the model given in Figure 1.2 of the first chapter. The previous part has clarified why the retailer represents the predestined party to perform the QoS-based selection. On the other side, several proposals exist that feature a QoS broker – either a separate dedicated or extended existing brokers – for trading individual services. As a conclusion, the retailer processes the QoS information on a macro-perspective; he processes the QoS-part of the trading in order to form the composition. A potential dedicated QoS-broker processes the QoS on a micro-perspective, because he processes individual queries. In summary, the characteristics of the retailer are defined as follows:

The retailer

- performs the **QoS establishment** of the composition,
- features a **QoS mechanism** to support this establishment, and
- is capable of processing **QoS information** to support the QoS establishment. The QoS information are the QoS requirements of the consumer and the QoS parameters of individual services provided either by a broker or directly by 3rd party service providers.
- In case a broker exists that processes the **QoS parameters**, the retailer submits them when advertising the composition as a new service.

The consumer, a broker and the 3rd party provider shall have the following characteristics:

The client

- submits **QoS requirements**, and
- receives **QoS parameters** about the offered composition.
- In case a broker exists that processes the **QoS parameters**, the QoS information is exchanged with the broker, otherwise the client communicates directly with the retailer.

The 3rd party service provider

- submits **QoS parameters**.
- In case a broker exists that processes the **QoS parameters** the QoS information is submitted to the broker, otherwise directly to the retailer.

The broker

- performs the **QoS establishment** of individual services,
- features a **QoS mechanism** to support this establishment, and
- is capable of processing **QoS information** to support the QoS establishment. The QoS information represents the QoS requirements of the client or the retailer, and the QoS parameters of individual services from 3rd party service providers or aggregated services (taken as individual) from other retailers.

3.2 Quality-of-Service Characteristics

This section discusses the QoS characteristics in detail. It will explain which particular QoS characteristics are relevant in an SOA. More specifically – because the topic of this thesis is QoS-based trading – the goal is to determine which QoS characteristics are considered as selection criteria for the trading of services. The first chapter stated that a requirement to the aggregation method is the independence from particular QoS characteristics. However, it is still necessary to identify the relevant characteristics to determine which the trading must support. The QoS Framework published by the Information and Technology Section of the ISO [52] presents a comprehensive set of QoS characteristics grouped into various QoS categories. This standard includes the following categories:

- **Time-related.** The first category mentioned in this standard covers all QoS characteristics that use the time as measure, such as the execution time or the time until the result of a service invocation arrives, which is the response time. A related characteristic is the delay. The delay denotes the time in which the request is transported over the network. In addition, the standard mentions the notion of lifetime, which describes the time that a service remains available.

- **Coherence.** The coherence describes how well a system maintains related data over temporal or spatial distances. Data must be kept coherent if the service actually provides various individual operations. In this case, the exchanged data must kept coherent during the individual invocations. Spatial coherence is required in a distributed system, if related data is distributed among different computers within this system.
- **Capacity.** The capacity subsumes different QoS categories that describe a measure for how much data can be processed within a given period. QoS categories in this sense are for example data capacity, throughput, or processing capacity. Throughput denotes how much data can be processed within a given amount of time.
- **Integrity.** The integrity describes aspects of correctness when processing information. It is related to the concept of accuracy which denotes how much a result meets the expectations of the requester.
- **Safety.** The safety subsumes QoS characteristics that describe to which degree the stability of an operation is ensured.
- **Security.** The security covers different aspects such as the protection of information which also involves a mechanism for access control, or the ability to authenticate users of a system.
- **Reliability.** The term “reliability” subsumes a set of very related QoS characteristics, namely availability, fault containment, fault tolerance, and maintainability. For services, the availability represents a measure for how often a service can be invoked while the reliability describes how often the result of a service invocation is correct.

This list of categories represents a general set of QoS categories in distributed systems. Thus, the question rises if all these are relevant for the application in an SOA, especially for the trading. The UML QoS-Profile of the OMG [99, section 10] provides a narrowed set of common QoS categories. The authors of the UML QoS profile suggest reusing these for particular QoS modelling efforts. The UML QoS-Profile considers also the ISO QoS framework as the main reference for the selection of application-independent categories. Because the UML QoS-Profile was published six years after the ISO framework, it also reflects, how the different QoS categories have evolved. It considers the following QoS categories:

- **Performance.** The UML QoS-Profile does not feature the category “time-related” as mentioned in the ISO standard. It rather uses the performance to provide a category that includes the time. Mentioned are: throughput, latency, efficiency, and demand. The latency denotes the time interval between request and response. The efficiency indirectly denotes the consumption of resources. The demand is a proposed measure for how many times a service is requested.

- **Dependability.** In the UML QoS-Profile, the category of dependability is related to the reliability in the ISO QoS framework. The dependability includes also the availability.
- **Security.** Compared with the ISO standard, this category includes in the UML QoS-Profile the following aspects: the protection in terms of access and protection against manipulation and the confidentiality of actors.
- **Integrity.** The integrity is used with the same meaning as presented in the ISO standard.
- **Coherence.** Like in the case of the ISO QoS framework, the coherence in this sense refers to the issue that in distributed systems, data might be locally distributed and therefore must be kept coherent.

This list of QoS categories differs mainly from what the ISO standard proposes by subsuming time-dependent and capacity-dependent categories under the main concept of performance. Time and capacity are similar categories; their subsumption under performance makes sense. However, similar to the categories mentioned by the ISO standard, the UML QoS-Profile does not explicitly cover the application in an SOA. Moreover, not all these categories and mentioned characteristics are relevant for the trading process. An importer can presume that a service is capable of processing data coherently and of ensuring the integrity of data. Thus, this thesis considers only the categories performance (time and capacity), dependability (reliability) and security as suitable for trading services from the ISO standard and the UML QoS-Profile.

3.2.1 Quality-of-Service Characteristics for Web Services

Regarding Web services, Menasce has presented different publications which discuss QoS issues in Web services. In his work he mentions as the relevant characteristics response time, throughput, security and availability [90]. In further work he also discusses the response time and the cost in compositions of Web services in particular [91, 92]. Regarding the composition of Web services, Zeng et al. have presented a framework for the QoS-aware composition of Web services [159]. Their work is closely related to this thesis and therefore it will be discussed in more detail in future sections. Their discussion covers the QoS characteristics price, duration, reputation, success rate, and availability. The reputation is a characteristic which is not specifically mentioned by the ISO framework or the UML QoS-Profile. It represents a rating that is provided by an importer based on his experience with using a service.

Patel et al. discussed the modelling of Web services and the creation of service descriptions which involved also a discussion about different QoS characteristics [105]. Their contribution focusses on a modelling structure for expressing the QoS of a service. Their selection of QoS characteristics is divided into two categories: The first category consists of the latency, which is used as a synonym for response time, throughput, reliability, and cost. The other category is named internet-specific and consists of availability, security, accessibility and regulatory. The characteristic regulatory denotes a measure for how well the service complies with given (organisational) regulations. In addition, Patel et al. define a separate QoS characteristic named task-specific which denotes how well the returned answer meets the expectations of the importer.

Section 3.1.2 has introduced different research work about processing QoS in an SOA: One part covers the QoS negotiation based on contracting approaches, another the extension of existing repositories with the ability to process QoS information. A third part discussed dedicated broker architectures. The contributions from Ludwig et al. (WSLA, [84]) and Tosic et al. (WSOL, [133]) discuss the negotiation of QoS. They do not mention particular QoS categories or characteristics. The WSOI, based on the WSOL that Tosic et al. have also presented, focusses on the response time, which measures the time from the submission of the request until the return of the result when invoking a service [132]. A similar contribution by Ludwig also discusses how to ensure the negotiated QoS from the perspective of the service exporter [83]. Regarding the considered QoS characteristics, he discusses those subsumed by the performance category in the UML QoS-Profile.

In the field of UDDI extensions, Ali et al. have introduced the UDDIe. They give no particular discussion about QoS characteristics but mention bandwidth as an example [2]. The bandwidth denotes how many requests a service can process within a given time and is a synonym for the throughput. Lee, who has also discussed the approach to extend UDDI for processing QoS, discusses the cost, which is a synonym for price as used by Zeng et al. [159], and the response time as the relevant characteristics for Web services [78]. Ran provides the most detailed discussion about relevant QoS categories when extending the UDDI specification [112]. He introduces a QoS model for Web services that provides the following four main categories:

- **Run-time related.** In the group of run-time related characteristics, Ran subsumes scalability, which describes a measure for transactions per second, and capacity which denotes the number of concurrent requests. Moreover, Ran puts performance into this main category, which contains the characteristics response time, latency and throughput. According to his work, response time denotes the time to process a request. Latency covers the situation that a request is queued at the side of the provider. Throughput shows a strong relation to scalability and capacity.

  As a third sub-category, Ran introduces the reliability, which contains the characteristics mean time between failure (MTBF), mean time to failure

(MTTF) and mean time to transition (MTTT), of which Ran admits that they show a strong relation to the concept of availability. MTBF is usually used if a failure denotes an exceptional state that can be recovered. MTTF refers to a failure that, in general, cannot be recovered. MTTT represents an unusual characteristic. It refers to the idea that an entity can show two general states: a good state and a bad state. Then, the MTTT denotes a measure about how often a change between the two states occurs [112]. In addition to these three values, this category contains also the characteristics robustness, exception handling and accuracy.

- **Transaction supporting.** With this category, Ran refers to the four main properties that represent the standard characteristics of a transaction, which are atomicity, consistency, isolation and durability, often abbreviated with ACID. Since the characteristics of this category are hard to quantify with numerical measures, this work interprets them as non-functional characteristics.

- **Configuration- and cost-related.** This group of QoS characteristics relates to organisational issues of the service and its development. These include "regulatory", which denotes the compliance with (governmental) regulations, a degree for supporting technology standards, stability, which refers to a description of how often the provided functionality is updated, as well as cost, and completeness. This characteristic denotes the completeness of provided functionality.

- **Security.** This category subsumes a set of standard security aspects which include the provided authentication technique, the supported authorisation mechanism, the facilities to ensure confidentiality, the support accountability and auditability, the provided degree of data-encryption, and the provided counter-measures to prevent repudiation after a service invocation.

Compared to the UML QoS-Profile, the set of security characteristics discussed by Ran is more comprehensive and more specific regarding the SOA. Another difference is that the main category, named "run-time related", subsumes also the category performance along with different other categories like throughput. The appropriateness of the category for configuration-related QoS characteristics is arguable, because they partially cover functional aspects (e.g. the concept of completeness) which might not fit into the scenario of trading services: Would a service importer query a broker with requirements such as the completeness of the provided functionality? For this thesis this question is answered negative: Section 7.2.2 will explain how a presumed matchmaking of functional suitability covers this point. As a conclusion, the characteristics found in Ran's categories "transaction supporting" and "configuration", excluding the cost, are considered not relevant to the trading of services. In summary, the relevant QoS categories presented by Ran are the run-time related, the cost and the security.

The third cluster of research work related to the QoS in an SOA is formed by the works that propose dedicated brokers for the trading of services. Table 3.2 presents an overview of the considered QoS characteristics in the field of research on brokers for trading Web services. Regarding the work of Maximilien and Singh [88], their list of QoS categories and characteristics provides more concepts than listed. They have been omitted, because they were discussed in the work by Ran. The listing has the purpose to identify the relevant QoS characteristics for the trading of services as identified by other researchers. In summary, these publications mention very similar QoS categories. Among the characteristics, the up time and availability can be considered equal. The same applies to the pair of MTBF and reliability, and the pair of timeliness and latency. If all QoS characteristics that were discussed in at least two of the five mentioned publications are considered relevant, then the list contains: response time, cost, availability, reliability, and throughput.

Research Group	**QoS Characteristic**
Wang et al., 2004 [136]	Categories: performance, reliability, timeliness and security, which each have one referring characteristic assigned.
Degwekar, et al., 2004 [115]	Response time, cost, and availability.
Maximilien, Singh, 2004 [88]	Response time, cost, latency, throughput, capacity, MTBF, and up time.
Yu, Lin, 2005 [157]	Response time, cost, reliability, and availability.
Serhani et al., 2005 [116]	Response time, cost, latency, throughput, and availability.

Table 3.2: QoS characteristics in the domain of Web services.

3.2.2 Summary of Quality-of-Service Characteristics

The different fields of research and standards have revealed many similar categories. From the research work mentioned above, the UML QoS-Profile, which is based on the ISO QoS framework, the work of Ran and the five publications mentioned give a broad set of QoS categories. In addition, the previous section has identified the set of commonly discussed QoS characteristics used for Web service brokers. The common QoS characteristics from these three areas are listed in Table 3.3.

This table mentions the set of common QoS categories that are considered for the use of service trading: the throughput, the response time, the cost and the availability. The UML QoS-Profile and the QoS model of Ran also mention the security which can be considered for trading as well. This aspect has been discussed by Wang et al. in detail [136]. In addition to the identified set, Zeng et al. consider the reputation as relevant for trading [159]. As defined by them, the reputation represents a measure for how well a service performs based on the experiences of different importers. For this thesis, the reputation is also considered as a measure

UML QoS-Profile	**Ran's QoS Model**	**Research on Web Services**
throughput	throughput	throughput
latency (response time)	response time	response time
efficiency (cost)	cost	cost
availability	availability	availability
reliability	reliability	reliability
security	security	reputation

Table 3.3: Summary of QoS characteristics

to capture non-functional service selection criteria that are usually not expressed with quantifiable measures. For example, services can be also selected by their organisational affiliation, if a business would like to prefer selected partners, by the country from which services are provided, if trade agreements are relevant, or by an individual measure of how much a service provider is trusted. These examples make clear that the term reputation is dependent on the view of the user or the party that assigns reputation values to service providers (cf. Kalepu et al. [69]). The definition of a measure that expresses reputation can involve many input parameters. For this work, it is assumed that depending on the different mentioned factors, the reputation results in a absolute value, which is free from a particular value range (contrary to the repsonse time) or unit.

Chapter 4

Aggregation of the Quality-of-Service in Service Compositions

Prior to performing the QoS-based selection, a method that allows the aggregation of the QoS in a composition of services must exist. Based on the given QoS characteristics of the individual services, a statement must be derived that covers the QoS characteristics of the composition. Otherwise, a selection algorithm could not determine the result of choosing a particular candidate. In addition, if there are requirements that the composition must meet, there has to be a method that proves if the assigned services will provide the required QoS characteristics.

As the first chapter has explained, such an aggregation must involve the structural arrangement of the services in the composition. Thus, the first step of such a method is a model that allows the description of the structure. For this model, two main goals exist:

- **Technology independence.** The structural model should not rely on existing languages to describe compositions. Although Section 2.4.1 has indicated that WS-BPEL might play a more important role in the field of service compositions than similar other proposals, such a structural model should not be limited to this particular proposal. On the other side, the existing composition languages must be considered to determine the relevant structures that occur in service compositions.

- **Focus on aggregation of QoS.** The model must represent an abstraction from all technical details and focus only on the information that is needed for aggregating the QoS characteristics. Composition models might include more than information relevant for designing and running the composition. Examples are the names of interfaces, the order of exchanged messages etc., which can be omitted for such a model.

4.1 The Business Process Execution Language

WS-BPEL has gained most attention when it comes to the description of service compositions. This language allows description of abstract service compositions – or business processes – as well as concrete compositions with individual available Web services. The basic element in this language is a process that consists of sub-elements. A sub-element represents a description of a structural part of the process: which services are involved, what information is exchanged, how the execution can be compensated in case the invocation is cancelled etc.

The structure of the process is important for the aggregation of the QoS and thus, where WS-BPEL is concerned, the structural sub-elements of the process element are relevant. WS-BPEL supports two ways to describe the structure: One is by using the link and flow elements; these elements allow the definition of a direct precedence between two tasks or services. With the link element, the order of invocations can be described as a directed graph. The second option supports different building blocks that allow the modeller to form a recursive graph to describe the structure of the composition. In such a structure, each building block has one in-going edge and one outgoing edge. Each building block defines an execution structure (e.g. parallel, sequential etc.) and consists of further building blocks or individual services. Within a building block, only one execution structure applies, for example, in a building block denoting a sequence, all enclosed elements are executed sequentially. In WS-BPEL, the building blocks are named "structured activities" and support the following execution structures [126, section 12]:

sequence	Executes all enclosed elements in a sequence.
if	Executes a selection of the enclosed elements according to a Boolean expression. Compared to the C programming language, this is similar to an if-then-else statement.
while	Executes the enclosed element while a given expression holds.
repeatUntil	Executes the enclosed element until a given expression holds.
pick	Executes enclosed activities when a defined triggering condition occurs. Compared to the C programming language this is similar to a switch statement.
flow	Executes the enclosed activities concurrently. Optionally the execution structure of the enclosed elements can be altered by using optional link elements, which can be used to define precedence relations between the activities.
forEach	Repeats the execution of the enclosed activities for a given number of times. This element can either execute the given amount of executions in parallel or in sequential manner.

In addition, the WS-BPEL also provides special elements for interrupting the execution for a given amount of time (the wait element) and for just doing nothing (the empty element), which is useful to put an execution on hold until an external event occurs. The elements to define arbitrary links among the tasks or services in the composition appear redundant to the building block elements. A modeller can design a sequential execution of services by either using the sequence element or by using the flow element with the corresponding link constructs. A sequence is a very basic structure that a modeller might use often. Thus, the use of such a basic building block offers a convenient way to describe simple task or service executions. Another reason for the integration of these two basic ways is the fact that the predecessor of WS-BPEL, namely BPEL4WS, represents the result of the merge of the XLANG composition language by Microsoft and WSFL by IBM (cf. Section 2.1.2). When comparing these two languages, WSFL provides a flow element with link elements only [80]. Contrary to that, XLANG does not have such a flow element but only provides basic building blocks (namely sequence, switch, pick, all, while, empty) [129]. When the two languages were merged in order to form the BPEL4WS language, obviously the authors have decided to keep both ways for describing the composition structure.

Kiepuszewski et al. discuss these two different groups of structural modelling elements for modelling workflows [73]. They distinguish between structured workflows, which is similar to what the XLANG represents, and arbitrary workflows, which have some standardised routing elements (or- and and-split and join operators) and a defined beginning and ending activity each. Their work explains that the arbitrary workflows have the advantage that they offer a more intuitive and suitable way to model workflows to modellers. Structured workflows have the advantage that less sophisticated verification mechanisms are needed and that they are easier to implement. The authors mention some examples of workflow management systems being capable of processing only structured workflows for these reasons.

In addition, Kiepuszewski et al. discuss in their paper possible transformations from arbitrary to structured workflows. A modeller can use such a transformation to start the modelling of a composition with more intuitive arbitrary workflows and then he can transform the model into a structured workflow. The discussion covers transformations based on the duplication of nodes which represent either tasks or routing elements and the use of auxiliary variables to decide on additional branches. Their results show that not all structural occurrences found in an arbitrary workflow can be transformed into a structured workflow.

4.2 Workflow Patterns

Section 2.1.2 has pointed out that the WS-BPEL represents a popular composition language in the area of Web services. Thus, the discussion in the previous section about the capabilities of WS-BPEL is useful to analyse the state-of-the-art in composition languages. However, one requirement for the structural model is its independence from particular languages or standards. An analysis is required that discusses the structural abilities of different languages available for this purpose. Section 2.2.1 has already mentioned the work of van der Aalst on the common structural elements found in composition modelling languages [143]. His work refers to a prior work by him and his colleagues about common patterns found in workflow management systems named workflow patterns [146]. The following part introduces the workflow patterns and discusses which patterns are relevant for a structural model for the aggregation of QoS.

The workflow patterns were created to define abstract capabilities for the comparison workflow management systems. They mainly describe the capability of executing different workflow structures and the functional behaviour of these systems. A main benefit of these patterns lies in the ability to compare workflow management systems by their functional rather by their non-functional aspects (like platform requirements or usage of hardware resources). Considering the workflow patterns offers the following advantages for this work:

- **Technology independence.** The workflow patterns are independent from particular technologies or composition languages. The workflow patterns were designed to provide a uniform approach for the comparison of workflow management systems and their flow definition languages. Thus, the patterns can be regarded as a comprehensive description of the structural characteristics of different workflow management systems and flow languages.

- **Matureness.** The first version of the workflow patterns was published in 2000 [145] and were revised since then in 2003 [146]. Also, different researchers and industrial efforts have taken up the patterns.[1] Therefore, the work about the workflow patterns can be regarded as often-reviewed research.

- **Service compositions.** As pointed out in Section 2.3, the existing composition languages have their roots in the modelling languages for workflows. As mentioned, a comparison of these languages based on the workflow patterns already exists [143], as well as a detailed analysis of the flow language BPEL4WS [151]. In these works, the authors explain the strong similarity between characteristics of workflows and service compositions.

Regarding the second requirement, which is the focus on QoS aggregation, the workflow patterns contain many elements that are not relevant for the aggregation of QoS. For example, the Pattern 19 “Cancel Case” describes the ability to cancel a running process. The application of the QoS aggregation to perform QoS-based selection takes place at the design time (cf. Chapter 7); it is about the planning of the composition. Consequently, the QoS aggregation is not concerned with capabilities such as cancelling a running composition and thus this workflow pattern is

[1]The research group working on the workflow patterns has published a comprehensive overview about the impact and use of their patterns on the Internet at http://is.tm.tue.nl/research/patterns/impact.htm.

No.	Workflow pattern	Synonym(s)	Rel.	Abstraction
Basic control flow patterns				
1	Sequence	Sequential routing	Y	Sequence
2	Parallel split	AND-split	Y	AND-split
3	Synchronisation	AND-join	Y	AND-join
4	Exclusive choice	XOR-split	Y	XOR-split
5	Simple merge	XOR-join	Y	General join
Advanced branching and synchronisation patterns				
6	Multi-choice	OR-split	Y	OR-split
7	Synchronising merge	Synchronising join	Y	OR-join
8	Multi-merge		Y	AND-split with AND-join
9	Discriminator	m-out-of-n, partial join	Y	m-out-of-n
Structural patterns				
10	Arbitrary cycles	Loop, iteration, cycle	Y	flow-link, loops
11	Implicit termination	Nothing to do anymore	N	
Patterns involving multiple instances				
12	M.I. without synchronisation		Y	AND-split with AND-join
13	M.I. with a priori design time knowledge		Y	represented by patterns 2-9
14	M.I. with a priori runtime knowledge		N	
15	M.I. without a priori runtime knowledge		N	
State-based patterns				
16	Deferred choice	External choice, deferred XOR-split	Y	XOR-split
17	Interlv'd parallel routing	Unordered sequence	Y	Sequence
18	Milestone	Test arc, state condition, Withdraw message	N	
Cancellation patterns				
19	Cancel activity	Withdraw activity	(Y)	Immediate End
20	Cancel case	Withdraw case	N	

Table 4.1: Workflow patterns [146] and their relevance for the QoS aggregation.

ignored in the discussion about QoS aggregation. Regarding the basic control flow patterns and the advanced branching and synchronisation patterns, these are also relevant for a structural model to aggregate the QoS. From the section "structural patterns", the arbitrary cycles are considered relevant. From that the remaining sections, starting with "Multiple Instances", they require an additional discussion.

Table 4.1 summarises the described workflow patterns. In addition, the table lists also the relevance of a pattern for the QoS aggregation as well as its proposed structural abstraction. As a main rationale, only patterns are relevant that address the structure at design time. The identification for some patterns is trivial (e.g. for the a sequence). For the parallel and conditional routing constructs, basic abstraction is used: The interpretation of an AND-element is to involve all possible tasks, whereas the XOR-element involves exactly one possible task. The OR-element involves less than all but more than one possible task. However, other patterns require some discussion about the relevance for composition:

- **Multi-merge.** This pattern describes a specific join operation of parallel executions in a workflow arriving at the joining point. At this point, for each execution thread arriving in parallel, the execution environment starts one following separate thread with execution of the following activities. These patterns can be simulated in the modelling phase by using AND-splits and AND-join operations for the QoS-aggregation.

- **Arbitrary cycles.** This pattern covers the ability to express precedence links between tasks in the workflow that form a loop. It allows links that go outside a building block, if a composition language provides the concept of building blocks. As Kiepuszewski et al. have pointed out, this pattern separates the structured workflow models from the arbitrary workflows [73]. The simple loop can be found among the structured activities of WS-BPEL (i.e. while, repeatUntil). However, the simple loop represents only a special case of what is covered by this pattern. The support of arbitrary loops will be discussed in detail in Section 4.5.

- **Implicit termination.** The implicit termination pattern denotes that an engine is capable of terminating the flow, if no processable data is available anymore. Whether an end is explicitly stated or ends implicitly does not have an impact on the aggregation of QoS properties during design time. Therefore, this pattern is not taken into account.

- **Patterns involving multiple instances.** This set of patterns generally targets the run-time abilities of a workflow management system. Therefore, in the modelling phase, each workflow or each composition can be seen as one unit for property aggregation. If at design time, the number of instances is already known, this results in a parallel split (for the workflow patterns 12 and 13) with an open end. If the number of instances is determined during runtime (workflow patterns 14 and 15) the handling of QoS properties must be processed by the run-time environment of the composition and therefore is not taken into account in the modelling process. The result of the QoS aggregation for this case would be a QoS statement covering one instance.

- **Deferred choice.** The difference between a deferred choice in workflow context and the XOR-split is that, in the deferred choice, the split is performed based on external input while the standard XOR-split relies on information being part of the workflow. In the description of the workflow patterns, the deferred choice is referred to as a “state based pattern”. Consequently, it is not regarded for having an impact on the modelling phase.

- **Interleaved parallel routing.** This pattern describes that involved tasks are sequentially executed in an arbitrary order. In case of the aggregation of QoS, this pattern is subsumed by the concept of a plain sequence. The upcoming Section 4.3 will explain why the order is not relevant for the QoS aggregation of sequential tasks.
- **Milestone.** In the case of service compositions, this pattern is not relevant because describes the ability of a workflow management system to trigger an external activity once a particular point in the workflow has been reached. Such a behaviour that does not affect the QoS of the workflow or of the service composition itself. Furthermore this pattern is also not supported by any of the flow languages analysed in [143].
- **Cancel activity.** This pattern describes the functionality to stop the workflow execution at a given time. Since this pattern can occur at any place in the workflow, no specific routing element or structural pattern can be given. A possible solution for the QoS aggregation emulates this behaviour with an XOR-split and a following link to the end of the workflow. The support of such a pattern will be discussed in Section 4.5.1.
- **Cancel case.** This cancellation pattern describes the ability of workflow management systems to cancel the execution of a workflow. Regarding the QoS aggregation at design time, this is regarded as exceptional behaviour and thus not considered relevant.

The abstraction results in three relevant sequential structural patterns: the trivial sequence, an unordered sequence, and a basic loop. For the parallel and conditional cases, the abstraction covers different split and join operations, which can be summarised mainly in AND-, OR-, and XOR-split and -join operations. The next section will discuss how these abstractions can be used to form a structural model of the composition in order to perform the QoS aggregation.

4.3 Structural Model of Service Compositions

Based on the analysis of the workflow patterns in the previous section, a structural model can be derived that provides the necessary structural elements for modelling service compositions for the QoS aggregation. The approach transforms a description of a composition into such a model. Then, an algorithm performs the aggregation on this model. This approach for the aggregation has been already introduced in previous work (cf. Jaeger et al. [59, 63, 64]).

The introduced model has been build upon on structural patterns rather than a classification of QoS values. In the previous work, it has been explained that classification approaches for QoS characteristics are not helpful to establish an aggregation mechanism. Software can use the properties of different characteristics to consider aspects such as the direction (i.e. either a larger or smaller value denotes better QoS), the definition (e.g. average, percentile, fixed value), or the value type (e.g. relative, absolute). However, QoS characteristics show many different properties and thus cannot lead to a unified approach as the following points explain:

- A general differentiation between increasing or decreasing directions can be applied for every QoS characteristic, because it is assumed that every QoS characteristic provides the concept of better or worse. However, an aggregation rule for one dimension still can be different depending on, whether relative or absolute numerical values are processed.
- QoS characteristics might have a direction but show only discrete values. As a consequence, the aggregation requires a different algorithm than for continuous dimensions: Instead of a calculation particular rules must be performed.
- QoS characteristics might refer to a statistical definition – like the variance, the mean or percentiles. For example, considering percentiles, the aggregation must cover specific definitions for aggregation rules covering parallel arrangements.
- Beyond the three mentioned points, QoS characteristics can have different measures, making an aggregation also more complicated.
- The definition of QoS characteristics may vary depending on the environment. Thus, a classification of QoS characteristics for generalisation cannot be established. Instead, a software environment that performs the aggregation must cover relevant QoS characteristics for a specific application case and provide the exact definitions of the aggregation rules for the selected categories.

Algorithms to aggregate the QoS in compositions already exist. For example, Zeng et al. have discussed an algorithm which finds the shortest (or longest) path in a graph in order to determine the maximum response time [159]. However, as the five points mentioned above indicate, for the different properties that a QoS characteristic can show, many different algorithms must be implemented in order to cover the different kinds and facets.

The work of Puschner and Schedl about calculating maximum execution times of computer programs [111] has already been mentioned as the work that inspired the idea of aggregating QoS in compositions. In their work, the authors present two main approaches: one works on the building blocks of a structured model to calculate execution times and the other is oriented to find a path in a graph-based structure. A particular graph-based approach by Puschner and Schedl expresses the problem as an integer linear programming problem, which can be solved with existing software tools. Besides, their work covers only the execution time and thus other QoS characteristics are not supported.

For this work, an approach is preferred that defines aggregation methods for the basic building blocks found in compositions – depending on the relevant QoS characteristics of the application case. For each of these structural elements – called composition patterns in the remainder of this work – aggregation rules are defined for each QoS characteristic. Then, an arrangement of composition patterns can be used to represent a model of an existing composition. Based on the aggregated QoS statement for a particular composition pattern, an algorithm can aggregate each pattern by aggregating sub-patterns until the entire composition is covered. In addition, such an aggregation method requires some basic assumptions to provide useful QoS statements as a result (cf. Jaeger et al. [63]):

- **Independence.** It is assumed that the services in the composition do not depend on each other regarding their successful execution. This assumption presumes that the result of the execution or the delivered QoS of one service does not affect the QoS characteristics of other services. Dependencies would occur, for example, if a computer would host multiple services of the composition and therefore have a common point potential failure which would affect QoS characteristics, such as the reliability, in a specific way.
- **Trust.** For the aggregation of QoS it is assumed that the given values of a service are correct. Trusting the correctness of QoS information represents a separate issue from the algorithmic aggregation. For the discussion about the aggregation, it is assumed that the QoS of individual services represents an already negotiated and accepted agreement between service importer and exporter.
- **Uniformity.** For the aggregation, it is assumed that the given values are compatible to each other. All individual values must conform to a common group of measures. For example, it is assumed that a property describing the response time uses compatible measures such as milliseconds, seconds, or minutes. Furthermore, it is assumed that all given values conform to the same definition. For example, all services start and end from the same point of measurements to define the response time, e.g. when the request has been submitted and the response has been delivered completely.
- **Equipartition.** The aggregation of QoS for selection of services to form a composition takes place at the design time. In this phase, it is assumed that the execution environment executes services in conditional join and split cases with equal probability. For example, if in the case of an XOR-split the current flow chooses a service from a number of services, it is assumed that the execution of the service follows an uniform distribution. The coverage of other kinds of distributions represents a possible extension to the composition model.

Based on the workflow patterns and these considerations, a model consisting of nine basic composition patterns can be derived. Figure 4.1 shows these patterns using a simple notation that indicates directed graphs. This notation uses rectangular boxes to denote a routing element and it uses circles to denote a task. Arrows define the precedence relations between the tasks and routing elements. From these patterns, two sequential patterns are defined (cf. Jaeger et al. [63]):

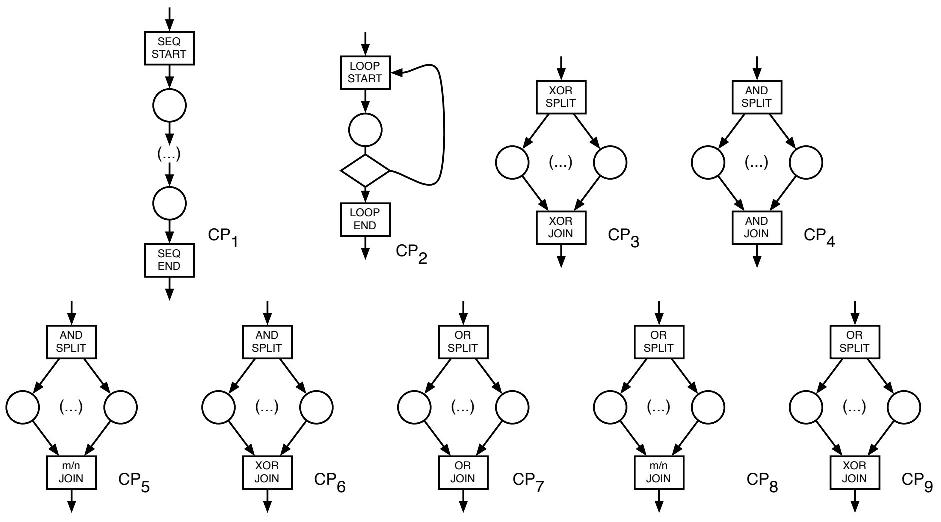


Figure 4.1: Composition patterns.

- **Sequence of service executions (CP_1).** A sequence can prescribe a specific order in which an execution environment executes the services. Alternatively, the execution environment executes the services sequentially in an arbitrary order. For the aggregation model, the order of the executions is not relevant. The aggregation rules for the sequence of service executions, which will be introduced in a subsequent section, entirely operate on unordered sets, or represent an addition or multiplication of the individual values. Since a set contains elements without any applied order and the addition and multiplication shows commutativity, such aggregation rules can be also applied to any order of execution.

- **Simple Loop (CP_2).** The execution of a service or a sub-arrangement of services and composition patterns is repeated for a certain amount of times.

For the parallel and conditional patterns, the relevant workflow patterns do not fit directly into a model for aggregation. The reason is that for the algorithmic aggregation of values, not only the split but also the join condition must be considered. This can be shown by the following example: In the example, the minimum response time is subject for aggregation. If a flow splits into a number of parallel flows, three join structures could be possible:

a) joining with synchronisation of all parallel flows or

b) joining with synchronisation of one flow, and

c) allowing joining of more than one flow but less than all flows.

Figure 4.2 shows an example of the three setups with each three services showing a minim execution of 3, 5 and 7 units. In case *a)*, the minimum response time would be the minimum response time of the greatest value (the slowest). In case *b)*, the smallest of the involved values represents the minimum response time possible. For the case *c)*, the relevant value depends on the number of considered flows for synchronisation: For example, if two flows are relevant for synchronisation, the second-smallest value denotes the minimum response time of this structure. It must be noted that this example presumes that after the synchronising flow has arrived the other flows are ignored.

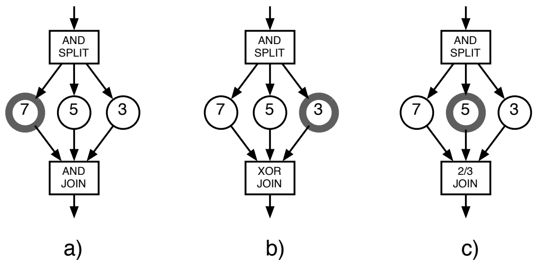


Figure 4.2: Example of join-relevant aggregation of response time.

In addition, another reason exists for combining different split and join patterns: A model that consists of composition patterns can be aggregated by a recursively working algorithm. Then, an algorithm must identify independent, self-contained atomic patterns in the composition. Consequently, each atomic structure must show a defined start and end of service executions. In the first example, various combinations between split and join operations are possible. However, only a part of them make sense in combination. For example, an XOR-split cannot occur in combination with an AND-join. The following relevant composition patterns were identified from the theoretically possible combinations (cf. Jaeger et al. [63, 59]:

- **XOR-split followed by an XOR-join (CP_3).** In a parallel arrangement only one task is started. Thus, the synchronising operation synchronises only the started task.

- **AND-split followed by an AND-join (CP_4).** From a parallel arrangement all tasks are started, and all tasks are required to finish for synchronisation.

- **AND-split followed by a m-out-of-n-join (CP_5).** From a parallel arrangement all n tasks are started, but less $m < n$ tasks are required to finish for synchronisation.

- **AND-split followed by an XOR-join (CP_6).** From a parallel arrangement all n tasks are started, but just one task is required to finish for synchronisation. This pattern has been introduced by Ladner [77] as was then published as an extension to the original model [63].

- **OR-split followed by OR-join (CP_7).** In a parallel arrangement a subset of the available tasks is started, and all of the started tasks are required to finish for synchronisation. For example, from four available services, the run-time environment starts always three of them which must also finish successfully.

- **OR-split followed by a m-out-of-n-join (CP_8).** In a parallel arrangement a subset of n tasks of all are started, and $m < n$ tasks are required to finish for synchronisation. For example, from four available services, the run-time environment starts always three, of which two must finish successfully.

- **OR-split followed by n XOR-join (CP_9).** In a parallel arrangement a subset of n tasks of all are started, and only one task is required to finish for synchronisation. This pattern has also been introduced by Ladner [77] as an extension to the original model of composition patterns.

Based on these elements, a structural model of the composition can be created that meets the following characteristics:

1. The structural model is a directed graph consisting of nodes which represent tasks and nodes which represent routing elements.

2. The arrangement of the nodes and routing elements is as such that the entire model can be collapsed into one node by applying the set of composition patterns as shown in Figure 4.1. This implies the following points:

   - This graph must not contain loops except those matching the loop pattern.
   - The routing elements must be used pair-wise as defined in Figure 4.1.
   - The graph must start with a routing element that has only one or more outgoing paths and end with a corresponding routing element that has only one or more incoming paths.
   - Within such pairs of routing elements, the number of outgoing paths of a splitting or starting element must be equal to the number of incoming paths of the corresponding joining or ending element.

3. Besides "normal" tasks and services, empty tasks that do not perform any functionality are possible. Empty tasks are necessary to form structures that are not directly supported by the given nine patterns. The application of these tasks will be explained further in the upcoming Section 4.5. Empty tasks behave neutral in terms of the applied QoS characteristics, i.e. their invocation costs nothing, does not take any time etc.

4. For drawing such structures by using the patterns as shown in Figure 4.1, the sequential pattern elements will be simplified and thus, the starting and ending element will not be shown. Moreover, empty tasks will not be drawn but just represent a line between one starting routing element and the corresponding finishing routing element.

The above mentioned description defines a structural model that preserves the precedence among tasks in a composition structure while conforming to the structural elements represented by the composition patterns. However, for the aggregation method, which will be introduced in the next section, the precedence relations between the elements in a sequence do not have an impact on the aggregation of the QoS values. An order of the elements is not required for the aggregation, and thus, the enclosed elements of a sequence can be represented by a set. Since precedence relations do not exist between the elements of parallel structures, a set is an adequate representation structure for these elements as well. The result from these considerations is a data structure that represents the composition structure as follows:

- A composition structure is represented by a set $\mathbb{K}_1$.
- The set $\mathbb{K}_1$ can contain tasks of the composition $t_1, \ldots, t_i$ or further sub-sets $\mathbb{K}_2, \ldots, \mathbb{K}_j$.
- Each set $\mathbb{K}_x$ has a special type assigned, namely one of the composition patterns $CP_1, \ldots, CP_9$.
- Any set $\mathbb{K}_x$ can contain tasks or further sub-sets.

This model poses the requirement that a composition structure can be represented functionally equivalent by the recursive structure of task-pattern-sets. This limitation prevents the direct application of this structural model to composition models which allow the use of arbitrary links between the tasks or services. The concept of flow and link elements found in WS-BPEL represents an example for such structures. The limitations of such a structured model has been discussed by Kiepuszewski et al. [73] and the upcoming Section 4.5 will give a further discussion on these limitations.

4.4 A Method for Quality-of-Service Aggregation

The basic approach for the aggregation is to step-wisely collapse a graph representing the composition structure into a single node by alternately aggregating simple sequences and parallel service executions. Figure 4.3 outlines this method: An algorithm would identify composition patterns and subsequently perform the aggregation on the level of each pattern until one statement remains. Thus, the algorithm does not consider the entire graph at once from a global perspective, but rather only local composition patterns.

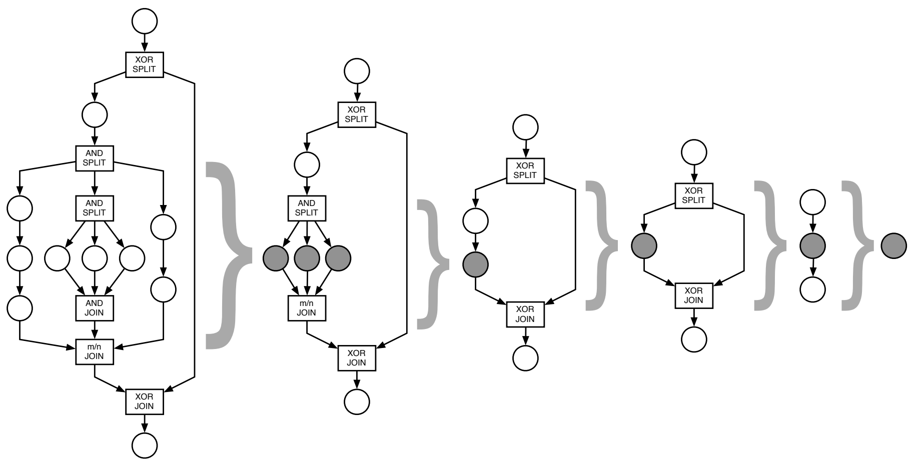


Figure 4.3: Collapsing the graph step by step.

This approach builds on the aggregation of different QoS characteristics for each set $\mathbb{K}_x$, which is defined to refer to a particular composition pattern. Therefore, for each characteristic and for each pattern type an aggregation rule must be defined. A basic set of aggregation rules has already been discussed in previous work (cf. Jaeger et al. [63, 64]. For this thesis, the discussion is extended and covers the different QoS characteristics that were mentioned in Section 3.2.

In the following sub-sections, the aggregation of relevant QoS characteristics is discussed and defined. For the definition, the following notation is used: k represents the number of services that the considered pattern contains, l represents the assumed number of repetitions for the loop pattern, x_i represents a given value for each service with index i denoting a particular element, and $\mathbb{K}$ is the set containing all x_i. For the case that a set $\mathbb{K}$ contains one more sub-sets, the method presumes that these have been aggregated first and thus are represented in $\mathbb{K}$ by a single value as well. The variable x_a represents the aggregated value. In the previous work, n has been used as the variable that refers to the individual services which collided with the “m-out-n-join”-constructs from the notation of the composition patterns.

This aggregation model requires two remarks: The first remark covers the loop pattern and the second covers the OR-split patterns. For the case that the number of loops will be determined at run-time, the model does not represent the loop pattern sufficiently. For example in WS-BPEL, the condition of a loop statement is defined with a Boolean expression (with a while-construct) and consequently the number of loops can be determined at run-time only. However, an algorithm must know the number of occurring loops for the aggregation of QoS. Otherwise, the model represents an estimation about the number of loops. Generally, the aggregation rule as proposed for the loop case would fit well in monitoring the QoS for analysis. Nevertheless, it cannot serve as a decision support in the modelling process if the aggregation task does not know the number of loops at the design time.

To address the OR-splits, an aggregation model must know which paths are taken into account for this split. To determine the upper and lower bound for QoS values all combinations of possible splits must be considered. The result is a set of possible combinations. For example, if a 2-out-of-5-OR-structure has the following elements $\mathbb{K} = \{x_1, x_2, x_3, x_4, x_5\}$, then the possible combinations would be:

$$\mathbb{K}_{\mathcal{C}} = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \ldots, \{x_4, x_5\}\}$$

In general, the resulting set $\mathbb{K}_{\mathcal{C}}$ contains all relevant combinations of the tasks depending on the OR-split and -join semantics. Then, an aggregation function can evaluate the resulting QoS of individual combinations. To simplify the notation in the tables, a notation is defined that a function $f_{\mathcal{C}}$ is applied to the relevant combinations of elements represented by individual sets in $\mathbb{K}_{\mathcal{C}}$:

$$f(f_{\mathcal{C}}(\mathbb{K}_{\mathcal{C}})) := f\Big(\{f_{\mathcal{C}}(\mathbb{S}_{sub}), \forall \mathbb{S}_{sub} \in \mathbb{K}_{\mathcal{C}}\}\Big)$$

where $\mathbb{S}_{sub}$ denotes an arbitrary combination in $\mathbb{K}_{\mathcal{C}}$.

The introduced model presumes that a transformation transforms the flow description of a composition into this model of composition patterns. Upcoming Section 4.5 will discuss such transformations further. Once such a representation exists, a selection process can perform the aggregation of QoS categories based on the QoS of the individual tasks.

Regarding the related work, other ideas and proposals for patterns exist to describe the structural arrangements of service compositions. Specifically for the compositions of services, Yang et al. have proposed a set of structural patterns to define the execution flow characteristics of a service composition [155]. Florescou et al. have developed an XML-based programming language to define execution statements to form characteristics of Web services [33]. WS-BPEL [126] can be seen as a general purpose flow language which represents also a structural model of a flow.

Besides the flow or workflow languages, the Object Management Group (OMG) has proposed a Unified Modelling Language (UML) which also provides elements for expressing a flow structure. The UML provides the activity diagram that services this purpose [100]. Activity diagrams contain also the concepts of sequences, parallel splits and joins. All these languages that were identified for describing compositions have in common that they cover different purposes than the aggregation of QoS. Such purposes can be, for example, the definition of a control flow in order to parameterise an execution environment. The composition patterns instead represent a set of structural elements for modelling compositions specifically designed for the aggregation of QoS values derived from the workflow patterns. The next sections will explain the aggregation for a selection of QoS characteristics, which were introduced in Section 3.2: throughput, response time, cost, availability and encryption grade.

4.4.1 Aggregation of Throughput

The throughput of a service denotes the amount of processable data per time unit. Usually, the throughput is given in requests per second and is interpreted as an increasing dimension, which means that a higher value denotes a better quality. The throughput can also denote the processable amount of data within a given time. However, the request per time matches better the characteristics of a service invocation and is thus preferred over the data per time. Otherwise the question rises if all services with different functionality in the composition process the data in a comparable way. Then, a detailed discussion for each case is necessary.

In a sequence, the node with the lowest value assigned determines the throughput for the aggregated property. In a parallel arrangement, it is possible that processable data splits among the parallel tasks. In this case, the aggregation of throughput would result in addition of the throughput values of the individual services. It is also possible that individual requests are split among the tasks. This would result in an addition of the given values as well. However, such an arrangement presumes that all the parallel services provide the same functionality. If the scenario is to implement a business model, it can be presumed that such redundancies are not taken into account on the level of the business model. Thus, such a redundant arrangement can be considered on an optional basis.

For the aggregation of throughput in parallel cases, the set of starting services is considered relevant. Otherwise, the server will refuse additional requests, if the limit of maximum possible requests is exceeded. Therefore, to ensure the successful execution of the composition, all services must be fully capable of accepting the occurring invocations. The aggregation of throughput is shown in Table 4.2. The given rules are based on the assumption that a service provider provides statements about the guaranteed minimum and maximum throughput.

4.4.2 Aggregation of Response Time

Contrary to the throughput, the response time is a decreasing dimension, meaning that a lower value is preferred. As mentioned in Section 3.2, the response time and latency or the execution time are used as synonyms. The response time of a service can be defined as the sum of time to transfer the request and response over a network and the time for queuing the request on the provider side (latency), and

# Comp. Pattern	Maximum Throughput	Minimum Throughput
1 Sequence	$x_a = min\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
2 Loop	$x_a = x$	$x_a = x$
3 XOR-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
4 AND-AND	$x_a = min\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
5 AND-N/M	$x_a = min\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
6 AND-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
7 OR-OR	$x_a = max\Big(min(\mathbb{K}_C)\Big)$	$x_a = min\{x_1, \ldots, x_k\}$
8 OR-N/M	$x_a = max\Big(min(\mathbb{K}_C)\Big)$	$x_a = min\{x_1, \ldots, x_k\}$
9 OR-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$

Table 4.2: Aggregation rules for throughput.

the time to process the request (cf. Ran, introduced in Section 3.2.1):

$$t_{total} = t_{transfer} + t_{queuing} + t_{processing}$$

The entire time needed to perform a task, i.e. t_{total}, is considered for this work. Applying the method to the aggregation, a definition for lower and upper bounds is shown in Table 4.3. In a sequence and a loop, the time is determined by the sum of the values of each service invocation. In a sequential case, the definitions for lower and upper bounds are the same. Because the response time represents a decreasing measure, the largest value in a parallel arrangement of services denotes the worst case. The nature of the response time also demonstrates that considering the same split operation (AND), different join operations result in different aggregation rules. To determine the minimum response time in the AND-AND case, the largest value of all involved services denotes the overall minimum value. However, for the AND-XOR pattern, the smallest represents the relevant value.

4.4.3 Aggregation of Cost

The cost of a service represents a measure for the resources consumed by a service execution. Therefore, the cost can be seen as a decreasing dimension, where a lower value denotes better quality. For the cost, two main models are possible: For example, Tosic et al. mention two payment methods: a pay-per-use use model that considers the number of service invocations and a subscription model that allows consumers to invoke a service on a flat-rate basis [133]. For the aggregation of cost, it is important that all statements refer to the same payment model.

Table 4.4 shows the aggregation definitions for the upper and lower bounds of the cost. Contrary to the response time, an aggregation rule must take all started services into account, regardless of whether they are relevant for the join operation or not. To cover the OR-split statements, the function $sum(\ldots)$ denotes to add the elements in each of the sets in $\mathbb{K}_C$.

# Comp. Pattern	Maximum Response Time	Minimum Response Time
1 Sequence	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
2 Loop	$x_a = lx$	$x_a = lx$
3 XOR-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
4 AND-AND	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = max\{x_1, \ldots, x_k\}$
5 AND-N/M	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
6 AND-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
7 OR-OR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\Big(max(\mathbb{K}_\mathcal{C})\Big)$ [1]
8 OR-N/M	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\Big(max(\mathbb{K}_\mathcal{C})\Big)$ [1]
9 OR-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$

[1] *For example, considering three tasks out of five, the min of the 3rd quickest is relevant.*

Table 4.3: Aggregation rules for the response time.

# Pattern	Maximum Cost	Minimum Cost
1 Sequence	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
2 Loop	$x_a = lx$	$x_a = lx$
3 XOR-XOR	$x_a = max\{x_1, \ldots, x_k\}$	$x_a = min\{x_1, \ldots, x_k\}$
4 AND-AND	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
5 AND-N/M	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
6 AND-XOR	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
7 OR-OR	$x_a = max\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$	$x_a = min\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$
8 OR-N/M	$x_a = max\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$	$x_a = min\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$
8 OR-XOR	$x_a = max\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$	$x_a = min\Big\{sum(\mathbb{K}_\mathcal{C})\Big\}$

Table 4.4: Aggregation rules for the cost.

4.4.4 Aggregation of Availability and Reliability

Also discussed in Section 3.2, the terms availability and reliability show similarities. The UML QoS-Profile discusses both the reliability and the availability. The author Ran discusses the availability as a characteristic denoting whether a service is present and ready for immediate use (cf. Section 3.2.1). Then, the availability is defined by the quotient of the available time divided by the total time considered [112]. For this discussion, the following definition of availability for *repairable systems* is considered:

$$A = \frac{MTTF}{MTTF + MTTR}$$

In this equation, $MTTR$ denotes the mean time to repair and $MTTF$ denotes the mean to failure, defined as $MTTF = \int_0^\infty R(t)dt$. In this definition the function $R(t)$ defines the probability that a system fails against the time.

This concept of availability represents an increasing dimension and according to this definition cannot, it exceed the value of 1. The availability is presented as a probability, because a service exporter cannot predict whether a failure will happen with the first or 1000th service invocation. As a consequence, for the aggregation, the values are also taken as probabilities. For a sequence, this means that the aggregated availability or reliability results from the multiplication of the individual values. In parallel cases, all invoked services are relevant for the availability, while for the reliability only the services required for the synchronising operation are considered. Table 4.5 shows the aggregation rules for the availability. To cover the OR-split statements, the function $prod(\ldots)$ denotes to multiply the elements in each of the sets in $\mathbb{K}_C$.

# Comp. Pattern	Availability
1 Sequence	$x_a = \prod_{i=1}^{k} x_i$
2 Loop	$x_a = x^l$
3 XOR-XOR	$x_a = min\{x_1, \ldots, x_k\}$
4 AND-AND	$x_a = \prod_{i=1}^{k} x_i$
5 AND-N/M	$x_a = \prod_{i=1}^{k} x_i$
6 AND-1	$x_a = \prod_{i=1}^{k} x_i$
7 OR-OR	$x_a = max\left\{ prod(\mathbb{K}_C) \right\}$
8 OR-N/M	$x_a = max\left\{ prod(\mathbb{K}_C) \right\}$
9 OR-N/M	$x_a = max\left\{ prod(\mathbb{K}_C) \right\}$

Table 4.5: Aggregation rules for the availability.

4.4.5 Aggregation of Reputation and Fidelity

Zeng et al. have defined a reputation based on the idea of a user-given ranking, like the online store Amazon allows ranking of products or the auction platform ebay allows ranking of the users [159]. Furthermore, as mentioned in Section 3.2.2, the reputation can be also considered as a measure to capture non-functional service selection criteria that are usually not expressed with quantifiable measures. One reason why the reputation is considered for this thesis is its characteristic: The reputation represents an increasing dimension that might be an absolute value (contrary to the availability) but an aggregation rule cannot summarise the values in a sequential arrangement for example. Thus, it must be aggregated differently as discussed for the other QoS categories. Kalepu et al. have discussed the reputation in the sense of the fidelity concept in detail [69]. According to their discussion, a general concept of reputation suffers from the fact that such a measure depends on the subjective evaluation of individual users. Such a metric would benefit from a comprehensive approach which eliminates subjective and counterproductive ratings.

The fidelity denotes a measure for the overall quality of a service execution. A formal, general definition of the fidelity is not given here, because the nature of the fidelity depends on many aspects, such as the purpose of the composition, the application domain, the nature of the output etc. Modelling and measuring the fidelity has been discussed for workflow applications (cf. Cardoso et al. [17]). Compared with the reputation, the fidelity represents a more precisely defined measure whereas the reputation asks for a plain judgement of individual consumers.

For the reputation, different aggregation rules are possible. For this thesis, the idea has been considered that the reputation represents the mean average of the ranks given by individual users. This is also the interpretation of Zeng at al. [159]. Consequently, the aggregation of reputation values should represent the average of the individual services. It is assumed that such an aggregation rule would also suit an aggregation of the fidelity. Other definitions can consider that the worst reputation value is relevant. For example in ebay, users tend to focus on the negative ratings while almost ignoring the positive ones. Thus, it is also possible to aggregate reputation by considering the worst values. Table 4.6 shows the aggregation rules based on the mean-value consideration.

4.4.6 Aggregation of Encryption Grade

The security can be interpreted in many different ways, as the UML QoS-Profile has explained. Most of the characteristics show a binary nature: Either the service supports a security characteristic (for example, an authentication method, facilities to ensure confidentiality etc., as discussed by Ran [112]), or the service does not. The aggregation of these binary characteristics does not require much convention that would require a particular aggregation method: If a security characteristic is required, each of the involved service candidates must provide this characteristic.

The only characteristic that can be described with numerical values is a measure that determines the length of a key that is used for encryption. Encryption algorithms use a key either to encrypt data or to create an electronic signature. As a rule of thumb, the longer an encryption key is the more difficult it is to break the encryption or to damage the signature. Consequently, this characteristic shows an increasing direction where a larger size denotes better quality.

For the aggregation of the encryption level in sequential patterns, the weakest key is considered significant. In the parallel case, the encryption level is a non-functional characteristic, which must be fulfilled by all significant parallel nodes in the same manner. However, if the node does not fulfil a demanded level of encryption, the execution would be worthless. Thus, this dimension is discrete and the aggregation based on, for example, the average of involved services cannot be applied. The aggregation for the encryption is listed in Table 4.6.

# Comp. Pattern	Reputation	Encryption Grade
1 Sequence	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
2 Loop	$x_a = x$	$x_a = x$
3 XOR-XOR	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
4 AND-AND	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
5 AND-N/M	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
6 AND-1	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
7 OR-OR	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
8 OR-N/M	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$
9 OR-1	$x_a = \frac{1}{k}\sum_{i=1}^{k} x_i$	$x_a = min\{x_1, \ldots, x_k\}$

Table 4.6: Aggregation rules for the mean reputation and the encryption level.

4.5 Support of Un-Structured Models

The structural model that is used by the aggregation method is not capable of expressing all possible structures of control flows. As mentioned in Section 4.2, the structural model considered by the aggregation method is equivalent to a structured workflow model (SWM), discussed by Kiepuszewski et al. [73]. However, flow structures are possible that do not conform to SWMs. Kiepuszewski et al. name them arbitrary workflow models and they define the following characteristics for them:

1. An arbitrary workflow consists of sets of process elements (i.e. nodes in a graph) and sets of transitions between process elements (i.e. edges in a graph). Two types of process elements exist: tasks and a set of routing elements which are similar to the composition patterns.

2. Regarding the routing elements that split the flow, AND- and OR-splits are considered. The AND-split has the same semantic as introduced for the composition patterns and the OR-split has the same semantic as the introduced XOR-split.

3. Regarding the routing elements that join paths, AND- and OR-joins are considered analogously. The OR-join has a different semantic compared with the XOR-join: For each incoming path, the routing element starts one instance of the following flow. The AND-join has the same semantic as the AND-join of the composition patterns.

4. Similarly to the composition patterns, process elements that do not have incoming paths represent initial items and process elements that do not have outgoing paths represent final items.

5. Furthermore, Kiepuszewski et al. consider the following additional assumptions: For example, all initial activities are supposed to start concurrently, and the workflow ends when all ending process elements have been reached, following an “until there is nothing else to do”-idea.

The main characteristic of an arbitrary workflow model is that the process elements that split and join different paths can be arranged with fewer restrictions compared with an SWM. Considering WS-BPEL as an example, the flow and link elements allow forming structures that cannot be expressed as an SWM. Contrary to these two elements, WS-BPEL also includes elements named structured activities that comply with an SWM. Besides WS-BPEL, the workflow patterns by van der Aalst et al. indicate that the arbitrary models are considered in the workflow domain, for example, as indicated by the arbitrary loop pattern. The advantage of allowing such arbitrary links between tasks and routing elements is obviously a better expressiveness in terms of which structural arrangements can be modelled. The downside of allowing such arbitrary models is that an invalid structural model can lead to deadlocking situations when the workflow or the composition is executed.

The arguments used in discussion about arbitrary workflows versus SWMs is related to the classic discussion on the use of $go\ to$-statements in program code. The discussion has influenced the design of programming languages, because arbitrary jumps in a program flow were considered “harmful” (cf. Djikstra [23] and Knuth [74]): Allowing $go\ to$-statements would tempt programmers to write unstructured, error-prone and hard-to-verify program code. Moreover, many cases were identified for which programmers use $go\ to$-statements, but could have used more appropriate structures to express an algorithm (e.g. a $while$-statement). Knuth drew the conclusion that it depends on individual application cases whether $go\ to$-statements are useful and offer advantages or they can be omitted by more appropriate programming constructs [74]. The same considerations apply when discussing advantages of SWMs over arbitrary workflow models.

This section will discuss examples that require the use of arbitrary workflow models. In the following, three main example cases of arbitrary workflows are presented that do not conform to an SWM: *1)* a splitting or joining routing element does not have a corresponding closing element, *2)* arbitrary loops or $go\ to$-statements and *3)* nested patterns where the flow from one pattern is directed inside another pattern.

4.5.1 Open Elements

The first case occurs if routing elements are not used in pairs or if corresponding routing elements do not form pairs in terms of the number of outgoing and incoming paths. The graph (a) shown in Figure 4.4 represents such a constellation: An AND-split element has two outgoing paths and no corresponding join element is provided. In a similar case, the open AND-split has a number of outgoing paths, but a corresponding AND-join element has less incoming paths. Because of the missing joining element or path in this example, the pattern-wise aggregation cannot be performed.

It must be noted that the diagram (b) shown in Figure 4.4 has the same pattern of an open AND-split. In this case the solution would be to add the two missing joining elements. This modification is possible, because a part of the definition for arbitrary workflows presumes that all final activities must end, in order to determine the end of the entire flow. Thus, adding the corresponding AND-splits would not change the operational behaviour. Obviously, the problem of open AND-splits occurs if this element is placed within another parallel element. In addition, the problem can occur with open parallel joining elements in the way that the composition starts at different at once. For these cases it is assumed that all starting nodes will be invoked at the same time. A corresponding example is shown in graph (c) of Figure 4.4.

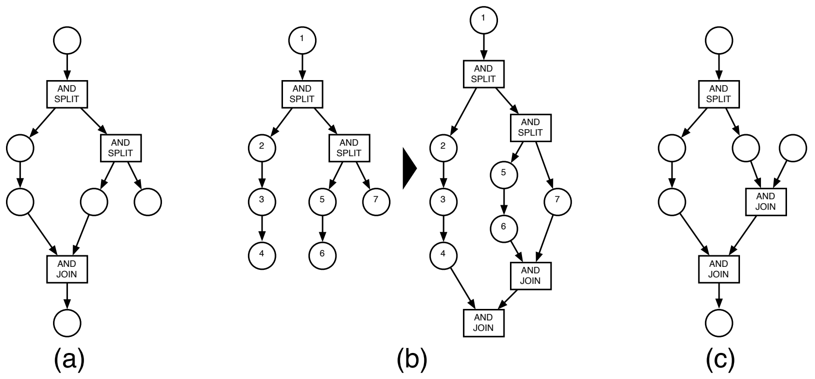


Figure 4.4: Examples of open parallel structures.

4.5.2 Arbitrary Loops

In a second case, loops are used in an unstructured manner. Such loops can point from inside of one pattern element into another pattern element. The diagram (a) shown in Figure 4.5 has a loop coming from a parallel structure pointing back to a place outside this structure. The pattern-wise aggregation cannot take place because of the open AND-split and the open XOR-join structure.

Contrary to that, a loop that occurs inside another pattern still conforms to an SWM. The graph (b) in Figure 4.5 shows such a structure. A possible transformation into a structured model can be obtained by replacing the split and join routing elements with the loop pattern. This case shows that a loop only becomes problematic for structured models if the enclosing pattern is left.

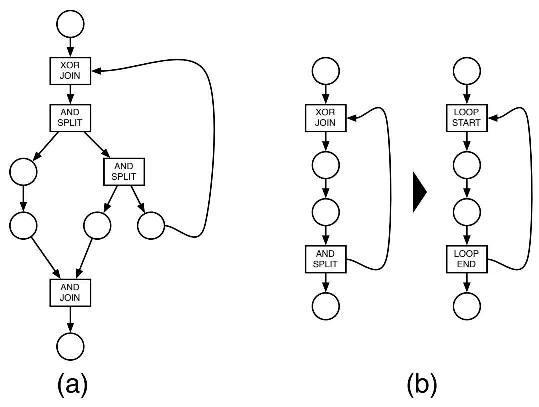


Figure 4.5: Examples of loops.

4.5.3 Nested Patterns

As the third case, a path can leave “its” pattern and point to a joining element of another pattern. Figure 4.6 shows an example of such a structure with three models (a), (b) and (c). The example given in (a) has been discussed by Kiepuszewski et al. [73] and by van der Aalst [144] and it serves as the classic case when it comes to the discussion of expressiveness among workflow or composition languages. All splitting and joining routing elements have a corresponding counterpart and all elements have two outgoing or incoming paths respectively.

The graph (b) also represents a special case, which Kiepuszewski et al. discuss as a specific overlapping structure. The graph (c) gives also an example for an occurring deadlock: The main two paths are never followed at the same time but both are required when the flow arrives at the lower AND-join elements. As a result, this structure results in a deadlocking situation. In all three cases, a pattern-wise aggregation cannot be performed, because every pair of AND-split and -join elements shows a dependency to an element outside of the pattern.

4.5.4 Transformations to Structured Workflow Models

The three cases introduced above represent structural arrangements that cannot be modelled with SWMs. As a consequence, the proposed aggregation model cannot be applied. Kiepuszewski et al. [73] have proposed transformations that convert arbitrary workflows into equivalent SWMs in order to cope with this problem. Their approach is to utilise node duplication and the use of auxiliary variables to form additional conditional branches. With node duplication, the invocation of tasks in the composition is duplicated. The authors have shown the equivalence of an arbitrary workflow to a corresponding SWM by using a technique named bi-simulation. By applying the bi-simulation technique, the equivalence of two graphs can be determined by a node-wise comparison of the possible states. Kiepuszewski et al. have

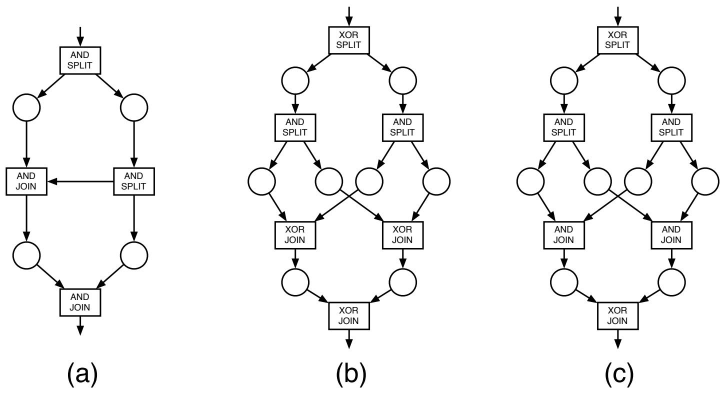


Figure 4.6: Examples of nested patterns.

shown the following:

- **A transformation is always possible** if an arbitrary workflow model does not contain any tasks executed in parallel. For this case, only splits are allowed that branch into exactly one flow. The equivalent structure of the composition patterns is the combination of XOR-split and -join elements. Transformations can also applied to structural models that contain an arbitrary loop.

- **A transformation is not always possible** if a model contains a parallel split or join element without its corresponding element or that contains nested structures. The authors explain that only models that contain parallel structures and that overlap in a predefined way (as shown in graph *(b)* of Figure 4.6), or do not merge at some point (as shown in graph *(b)* of Figure 4.4) can be transformed into an SWM.

Transforming Parallel Cases to Structured Workflow Models

Kiepuszewski et al. have distinguished six sub-cases of parallel flows. For each it was discussed whether a transformation to an SWM is possible or not. From these six cases, they have identified four which allow deadlocking flows. Such structures cannot be transformed into an SWM until this deadlock potential has been eliminated. The remaining two cases are named parallel exits (cf. Figure 4.4 *(a)*), and a synchronised entry into a parallel structure (cf. Figure 4.4 *(c)*). For these two structures, Kiepuszewski et al. have shown that no transformations into an SWM are possible.

The case of open parallel structures and the case of arbitrary loops represent such parallel exits or synchronised entries. They can be modelled as a nested arrangement of parallel structures. For the case of the open parallel structures, it is assumed that all starting tasks start at the same time. For the ending tasks, it is assumed that all need to end in order to finish the entire composition. Then, an AND-split or an AND-join respectively can be added as it is shown in Figure 4.7 *(a)*. The result is a composition structure that shows nested parallel routing elements. This nested structure is still not compatible with the proposed aggregation method and thus, the aggregation cannot be performed.

In the case of occurring loops, a loop structure can be unrolled. This is also named "loop unwinding". By loop unrolling, the looped part of the execution flow is written repeatedly for the amount of loops. Of course, this also presumes that the amount of loops is known at design time or at least a maximum number of loops can be guessed. In the previous section, it was explained that knowing the number of occurring loops in advance is necessary to perform the QoS aggregation in general. Therefore, only this case is considered for the aggregation model. If the number of loops is known, then the loop can be unrolled: The repeated part is added to the graph as shown in 4.7 (b). The loop case can also be seen as a variant of the case of nested parallel routing elements. And as a consequence, the aggregation cannot be applied as well.

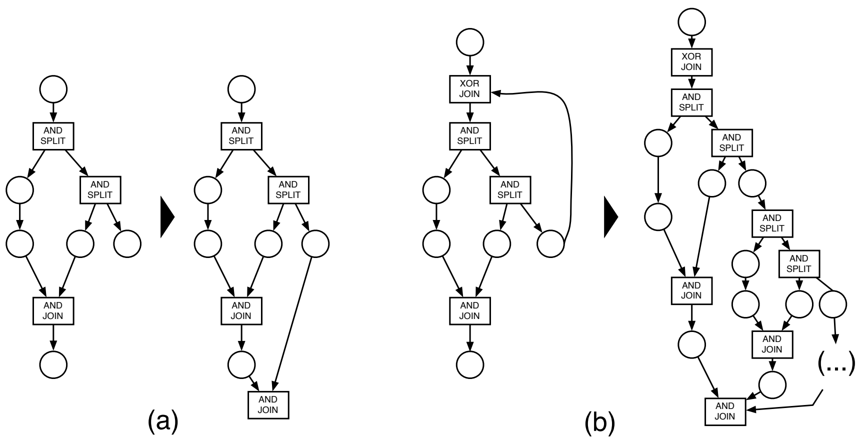


Figure 4.7: Open parallel and arbitrary loop structures and possible transformations.

Kiepuszewski et al. discuss the classical example of nested parallel elements, which is shown again in graph *(a)* of the Figure 4.8: In this graph, two pairs of sequential tasks are executed in parallel. In addition, after the task *1* has been finished it synchronises with the first task of the other path only. For this case, Kiepuszewski et al. have proven that it is not possible to transform the arbitrary workflow into an SWM with equivalent functional behaviour, because the parallel paths have inter-relating dependencies.

It must be noted that conditional branches do not result in this difficulty: The case shown in graph *(b)*, without parallel but with conditional branches, can be

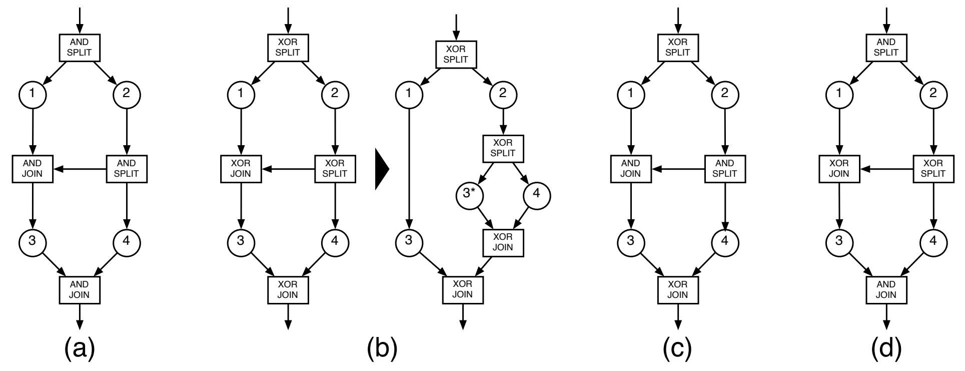


Figure 4.8: Structures that contain nested parallelism.

transformed into a structured model using node duplication: The first XOR-split defines that only one of the outgoing paths is started. Thus, the first XOR-join after task *1* always has only one possible incoming path. Then, the middle XOR-join could be dropped and to the XOR-split element, a corresponding XOR-join element with a duplicated node *3* can be added. Because the given aggregation rules consider only the selection of a minimum or a maximum value, the additional node will not change the result of the aggregated QoS as compared with a manual inspection of the QoS, which would involve the paths *(1,3)*, *(2,3)*, *(2,4)*.

The example shown in graph *(c)* represents a case which would result in a deadlock, because the AND-join routing element would wait for both incoming paths, but only one of them would have been started before. A similar example is given by graph *(d)* where the given structure can result in a deadlock: It could happen that the right path does not arrive at the bottom AND-join element. Both examples do represent valid workflow models. However, their execution can result in erratic behaviour. Both examples also show the disadvantage of using arbitrary structures: It is possible that deadlocking flows can be defined while structured models prevent any irregular behaviour.

Transforming Parallel Cases to Non-Equivalent SWMs

The goal of the discussion given by Kiepuszewski et al. is to provide transformations that result in functional equivalence compared with the original arbitrary workflows. This is a different goal from what is required for the aggregation method: For the aggregation, structural models are required that result in suitable QoS statements referring to the original arbitrary model. Thus, not the functional equivalence is relevant for transformations between arbitrary workflows and SWMs. The proposed transformation focusses on the main example given in Figure 4.8(a), because of its relevance for loop-unrolling and open parallel structures.

The proposed transformation is outlined in Figure 4.9 with two examples (a)

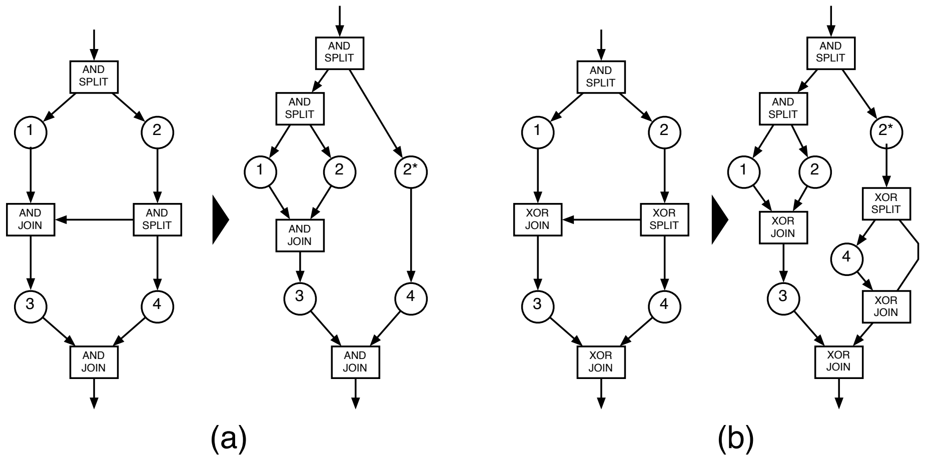


Figure 4.9: Structures that contain nested parallelism and possible transformations.

and (b). The transformation is performed as follows: For each branch that defines a dependency, concurrent paths are created that have duplicated nodes. Of course, applying the aggregation method to the resulting SWM will result in a different QoS, because the duplicated node also results in a duplicated QoS. For example, if the cost is relevant, the corresponding QoS statement is counted twice and thus the resulting cost is higher than in reality. Thus, for each QoS characteristic that is based on an arithmetic operation (e.g. the addition or the multiplication), the aggregated QoS statement must be corrected by neutralising the corresponding duplicated node. The conclusion is that such a transformation is possible. Nevertheless, performing the QoS aggregation by this way is more complex than considering SWM-compatible structures only.

Conclusion

For cases where the composition structure does not conform to an SWM, Kiepuszewski et al. have shown transformations from specific cases of arbitrary workflow models into SWMs. Consequently, the aggregation method can be also performed for these cases. In summary, the introduced transformations allow the support of the following cases by the aggregation method:

- **Arbitrary workflow models without parallelism.** Flow models are supported that conform to the definition of arbitrary workflows and do not contain any tasks that are executed in parallel. For these models, transformations exist that result in an SWM.
- **Arbitrary workflow models with structured parallelism only.** If parallel structures occur in a workflow model, they must form an independent sub-

part that conforms to the definition of an SWM. For these models, transformations can be also applied to achieve a model that can be entirely reduced into one statement.

- **Arbitrary workflow models with special cases of unstructured parallelism.** Special cases of parallel flows that do not conform to SWMs can be transformed into an SWM. The work of Kiepuszewski et al. has discussed such special cases of unstructured parallelism and showed corresponding transformations [73].

Regarding the remaining structures, in particular the nested parallelism, a transformation is possible that would result in a functionally different SWM. In addition, the aggregation method would also require a modification to process the QoS on the resulting model. Such a method raises the efforts to perform the aggregation method. The decision that the aggregation method only covers SWMs and the above mentioned three cases of arbitrary workflow models represents a compromise between support of possible flow models and feasible QoS aggregation.

4.6 Related Methods for Quality-of-Service Aggregation

The work of Puschner and Schedl was already mentioned. Besides their own approach, the authors mention also a couple of related research efforts in the field of software development for real-time environments [111]. Contributions to this field either presume a structured model like SWMs to allow the computation of response times or propose specific approaches that cover the response time and a measure for reliability. Thus, these approaches were not regarded as feasible as a foundation for an aggregation method that covers different QoS characteristics in service compositions.

Lee discusses the problem of QoS-based selection to form compositions [78]. To achieve QoS statements for the composition, he proposes aggregating cost and response time by the addition of individual values. This approach does not result in wrong statements. However, a structure-aware model leads to a more appropriate statement that will be closer to the delivered QoS during run-time. For example, only one of the values in parallel XOR-structures is relevant instead of the added total. An addition of all the response time values presumes that all tasks in the composition will be executed in a sequential manner. Thus the proposed aggregation by forming a sum is limited to covering this case. Yu and Lin [158] also consider a sequential structure, which includes either a sequence of tasks or, as they mention, a service composition that shows a sequence of invocations that is considered to be relevant for the composition. As explained for the aggregation method by Lee, the resulting QoS statements might not be as close to the run-time QoS as the values obtained by the pattern-based aggregation for parallel structures.

Zeng et al. present a framework that covers many aspects of forming compositions while considering the QoS as selection criteria [159]. In their work, they use a more detailed method than the aggregation proposed by Lee. They discuss a set of QoS characteristics, but they also make clear that their approach is not limited to these. Their aggregation method specifically considers the execution duration, which is equivalent to the response time mentioned in this thesis. In their approach, an algorithm identifies a critical path that shows the longest time for the given setup. Then, the addition of the individual QoS values is taken as the result for the overall execution of the composition. For the other QoS characteristics, the proposal is to consider the QoS values of the tasks that belong to the critical path. For example, to aggregate values for availability, the product of the tasks of the critical path is relevant. This approach is justified with the assumption that tasks which do not belong to the critical path will also not affect the resulting QoS during run-time.

Although this method might deliver appropriate results, an example can be given where the pattern-based aggregation will lead to a more precise aggregated statement. This example is shown in Figure 4.10: An arrangement of four tasks is given from which only one will be invoked at a time (the XOR-split). In this arrangement, a critical path algorithm considering the execution or response time would identify the path with task 2. Consequently, the resulting availability for this statement is 0.98. If it is assumed that the other paths will be taken with a reasonable frequency, the resulting availability will be lower than 0.98. Thus, the proposal to consider the average mean results in a statement which is more likely to reflect the QoS during run-time.

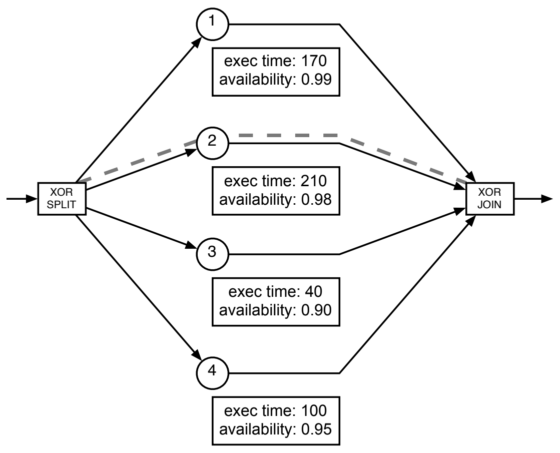


Figure 4.10: Example for the aggregation based on a critical path.

A possible extension to the proposed aggregation method by Zeng et al. would be to apply the critical path algorithm not only to the execution or response time but also to the other QoS characteristics. Based on the critical path found for each QoS characteristic, the product or the sum of the individual values will also lead to similar results as the pattern-based aggregation does. However, this approach leads to the problem that, for each considered QoS characteristic, an algorithm must be designed that identifies the critical path. This is considered to be more complex than the definition of aggregation rules for structured patterns.

Cardoso discusses the aggregation of QoS in the context of workflows [15]. He proposes a structured model as well and accordingly the aggregation method is based on defined sets of structural patterns. Their contribution also includes the probability of invocations in conditional parallel structures, which is similar to the monitoring approach discussed in the previous section. The approach of Cardoso does not cover the m-out-of-n-joins (and splits) as mentioned among the workflow patterns by van der Aalst et al. By using only AND- and XOR-elements, the semantics of the m-out-of-n joins can be simulated as van der Aalst et al. have shown [146]. However, this equivalent structure, achieved by using the simpler routing elements, represents an arbitrary workflow model and cannot be covered by the aggregation method that presumes an SWM. Thus, these additional structures were added to cover also these patterns. Then, the resulting QoS statement for the composition will be more precise.

4.7 Aggregation for Quality-of-Service Monitoring

The pattern-based aggregation method can be also applied to values that result from monitoring the QoS of the composition during run-time. With monitoring functionality, a retailer can capture the resulting QoS when the particular services are invoked. For this monitoring process, a centralised, mediator-based structure is presumed. Different setups of compositions are ignored for this discussion, such as the brokered composition or a composition in a peer-to-peer environment as described by Hull et al. [48]. The research conducted in the field of monitoring workflows is also called workflow history management. Koksal et al. discussed the motivations for monitoring workflows, and among the main reasons the following apply as motivation in the field of service compositions [75]:

- **Recovery and Balancing.** Changes in the delivered QoS are inherent characteristics of open distributed systems [35]. Service importers want to detect QoS violations during the run-time and establish appropriate recovery activities. For example, a service invocation might take longer than guaranteed and then compensation activities must be established. From the perspective of the service exporter, another motivation for monitoring the QoS is to ensure different QoS parameters negotiated with different exporters. For example, a service exporter must prioritise between different service importers in case of shortages.

- **QoS Analysis.** If metrics about the response are captured, it is possible to check whether services are delivered with the desired QoS or not. However, monitoring the QoS only applies to metrics that can be directly derived from the execution, such as response time or availability. For example, capturing the cost during the execution depends on the payment method (flat-rate vs. per-request).

  The response time of the invocation of individual services can be used to analyse the performance. Then, the main performance bottlenecks of the composition can be identified. Regarding other QoS characteristics, the monitored invocation frequency of particular tasks in conditional branches can lead to a more precise aggregation of other QoS characteristics, such as the availability.

- **Controlling.** The intended application of service compositions is the realisation of an entire business process or parts of it. Financial controllers can benefit from up-to-date information about the progress or running instances of the composition. By the aggregation of the cost it can be determined, how many resources were consumed and how many resources might be consumed within a given time.

To establish recovery tasks when errors occur during the execution or to check the delivered QoS of individual services, the aggregation of QoS values is not required. However, to analyse the performance of the composition, and to predict future characteristics for controlling purposes, the proposed aggregation of QoS values can be performed to deliver a more accurate estimation of the delivered QoS.

In order to capture the delivered QoS in a Web service environment, different approaches that share the mentioned motivations already exist. Then management infrastructure WSOI developed by Tosic et al. provides the monitoring of the QoS and that facilitates the exchange of services in case of unfavourable QoS [132]. This contribution targets the first motivation mentioned for monitoring. Technically, the functionality is achieved with an additional management tier on the side of the service exporter. A retailer can use such a facility to adapt the QoS parameters delivered to the consumer. Erradi and Maheshwari propose a system with similar objectives, but contrary to the work of Tosic et al., they promote a separate broker-oriented infrastructure that can cover services of different exporters [28]. This approach results in a new role independent from the retailer and the 3rd party service provider. Dogdu and Mamidenna specifically discuss such an independent broker component that performs the invocation instead of the service importer [25]. With such an arrangement, this component can specifically monitor the QoS and perform balancing among the pending requests, to provide an efficient scheduling of invocations.

Baresi et al. have also discussed the monitoring of the QoS in service compositions [5]. They propose an infrastructure consisting of software components that a retailer would use. In addition, they provide modules that 3rd party service providers can integrate into their service infrastructure in order to stay compatible with the monitoring environment of the retailer.

4.7.1 Aggregation of Mean Values

The difference between the aggregation during the modelling or the design time and run-time lies in the nature of the computed values: The QoS aggregation during design time provides proper upper and lower limits of a QoS characteristic, like cost or response time. For QoS categories that require a mean-oriented aggregation, the aggregation performed at design time represents an estimation. However, to properly calculate mean values from the given individual QoS statements, the aggregation rule must know the distribution of task executions in conditional branches in advance.

During run-time, monitoring facilities can also capture the distribution of service executions in branching structures. Based on monitoring data, an aggregation algorithm can consider the invocation occurrences to apply a weight to the individual values. Then, the aggregation would result in an extrapolation of values in parallel structures. For the given aggregation rules in Table 4.7, the weighted mean for parallel patterns is considered. The definition distinguishes between the weight of each value in the split operation g_i and the weight of each value in the join operation h_i, where the index i refers to the individual service. The aggregation rules for mean response time and mean cost are given in Table 4.7. To cover the OR-split statements, the function $sum(\ldots)$ denotes to add the elements of the enclosed set, if this set contains the values. Or it denotes to result in individual sums from the enclosed set of sets (in the case of $sum(\mathbb{K}_\mathcal{C})$).

The aggregation of mean response time represents a special case: An appropriate approximation for all parallel patterns except the XOR-XOR combination cannot be given. The following example explains the reason: A parallel AND-AND arrangement contains three services. In this case, the distribution for the split and join condition does not matter. It is assumed that all the services execute either in 3 or in 7 seconds – each by a probability of 50%. The aggregation for this parallel arrangement would result in the value 5. Considering the setup, it is likely (at a probability of $0,875\%$) that at least one of the services would need 7 seconds, so the mean for the aggregation should result in a value closer to 7 than the value 5.

This example shows that for characteristics that have aggregation rules depending on the join operation (as the response time), an aggregation method must take the resulting distribution of the provided QoS into account or as an alternative, it must capture the delivered QoS directly. This applies to all parallel arrangements with a synchronising join involving more than one service. The average mean of the given mean values would lead to a useless calculation. Thus, the proposed aggregation rules can only refer to the upper and lower limits of response time. In addition, the weighted mean must be interpreted as a statistical measure and thus the aggregation does not make sense for just covering very few executions.

# Pattern	Mean Response Time	Mean Cost
1 Sequence	$x_a = \sum_{i=1}^{k} x_i$	$x_a = \sum_{i=1}^{k} x_i$
2 Loop	$x_a = lx$	$x_a = lx$
3 XOR-XOR	$x_a = \frac{1}{k} \sum_{i=1}^{k} g_i x_i$	$x_a = \sum_{i=1}^{k} g_i x_i$
4 AND-AND	*(see text)*	$x_a = \sum_{i=1}^{k} x_i$
5 AND-N/M	*(see text)*	$x_a = \sum_{i=1}^{k} x_i$
6 AND-1	*(see text)*	$x_a = \sum_{i=1}^{k} x_i$
7 OR-OR	*(see text)*	$x_a = \frac{1}{\lvert\mathbb{K}_C\rvert} sum\Big(sum(\mathbb{K}_C)\Big)$
8 OR-N/M	*(see text)*	$x_a = \frac{1}{\lvert\mathbb{K}_C\rvert} sum\Big(sum(\mathbb{K}_C)\Big)$
9 OR-1	*(see text)*	$x_a = \frac{1}{\lvert\mathbb{K}_C\rvert} sum\Big(sum(\mathbb{K}_C)\Big)$

Table 4.7: Aggregation rules for the mean response time and the mean cost.

Chapter 5

Quality-of-Service-based Selection of Services

The introduction in Chapter 1 has presented the abstract parts of service trading as specified by the RM-ODP. Performing these parts subsequently narrows down the set of available services in order to identify the optimal selection of services to perform the tasks of the composition. According to the trading specification, these parts are as follows: *1)* a keyword-based search among the available services for a rough pre-filtering, *2)* a matchmaking process identifying services that match the required functionality to perform the according tasks, *3)* a selection process that assigns one candidate for each task in accordance with the selection criteria, i.e. non-functional or QoS requirements, and *4)* the application specific of trading policies.

This chapter will discuss the third step of these four. It will discuss the selection process from a combinatorial point of view; the discussion will focus on the abstract functionality of this process rather than refer to specific technologies. In the process, the QoS serves as selection criteria and the resulting problem is named the "QoS-based selection".

In Section 5.1, an introduction to the selection problem is presented. Based on the given description of this problem, a problem model will be defined in the subsequent Section 5.2. The problem model provides the basis to discuss several aspects of the selection problem: *a)* it enables the comparison with other known combinatorial problems (Section 5.3), *b)* it is used for the discussion about computational efforts to solve the problem, given in Section 5.3.5, and *c)* it is the basis for the explanation of different heuristics when applied to the problem. The application of the heuristics will be discussed in Section 5.4 of this chapter.

5.1 Introduction to the Selection Problem

The trading specification provides the foundation for discussion on QoS-based selection in this work. As one of the main presuppositions defined by this specification, it is presumed that the matchmaking process identifies functionally suitable candidates and that this step results in more than one candidate for a particular task. Otherwise, if the matchmaking has determined only one candidate for a task, the selection is trivial and the only one identified service is assigned. Furthermore, it is assumed that each task requires a different functionality and thus, the service candidates each represent a candidate for a particular task.

If more than one candidate is available to be assigned to perform a particular task, a selection must be made among them. As explained in Section 3.1.2, the retailer sets different – at least more than one – QoS requirements that serve as selection criteria applied to the set of candidates. Two basic kinds of QoS requirements are possible: Either the retailer sets one or more constraints that refer to a specific QoS characteristic, or the retailer identifies one or more QoS characteristics that are subject to optimisation. Chapter 3 has identified different QoS characteristics relevant for the application in service compositions, which are therefore relevant for the QoS-based selection. These characteristics are the throughput, the response time, the cost, the reliability, the availability, security aspects and the reputation of a service provider.

To explain the nature of the selection problem, the first chapter has presented a simple example (cf. Section 1.3) of a composition that arranges two tasks in parallel. In this example the optimisation of response time and cost represented the selection criteria. The given example has suggested that a problem with the selection arises in parallel structures with two or more selection criteria. However, a composition that arranges all involved tasks in a sequence can also result in a combinatorial problem when the QoS-based selection is applied: As an example, a sequence of tasks can be considered. For each task, more than one candidate exists. The optimisation goal is to achieve the lowest response time of the composition. In addition, a constraint on the cost is set, denoting that the assignment of services must not exceed a given cost limit.

The problem can be formulated as follows: a represents the number of tasks and b_i represents the number of candidates referring to a particular task denoted by index i. Then, an integer programming formulation is as follows:

$$\text{minimise} \quad t_{agg} = \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} t_{ij}$$

$$\text{while keeping} \quad c_{limit} \geq \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} c_{ij}$$

$$\text{and} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{1, \ldots, a\}, \quad p_{ij} \in \{0, 1\}$$

In this formula, t_{ij} represents the given response time of a candidate and c_{ij} the cost. The variable p_{ij} denotes whether a candidate is selected or not. These equations were discussed by Lee [78] and also by Yu and Lin [158]. In addition, these equations are a common representation of the Multiple Choice Knapsack Problem (MCKP), which is a popular optimisation problem. Section 5.3.1 will discuss the family of knapsack problems in further detail. Since the problem formulation represents a standard integer linear programming problem, Lee proposes solving instances of the problem with the help of available off-the-shelf tools for integer linear programming [78]. Yu and Lin propose performing the selection in accordance with the given description based on an algorithm by Pisinger for the MCKP which also builds on this integer programming formulation [158]. Without formal proof, the described variant of the selection problem can be transformed into an MCKP. Thus, the selection problem could be reduced to an MCKP and a solution algorithm for MCKP can be successfully applied to the selection problem. Figure 5.1 outlines this relation.

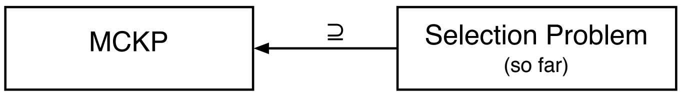


Figure 5.1: Relation between the MCKP and the selection problem.

However, when it comes to QoS characteristics like the reliability or the availability, the problem formulation does not result in a linear system of equations. The aggregation rules given in Section 4.4 define a multiplication of the values and the problem cannot be solved with linear programming techniques. Thus, Zeng et al., who also propose solving the selection problem with integer programming methods, transform the non-linear equation, resulting from constraints on availability or reputation for example, into a linear one [159], i.e. that the used variables have an exponent of 1. Applied to a sequential execution of services, an equation covering the availability is:

$$\text{minimise} \quad a_{limit} = \prod_{i=1}^{a} \prod_{j=1}^{b_i} a_{ij}^{p_{ij}}$$

Identical to the equations given before, a_{ij} denotes the availability of a given service and p_{ij} denotes whether a candidate is chosen or not. Then, the logarithm function is applied in the following manner:

$$\ln(a_{limit}) = \sum_{i=1}^{a} \ln\left(\prod_{j=1}^{b_i} a_{ij}^{p_{ij}}\right)$$

$$\ln(a_{limit}) = \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} \ln(a_{ij})$$

$$\text{and} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{1, \ldots, a\}, \quad p_{ij} \in \{0, 1\}$$

This is a linear equation again and thus can be considered with a integer programming approach. Thus, the transformation as proposed by Zeng et al. makes it possible to consider QoS characteristics that would normally result in a non-linear equation. Lee [78] and Zeng et al. [159] have also discussed the hardness of the problem in terms of its required effort to solve. The hardness will be discussed further in Section 5.3.5. As for the application to the selection problem, both Lee's and Zeng et al.'s approaches propose using the IBM Optimization Solutions and Library (OSL) which is a software tool for solving integer programming problems. As a conclusion, non-linear optimisation and constraint statements resulting from QoS characteristics that require a multiplication for their aggregation can also be transformed into linear statements and thus be reduced to the linear formulation of the MCKP. Figure 5.2 outlines this relation.

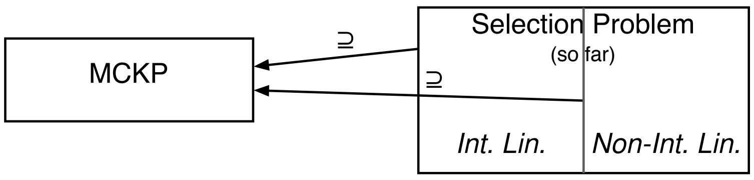


Figure 5.2: Relation between the MCKP and the integer linear and non-linear variant of the selection problem.

However, the approach by Lee, Zeng et al., and Yu and Lin is to presume that a composition can be represented by a sequential arrangement of relevant tasks. Nevertheless, a composition can also contain parallel structures. In Section 4.6 of the previous chapter, an example has explained that the approach to consider a single critical service leads to a problem when considering different QoS characteristics. To discuss the impact on the problem formulation, Figure 5.3 shows an example of a possible composition structure.

The composition structure shown in this figure contains two parallel arrangements. The first defines that one of the parallel task is executed while the second arrangement defines that all tasks are executed in parallel. The solution proposed by the mentioned works is to determine the critical path of the execution. Zeng et al. for example, propose considering the response time as the relevant QoS characteristic to identify the relevant services that form the critical path. Then, as for the QoS, its aggregation can be performed with one aggregation rule resulting in a linear or non-linear equation. However, the example makes clear that a more precise formulation is possible. Regarding the first parallel statement, only one service will be invoked at a time, but all are considered relevant. Consequently, a simple multiplication of the value would predict a worse availability that actually deliv-

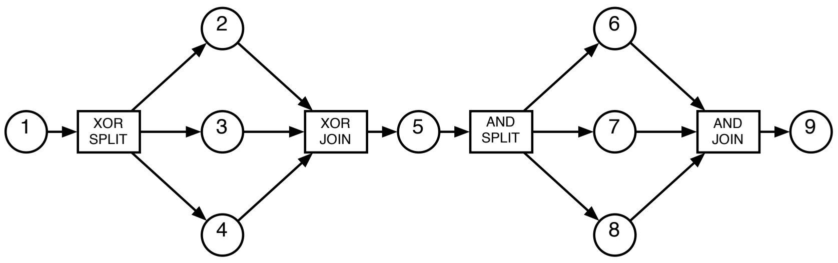


Figure 5.3: Composition structure that contains parallelism.

ered. For this structure, the assignment that results in the average is considered relevant.[1] Besides statements about response time, cost or other QoS categories, covering the aggregation of the availability of the first parallel structure results in:

$$\ln(a_{limit_1}) = \frac{1}{3} \sum_{i=2,3,4} \sum_{j=1}^{b_i} p_{ij} \ln(a_{ij})$$

$$\text{with} \quad 1 = \sum_{i=2,3,4} \sum_{j=1}^{b_i} p_{ij}$$

Then, the remaining availability values could be expressed with a single multiplication. To provide a more structured form, a detailed expression for the second parallel structure and the sequential arrangement would be as follows:

$$\ln(a_{limit_2}) = \sum_{i=6,7,8} \sum_{j=1}^{b_i} p_{ij} \ln(a_{ij})$$

$$\text{with} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{6,7,8\}, \quad p_{ij} \in \{0,1\}$$

$$\ln(a_{limit_3}) = \sum_{i=1,5,9} \sum_{j=1}^{b_i} p_{ij} \ln(a_{ij})$$

$$\text{with} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{1,5,9\}, \quad p_{ij} \in \{0,1\}$$

These equations do not define a global constraint, and therefore the following equation must be added in order to express a constraint on the availability that covers the entire composition:

[1]As explained in Section 4.3, for the model it is assumed that the services are invoked with equal probabilities. If necessary, the model could be extended with individual invocation probabilities which would result in a weighted average.

$$\text{maximise} \quad \ln(a_{limit}) = \ln(a_{limit_1}) + \ln(a_{limit_2}) + \ln(a_{limit_3})$$

This case demonstrates that a more precise problem formulation, with the goal to express a problem instance as a integer programming problem, depends on the given composition structure. If the selection problem is transferred into this formulation, then this problem formulation does not conform to the MCKP anymore. Figure 5.4 depicts this relation. In this figure, the selection problem is divided into four variants resulting from whether a sequential structure is considered and whether the problem involves non-linear statements which require a transformation.

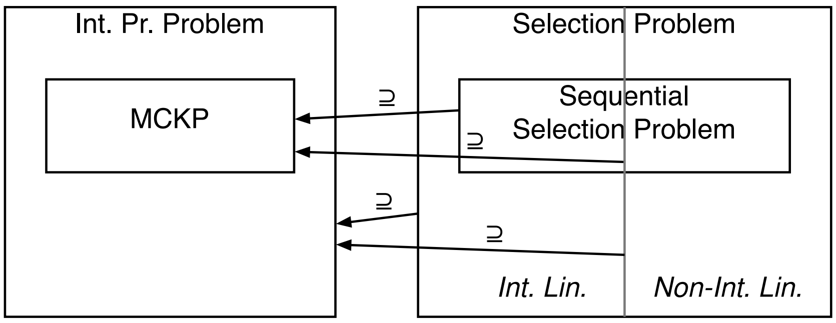


Figure 5.4: Relation between the MCKP and the selection problem involving the number of constraints.

The conclusion is that for each problem instance a new set of equations must be formed. Thus, the approach of this thesis is to express a problem model which captures the structural differences with their implications of the aggregation of different QoS categories. Moreover, if multiple criteria are considered for a combined goal function, the different criteria would require particular weights in order to provide significant solutions. The setting of appropriate weight factors becomes even more important, if non-linear QoS characteristics require a transformation involving a logarithmic operation. Then a weight must also compensate the non-linear operation applied to the QoS values. Because of this characteristic, different heuristics are discussed as an alternative to the approach of using integer linear programming. To summarise the discussion above and what has been presented in previous chapters, the problem model must capture the following characteristics of a service composition and possible instances of selection problems:

- **Multiple Optimisation Criteria.** The problem model must allow expressing multiple optimisation criteria. For example, a retailer can define two or more QoS characteristics at once that are subject for optimisation.

- **Multiple Constraint Criteria.** In a similar manner, multiple constraint criteria must be possible, i.e. a retailer can define more than one QoS characteristic as a constraint criterion.

- **Given Setup of Tasks and Candidates.** A preceding matchmaking process will result in a set of candidates for each task.

- **Composition Structure.** As the example of this section has shown, the aggregation of the composition depends on the arrangement of the given tasks and services. Thus, a problem model must also express these in order to correctly aggregate the given QoS values.

The different characteristics indicate that a selection problem can also involve more than one constraint and more than one optimisation goal. This aspect has been ignored in the previous discussion. Consequently, Figure 5.5 also identifies the case of one constraint and one optimisation goal as the special case that offers the transformability to the MCKP.

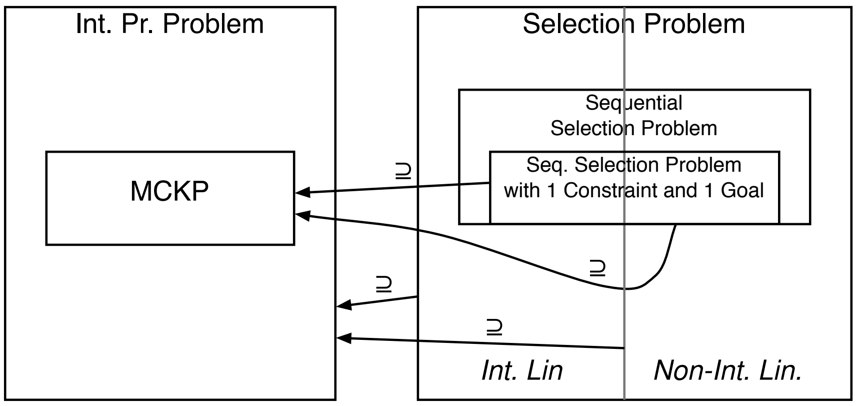

Int. Pr. Problem
MCKP
Selection Problem
Sequential Selection Problem
Seq. Selection Problem with 1 Constraint and 1 Goal
⊇
⊇
⊇
⊇
Int. Lin
Non-Int. Lin.

Figure 5.5: Summary of relations between the MCKP and the selection problem.

5.2 The Problem Model

Based on the requirements on the problem model from the previous section, this section will introduce a model that allows expressing given instances of a selection problem. Regarding the setup of the given composition case, this model contains the following elements:

- **Tasks.** The model must contain a set of tasks $\mathbb{T} = \{t_1, t_2, \ldots, t_a\}$ where the number a represents the total number of tasks in the composition.

- **Candidates.** The output of the matchmaking process is a set of (service) candidates $\mathbb{S} = \{s_1, s_2, \ldots, s_b\}$ where the variable b represents the number of all candidates. It is presumed that a task potentially requires a different type of functionality when compared to other tasks. Consequently, available services will provide a particular functionality and thus may not be suitable to perform different tasks. If the case occurs that a service can serve two different types of functionality, a candidate identified for one and another task is counted twice.

  An entire set containing all candidates does not express the described conditions. Therefore, the model is extended in the following way: The outcome of a discovery process results in a set of candidate-sets $\mathbb{U}$, each holding the set of candidates $\mathbb{S}_i$ for a particular task $t_i \in \mathbb{T}$.

$$\begin{aligned} \mathbb{U} &= \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_a\} \\ \mathbb{S}_i &= \{s_{i1}, s_{i2}, \ldots, s_{ib_i}\} \\ i &= 1, \ldots, a \end{aligned}$$

  in this definition, b_i denotes the number of candidates found for a particular task.

- **QoS characteristics.** For identifying different QoS characteristics, which is necessary when their values are used in optimisation statements, a number from 1 to q is used, with q denoting the total number of QoS characteristics.

- **QoS vector.** If different QoS characteristics are considered, different QoS values are assigned to the candidate services. Because more than one QoS value can be assigned for each candidate, a vector is used to represent a candidate. This *QoS-vector* holds the different QoS values denoted by the index $n,\ n = 1, \ldots, q$. The index q represents the different QoS characteristics that are taken into account. The result is a vector

$$\vec{s}_{ij} = \begin{pmatrix} s_{ij1} \\ s_{ij2} \\ \vdots \\ s_{ijq} \end{pmatrix}$$

  with index $i = 1, \ldots, a$ denoting the corresponding task and index $j = 1, \ldots, b_i$ denoting the number of the according candidate at task i. This resembles the indices as used in the previous section. In summary, the set $\mathbb{U}$ represents a set of sets of vectors.

5.2.1 The Selection Criteria

The selection of candidates involves the application of selection criteria. QoS characteristics can serve as criteria in an optimisation function. If one or more QoS characteristics are relevant for optimisation, the corresponding value of the QoS vector is subject to an optimisation function. This function depends on the direction of the dimension. For characteristics with a decreasing dimension, meaning that a lower value denotes a better QoS, such as response time or cost, the function must minimise the aggregated value. For categories with an increasing dimension, the function must maximise the value. Considering the aggregation rules that were presented in the previous chapter, a general formulation of optimisation goals is defined. The optimisation statement expresses either a minimisation of maximisation of the resulting QoS value, aggregated by function $f_n,\ n \in \{1, \ldots, q\}$ where n denotes the considered QoS characteristic:

$$\{minimise|maximise\}(f(\mathbb{U}_n)) \quad \mathbb{U}_n = \{s_{11n}p_{11}, \ldots, s_{ab_n}p_{ab}\}$$

$$\text{with } p_{ij} = \begin{cases} 1 \text{ if selected, and} \\ 0 \text{ otherwise.} \end{cases}$$

By using this notation, it is assumed that a resulting 0 represents a neutral element with respect to the aggregation function. For aggregation functions that multiply the given values, the neutral element would be the value 1. The previous section has given an example that has used the indices i and j in the same manner.

In addition to the optimisation criteria, one or more QoS categories can be relevant for expressing a constraint on the composition. Depending on the direction of the considered QoS characteristic with index n, the constraint denotes an upper or lower bound for the resulting aggregated value:

$$\{c_n > |c_n <\}(f(\mathbb{U}_n)) \quad \mathbb{U}_n = \{s_{11n}p_{11}, \ldots, s_{ab_n}p_{ab}\}$$

$$\text{with } p_{ij} = \begin{cases} 1 \text{ if selected, and} \\ 0 \text{ otherwise.} \end{cases}$$

In this definition, the neutral element has the same characteristics as given above with respect to the considered aggregation operation and to the use of the indices.

5.2.2 Modelling the Structure

In addition to the modelled sets of tasks and candidates, the discussion has suggested that the structure of the given composition is also relevant for the proper aggregation of different QoS characteristics. Thus, the problem model must involve a model of the composition structure based on the composition patterns as introduced in the previous chapter. The structural model is based on the model used for the aggregation as introduced in Section 4.3. It is defined as follows:

- A composition structure is represented by a set $\mathbb{K}_1$.
- The set $\mathbb{K}_1$ can contain tasks of the composition $t_1, \ldots, t_a$ or further sub-sets $\mathbb{K}_2, \ldots, \mathbb{K}_j$.
- The aggregation method has explained that the precedence relations between the elements in a sequence are not relevant for the aggregation of the QoS values. Since, order of the elements is not required for the aggregation, the enclosed elements of a sequence can be represented by a set $\mathbb{K}_x$. Because precedence relations do not exist between the elements of parallel structures, a set is an adequate representation structure for these elements as well.

  Consequently, each set $\mathbb{K}_x$ has a special type assigned, namely one of the composition patterns $CP_1, \ldots, CP_9$.
- In general, any set $\mathbb{K}_x$ can contain tasks or sub-sets.

For a formulation of a given selection problem, it is required that the composition is expressed by using the model as defined above based on the composition patterns. Moreover, since the selection problem focusses on the QoS in a composition, not required information, such as the precedence relation of tasks in a sequence, is not covered by this model. If a composition shows elements that not conform to the definition the above proposed model, the composition must be transformed first as discussed in the Section 4.5. The definitions of the composition structure together with the considered QoS characteristics also imply the application of the aggregation rules as introduced in the previous chapter.

5.2.3 Problem Model Summary

Incorporating all the previously discussed aspects of the selection problem, this problem can be defined by its following elements:

- $\mathbb{T}$ represents the set o tasks $t_1, \ldots, t_a$, with “a” tasks in total.
- $\mathbb{U}$ represents the set of candidate sets $\mathbb{S}_1, \ldots, \mathbb{S}_a$, each containing the candidates assigned to a particular task. Each candidate is represented by vector of elements that consists of the considered QoS values.
- $\mathbb{W}_O$ represents the set of optimisation goal functions, which can contain none, one or more optimisation statements.
- $\mathbb{W}_C$ represents the set of constraint statements, which can contain none, one or more optimisation statements.
- A data structure $\mathbb{K}_x$ that represents the composition structure containing the elements of the structural configuration as required for processing the QoS of the involved services.

- A valid solution is a list that contains one candidate for each task which can be expressed as a tuple L, $L = (\vec{s}_1, \ldots, \vec{s}_a)$. This tuple represents a selection of service candidates for which *a)* the aggregated QoS complies with the constraint statements of $\mathbb{W}_C$ and *b)* the aggregated QoS represents the optimal solution defined by goal functions in $\mathbb{W}_O$.

  A valid solution can be also expressed as the an assignment of the integer variables p_{ij}, when the problem is formulated as an integer linear optimisation problem.

This problem formulation does not presume the use of integer linear statements to express an integer programming problem. Of course, their expression for a given problem instance based on this model is possible. Such a formulation can be provided in the same manner as the aggregation rules are given in Section 4.4: for each pair of QoS characteristic and structural pattern, a statement is derived. However, as the previous section has shown, each given instance of a selection problem results in a different set of equations for expressing constraints and optimisation statements. Moreover, the application of transformation involving a logarithmic operation require the setting of appropriate weight factors.

This model does not capture any dependencies between the given QoS values. Such dependencies can occur if different QoS characteristics form a trade-off couple. A common trade-off couple is response time and cost where a shorter response time, i.e. a better quality, usually results in a higher cost. As for the QoS characteristics discussed in the previous chapter, the cost can form a trade-off couple with any of the remaining characteristics. A possible trade-off relation is not considered relevant for the discussion of the model. Instead, algorithms that attempt to solve a given selection problem can take advantage of such a dependency on an optional basis.

5.2.4 Aggregation of Multiple Optimisation Criteria

Different QoS characteristics can be considered at once for optimisation. Then, an aggregated goal function is required that allows to consider the different optimisation statements $\mathbb{W}_O$. When an algorithm tries to solve the problem, this results in the comparison of different QoS vectors that represent the aggregated QoS of the composition resulting from a particular assignment. For this comparison, the Simple Additive Weighting (SAW, [49]) method is proposed, which was introduced in the context of Multiple Criteria Decision Making (MCDM). The SAW approach normalises the individual value ranges from different QoS characteristics to value ranges of values between 0 and 1. For applying the SAW, a QoS vector with individual QoS values s_{ij_n} is considered, where n represents a QoS characteristic. Then, each value s_{ij_n} is replaced by the normalised value ν_{ij_n}:

$$\nu_{ij_n} = \begin{cases} \frac{max(s_{i1_n},\ldots,s_{ib_{in}})-s_{ij_n}}{max(s_{i1_n},\ldots,s_{ib_{in}})-min(s_{i1_n},\ldots,s_{ib_{in}})} & \text{for decreasing QoS char., and} \\ \frac{s_{ij_n}-min(s_{i1_n},\ldots,s_{ib_{in}})}{max(s_{i1_n},\ldots,s_{ib_{in}})-min(s_{i1_n},\ldots,s_{ib_{in}})} & \text{for increasing QoS char.,} \end{cases}$$

The internal $s_{i1_n}, \ldots, s_{ib_{in}}$ refers to all values from the considered QoS vectors referring to the relevant QoS characteristic n. As an example, the following QoS vectors are given:

$$\begin{pmatrix} 200 \\ 5 \\ 0.99 \end{pmatrix} \begin{pmatrix} 250 \\ 6 \\ 0.98 \end{pmatrix} \begin{pmatrix} 150 \\ 8 \\ 0.95 \end{pmatrix} \begin{pmatrix} 220 \\ 4 \\ 0.955 \end{pmatrix}$$

This could represent the three QoS characteristics response time, cost and availability. However, the meaning of the values is not relevant for the application of the SAW. Performing the SAW method replaces every value with a normalised one:

$$\begin{pmatrix} 0.5 \\ 0.75 \\ 1.0 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.5 \\ 0.75 \end{pmatrix} \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix} \begin{pmatrix} 0.3 \\ 1.0 \\ 0.125 \end{pmatrix}$$

Based on the normalised values, a score σ_j can be applied to each candidate [49], defined as:

$$\sigma_j = \frac{1}{q} \sum_{n=1}^{q} w_n \nu_{ij_n}$$

The variable q represents the amount of considered QoS categories. The weight w_n is applied to a particular QoS characteristic by the user's preference and could be omitted. The result of this procedure is a score for each candidate service or QoS vector. The result denotes the better QoS vector when two QoS vectors are compared. In the example, the resulting scores are in the same order: 0.75, 0.42, 0.33, and 0.52. Thus, presuming a neutral weighting, the candidate or aggregated QoS associated with the first QoS vector is considered as the best overall QoS.

5.3 Relations to Other Combinatorial Problems

If exactly one QoS characteristic is relevant for the optimisation, a selection algorithm must choose the candidate that provides the optimal value for each task. The effort for this operation is equivalent to a sort operation and therefore this specialisation of the selection problem can be regarded trivial. If more than one QoS characteristic is relevant for optimisation or for expressing a constraint, the selection can be regarded as a combinatorial problem. This problem has similarities with the knapsack problem, to a specific kind of a project scheduling problem (PSP), the QoS-based routing of packets in the Internet and the QoS-based scheduling of queries in the field of database systems. The upcoming subsection will explain their differences to the selection problem.

5.3.1 The Knapsack Problem

At the beginning, this chapter has already mentioned the relation of the selection problem with a member of the family of knapsack problems named multiple choice knapsack problem (MCKP). In the following, this relation will be explained in more detail. The core knapsack problem is about selecting a subset of available items for putting them into a knapsack. This problem is based on the simpler problem named “subset sum”. The subset sum problem is quickly explained: Considering a set of elements (numbers) $s_1, s_2, \dots, s_b \in \mathbb{N}$ and a number $c \in \mathbb{N}$, the goal is to find a selection of elements that represents a subset $\mathbb{I} \subseteq \{1, 2, \dots, b\}$ that fulfils:

$$\sum_{\forall i \in \mathbb{I}} s_i = c$$

The knapsack problem represents an optimisation variant of the subset sum problem. In this problem, each element has two dimensions, a weight s_{i_1} and a value s_{i_2}. Instead of the sum that must be identified, the knapsack problem has got a weight capacity with the goal to select a set of elements while respecting the given weight limit c. Considering the weight only, this represents a trivial problem: A solution can be found by sorting all elements by their weight. Then, starting with the lightest elements, they are added to the selection until the weight is reached. The problem becomes hard if the additional goal is to maximise the value v of the selected elements $s_{1_2}, s_{2_2}, \dots, s_{i_2}$. Then, the knapsack problem is about:

$$\begin{aligned} \text{maximise} \quad & v = \sum_{i \in \mathbb{S}} s_{i_2} \\ \text{while keeping} \quad & c \geq \sum_{i \in \mathbb{S}} s_{i_1} \end{aligned}$$

In other words, the problem is about maximising the value of taken items while the weight capability of the knapsack does not allow taking all items. Based on this problem statement, the literature distinguishes the fractional knapsack problem and the 0/1- or binary knapsack problem. The difference between both is that the 0/1-knapsack problem defines that splitting the elements is not allowed. Otherwise, an algorithm could sort all items by their value density s_{i_1}/s_{i_2} and begin to pack the knapsack starting with the highest value density. Then, if the algorithm meets the element that would exceed the given limit, only a fraction of this element is taken that still fits into the knapsack.

A further variant of the 0/1-knapsack problem is the Multiple Choice Knapsack Problem (MCKP). In this problem, the items for selection are sorted into $a \in \mathbb{N}$ classes and associated to each class is a set of items $\mathbb{S}_i$ which has got $b_i = |\mathbb{S}_i|$ number of elements. The goal of the MCKP is to select exactly one item of each set while keeping the constraints and maximise or minimise the optimisation goals. A model for problem would be as follows:

$$\text{maximise} \quad v_{agg} = \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} t_{ij}$$

$$\text{while keeping} \quad c_{limit} \geq \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} c_{ij}$$

$$\text{and} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{1, \ldots, a\}, \quad p_{ij} \in \{0, 1\}$$

These are the same equations as given at the beginning of this chapter, where it was also mentioned that Lee [78] and Yu et al. [158] have already discussed the MCKP in the context of the QoS-based selection of services to form compositions. The only difference is that mostly the MCKP considers a value that is about to be maximised while the selection problem as discussed by Lee and Tao et al. considers a time measure which is considered for minimisation. The difference between either minimising or maximising a value is considered as irrelevant when discussing the combinatorial issue.

To involve multiple constraints, the problem can be modelled as a Multiple Dimension Knapsack Problem (MKP), where – literally spoken – the knapsack has more than one limiting dimension. Such limiting dimensions could be the volume and weight of a knapsack and can be denoted by an index $1, \ldots, q$. This represents also a specific variant of the 0/1-knapsack problem. Then, the constraint to hold can be expressed as:

$$\sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} s_{ij_n} \leq c_n \quad \forall n \in 1, \ldots, q$$

$$p_{ij} \in \{0, 1\} \quad \text{identifies the selected elements/candidates.}$$

This combination between the MKP and the MCKP is known in the literature as Multiple Choice Multiple Dimension Knapsack Problem (MMKP). In Figure 5.6 this relation is outlined. Contrary to previous figures, the distinction between integer programming statements and non-linear statements that must be transformed differently is not depicted.

The following discussion will explain why the structure of the selection problem cannot be ignored and consequently the application of algorithms for the MCKP does not necessarily solve the selection problem: The example given in Figure 5.3 has shown that the parallel structures require a different model than the MCKP. From the interpretation of the problem formulation as given in the literature (e.g. Dudzinski and Walukiewicz [26], or Pisinger [108]), the MCKP covers only the scenario of an (unordered) sequence of service executions. A composition consisting of services all arranged into a parallel AND-split with AND-join

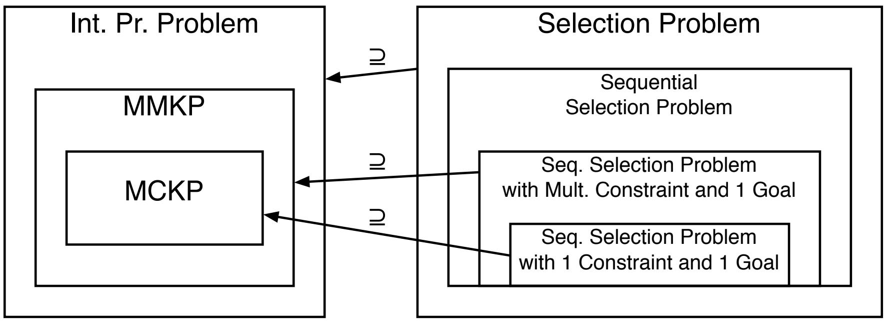

Int. Pr. Problem
MMKP
MCKP
Selection Problem
Sequential Selection Problem
Seq. Selection Problem with Mult. Constraint and 1 Goal
Seq. Selection Problem with 1 Constraint and 1 Goal
⊇
⊇
⊇

Figure 5.6: Relation between the MCKP, the MMKP and the selection problem.

arrangement is not covered. A solution denoting a selection of services found for a parallel arrangement would be different than for an unordered sequence. Moreover, as the example has explained, a given selection problem would result in a set of varying integer programming statements which puts the equivalence of the selection problem to the MCKP into question. This issue allows the assumption that the selection problem is at least as hard as the MCKP. However, considering the structural arrangements and different QoS characteristics, it requires a different problem model. Therefore, algorithms for finding or guessing solutions to a MCKP will not result in the same level of performance when applied to the selection problem.

To discuss the difference, a popular solution algorithm to the MCKP will be considered: A strategy for solving knapsack problems is known in literature as dynamic programming. The idea of dynamic programming is that a problem adheres to a basic principle of optimality which has been introduced by the mathematician Bellmann. This principle describes that the optimal solution to a problem can be represented by the combination of the optimal solutions to its sub-problems. If the determined partial solutions overlap and are stored, an algorithm can save efforts from the overlapping partial solutions.

The solution approach for applying the dynamic programming algorithm to the (basic) 0/1-knapsack problem is as follows: For a knapsack with given size constraint c, there is a selection $\mathbb{I} \subseteq \{s_1, \ldots, s_m\}$ that fulfils the constraint criteria. Then, for each element, $s_i \in \mathbb{I}$ holds that a partial problem with a smaller knapsack $c' = c - s_{i_1}$ has the solution $\mathbb{I} - s_i$. In this case, s_{i_1} represents the weight and s_{i_2} the value of the element. Based on this idea, an algorithm can determine the value v for a given set and sub-value as follows [114]:

$$f(s_i, c') = \begin{cases} 0 & \text{if} \quad i = 0 \\ f(s_{i-1}, c') & \text{if} \quad i > 0, s_{i_1} \geq c' \\ \max\Big\{f(s_{i-1}, c'), f(s_{i-1}, c' - s_{i_1}) + s_{i_2}\Big\} & else \end{cases}$$

By this definition, an algorithm can be derived. Such an algorithm operates on a numbered set of elements and starts with the elements with index i, showing the following behaviour:

- The first statement covers that no element is available for putting into the knapsack. Then, a value of zero is returned.
- The second statement covers that an element exists but the weight of the current element is larger than the remaining volume allows. Then, the function is invoked recursively with the previous element.
- In all other cases, the function chooses the maximum resulting value of either considering the actual element – and adding the value – or skipping the actual element. In both cases, the function is invoked with the previous element.

Based on this procedure, the algorithm creates a two dimensional array $[0..i, 0..c]$. For each pair of item and weight (i, c'), the algorithms computes the value according to this definition. The effort to create the array is just i by c' resulting in the worst-case effort of $O(n^2)$ added to the effort needed to perform the calculation of v for each table entry. The entry that shows the highest value v represents the solution. Such an approach results in a so-called pseudo-polynomial effort: Counting the computational steps results in a polynomial effort, while ignoring the effort to process the calculation of a table entry. As a consequence, for small values of i and c', such an algorithm usually shows acceptable performance.

In order to apply the dynamic programming approach to the MCKP, this algorithm must be modified to take at least one element of each class. The algorithm must class-wisely process the selection of candidates. The basic structure of the definition remains valid. However, it applies only to the selection of elements belonging to a particular class i. Dudziński and Wolukiewicz have proposed a dynamic programming algorithm for a MCKP [26]. At first, for a partial MCKP with a partial constraint c_{pt} can be formulated as follows:

$$\text{maximise} \quad v_i(c_{pt}) = \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} t_{ij}$$

$$\text{while keeping} \quad c_{pt} \geq \sum_{i=1}^{a} \sum_{j=1}^{b_i} p_{ij} c_{ij}$$

$$\text{and} \quad 1 = \sum_{j=1}^{b_i} p_{ij} \quad \forall i \in \{1, \ldots, a\}, \quad p_{ij} \in \{0, 1\}$$

$$\begin{aligned} v_0(c_{pt}) &= 0 \qquad \text{for} \quad c_{pt} \geq 0 \\ v_i(c_{pt}) &= -\infty \quad \text{for} \begin{cases} c_{pt} \leq 0 & \text{and} \quad i > 0, \quad \textit{or} \\ c_{pt} < 0 & \text{and} \quad i = 0 \end{cases} \end{aligned}$$

then

$$v_i(c_{pt}) = \max_{\forall j, j=1,\ldots,b_i} \Big\{ t_j + v_{i-1}(c_{pt} - c_j) \Big\}$$

By this definition, a partial MCKP is defined in the same way as an entire MCKP. For performing this algorithm, this definition also provides a statement that covers the iteration through the classes: If the algorithm reaches the class with the identifier "zero" but the constraint c_{pt} is greater or equal zero – the knapsack has some space left – then the resulting value is set to zero. If the resulting constraint c_{pt} is below zero, then the algorithm has encountered a class where all elements would violate the constraint. Thus, in this case, the resulting value is set to an infinite negative value denoting an invalid solution.

An algorithm based on this definition iterates through all elements of a class. It starts with the last class. If all elements of this class were tested, it proceeds with the previous class. The algorithm would calculate an array entry for each element. Since the creation of the array still results in a quadratic effort, this represents a pseudo-polynomial algorithm as well.

Applying this algorithm to a given selection problem can result in non-optimal solutions, because – besides the idea that it does not consider the composition structure – it presumes the same type of definition for aggregation of the QoS values (addition and subtraction) regardless of the required aggregation method, depending on the structure. Moreover, the structural requirements require the evaluation of several classes at once (for example, in a parallel arrangement), which cannot be performed with a recursive approach. A simple example clarifies this issue, considering a selection problem instance with the following characteristics:

- $\mathbb{T} = \{t_1, t_2, t_3\}$
- $\mathbb{U} = \left\{ \begin{array}{lcl} \mathbb{S}_1 & = & \Big\{ \vec{s}_{11} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \vec{s}_{12} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \Big\}, \\ \mathbb{S}_2 & = & \Big\{ \vec{s}_{21} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}, \vec{s}_{22} = \begin{pmatrix} 7 \\ 1 \end{pmatrix} \Big\}, \\ \mathbb{S}_2 & = & \Big\{ \vec{s}_{31} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \vec{s}_{32} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \Big\}, \end{array} \right\}$

  The upper value of each candidate can be seen as the response time and the lower the cost.

- $\mathbb{W}_O = \{(min(f(\mathbb{S}_{i_2})) \quad \forall \quad \mathbb{S}_i \in \mathbb{U})\}$ which means that the cost, the second value, is about to be minimised. it must be noted that in the domain of knapsack problems, such an optimisation criterion is named "negative value". Accordingly, the recursive definition requires a function that seeks the minimum instead of the maximum.

- $\mathbb{W}_C = \{(c = 10 \geq f(\mathbb{S}_{i_1})) \quad \forall \quad \mathbb{S}_i \in \mathbb{U})\}$ which represents a constraint statement about the response time, the first value, not being allowed to exceed the value of 10.
- G represents the graph consisting of pattern elements and tasks. The graph consists of one parallel pattern element which holds all the three tasks. The structure of this graph is depicted in Figure 5.7

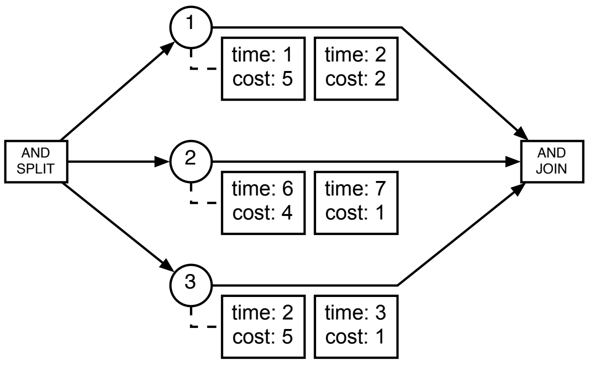


Figure 5.7: Graphical representation of the example selection problem (with candidates in rectangular boxes).

According to the definition of the algorithm for the MCKP, the algorithm would create a two-dimensional array which has c-number columns and has $\sum_{n=1}^{a} |\mathbb{S}_n|$ rows. Because the algorithm needs to keep the weight resulting from the selection, each entry consists of a tuple of $(\text{cost}, \text{time})$. The algorithm fills out the table starting from $i = 1$ and $j = 0$ and ends with $i = 6$ and $j = 10$ following the previously given definitions. If the algorithm encounters an already computed value v_{ij}, this is adopted from the table. Because the MCKP defines that one task needs to be taken from each group, a solution is found only in the rows that represent the last group involving a selection from the other group. Since the goal is to find the lowest cost while not exceeding the response time of 10 units, the tuple for $i = 6$ and $j = 10$ represents the optimal result. The combination resulting from this selection is (s_{11}, s_{21}, s_{31}).

With this result, it is clear that such an algorithm does not find the optimal solution, because it ignores the parallel structure in which the tasks are arranged. The optimal selection for this example would be the candidates (s_{12}, s_{22}, s_{32}) with a cost of 6 units and a response time of 7 units. In the example, the maximum value is relevant for the response time in a parallel arrangement. Of course, the definition of the recursive $\max\{\ldots\}$ could be modified to cover the parallel case. Consequently, sequential arrangements would not be possible. To solve this issue, the definition could be conditionally adapted depending on whether a parallel structure or a sequential structure needs to be processed. However, this would not cover

i\j	0	1	2	3	4	5	6	7	8	9	10
1	$-\infty$	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)	(5,1)
2	$-\infty$	(5,1)	(5,1)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)
3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	(9,7)	(6,8)	(3,9)	(3,9)
4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	(9,7)	(6,8)	(6,8)	(6,8)
5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	(14,9)	(13,10)
6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	**(10,10)**

Table 5.1: Array values resulting from dynamic programming approach.

the case that parallel and sequential structures would occur in the same problem instance.

The conclusion is that an algorithm known to find an optimal solution to any MCKP, if a solution exists, does not guarantee finding an optimal solution to a given selection problem. Thus, the MCKP is different from the selection problem. It can be assumed that the principle of optimality does not apply to the selection problem as it is modelled in this work. The structural characteristic of the selection problem requires considering a selection of tasks at once for a decision. Thus, splitting the problem into sub-problems by classes or tasks does not cover the structural characteristic of the given composition case.

5.3.2 The Project Scheduling Problem

If multiple QoS characteristics are subject to the optimisation or if they are considered to form a constraint, the selection problem is similar to a Resource Constrained Project Scheduling Problem (RCPSP). A project scheduling problem occurs when resources (usually humans) must be distributed to jobs of a project. The most common optimisation goal of the basic RCPSP is to reduce the duration of the project while spending as few resources as possible. Two types of RCPSPs are distinguished: One is named Single Mode RCPSP and the other is known as Multi-Mode RCPSP (MRCPSP). The single mode RCPSP only deals with fixed values for the duration and the cost of a task. In an MM-RCPSP, a job can be done by using different modes which vary in cost and duration. Therefore, an MM-RCPSP is primarily considered as relevant in comparison to the selection problem. In addition to its mode, an RCPSP can be classified by a couple of other characteristics. Based on an overview about RCPSPs by Yang et al. a selection problem seen as RCPSP has the following characteristics [153]:

- **Objective.** Objectives are distinguished by being regular or irregular. Regular objectives do not interfere with the goal to minimise the duration of the project, while irregular objectives allow following another objective – for example to equalise the consumed resources among involved parties [153].

Applied to the selection problem, the RCPSP has a non-regular objective because depending on the considered QoS categories and the applied weight; a worse duration can be considered better if, for example, the cost is reduced accordingly. The objective can be defined as an optimisation function, as given in Section 5.2.1.

- **Precedence Relation.** Two types of precedence relations are discussed in the literature. Either a task can be started with a specified time window after a preceding task has finished or a succeeding task can start any time after the preceding task has finished. Regarding a composition of services, the common case is that a task is started immediately after the preceding task finished.

  The constraint for the precedence in service compositions can be defined as follows: α_{ib_i+1} is the start time of a candidates $s_{ib_i+1} \in \mathbb{S}_i$ of task $i, i = 1, \ldots, n$, and ω_{ib_i} the finish time of a candidate $s_{ib_i} \in \mathbb{S}_{i-1}$ for the preceding task $i-1$. Then, the precedence constraint is:

  $$\alpha_{ib_i+1} q_{ib_i+1} \geq \omega_{ib_i} q_{ib_i} \quad \text{with } q_{ib_i+1}, q_{ib_i} \begin{cases} 1 & \text{if selected} \\ 0 & \text{otherwise} \end{cases}$$

  Between two tasks, a time delay might occur, because an execution environment cannot execute a task at exactly the same moment when another task has finished. However, this technical issue is ignored in this discussion.

- **Preemption.** If preemption is allowed, the execution of a task can be suspended in order to execute another task in the meantime. This is useful if some resources are only available within a specific period and other tasks can be suspended then. Since the invocation of services in the context of a composition usually is an atomic operation, preemption is not considered possible.

- **Resource requirements per period.** In the domain of project scheduling, using resources at different times can result in different costs. For example, performing a task at night results in higher payments for night shifts. For computer systems, a similar idea can be applied: The execution of a service gets more expensive at peak hours. However, since currently no such example exists in the domain of electronic services, this aspect is ignored in the further discussion.

- **Trade-offs.** In the context of RCPSPs, the trade-off is characterised by two criteria, where an optimisation of the one means a change to the worse for the other. An algorithm must find a counterbalanced solution. A usual trade-off pair is formed by time and cost, whereas the cost and the time should be kept both as low as possible. Possible trade-off couples can be formed for the selection problem of cost vs. one of the other mentioned categories, in the sense that a higher quality results in a higher cost. However, a trade-off couple could be also time vs. availability, for example for a provider, where services usually execute very quickly but may fail.

These points show that the selection problem can be transformed into an RCPSP that resembles the precedence relations by the execution graph and does not allow preemption. The relation between the two problems is outlined in Figure 5.8: Without giving a formal proof, the given description explains that the selection problem can be reduced to an RCPSP. A large number of heuristics are already available for RCPSPs [153]. Not every approach can be applied to the selection problem: RCPSPs and service compositions cover the execution order of tasks differently. In compositions, the order is in most cases pre-defined by a flow description, while the tasks of a project are subject to precedence relations, which may allow to push a particular task for- or backwards in order to optimise the utilisation of resources.

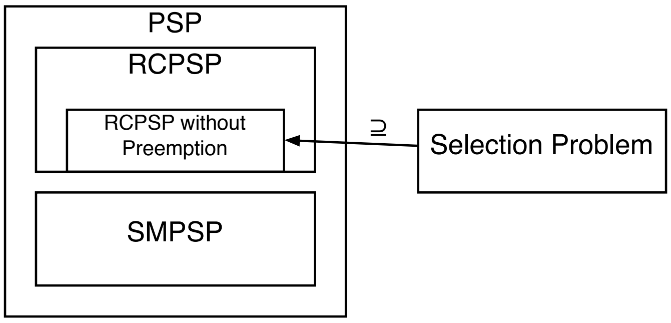


Figure 5.8: Relation between the RCPSP and the selection problem.

5.3.3 Query Planning based on Quality-of-Service

The work of Naumann and Leser discusses the optimisation of queries to a federation of data sources by using the QoS as criteria [95]. The authors have introduced an approach to optimise the QoS of a set of queries by two steps: First, data sources are sorted out which would result in a bad QoS in any case. Then, different combinations of querying data sources, which provide the same required information, are compared in terms of their resulting QoS.

The basic setup is as follows: A mediator serves as a front-end to users. The mediator represents a similar role as the retailer for the selection of services; a mediator wraps different data sources for the end-user. These data sources can either provide similar or different data and are all described with QoS characteristics. Accordingly, a query by the user (user query, UQ) submitted to the mediator is translated into queries to the wrapped data sources (wrapper queries, WQ). The general approach is to define such translations between UQs and WQs with Query Correspondence Assertions (QCA). Because the available data sources can provide overlapping or similar data sets, a user query can result in different possible QCAs and combinations of them. Then, each set of QCAs, which completely answers the UQ, can be ranked based on the aggregated QoS provided by the involved data sources.

The result of their approach is the identification of query plans that provide the optimal QoS. Naumann and Leser present a set of QoS characteristics which are tailored to the application of database systems. This set includes characteristics which have been also discussed in Chapter 3 (e.g. response time, availability or cost). Other characteristics hardly fit into the domain of services, which are "ease-of-understanding" or "completeness". The completeness denotes the level of how well a data source meets the information required for a possible UQ. Nevertheless, for a discussion of the combinatorial problem, the interpretation of the numerical values does not have any impact on the hardness of the problem.

The combinatorial issue with this approach is the evaluation of possible query plans, meaning that valid sets of QCAs must be determined. Naumann and Leser explain that the theoretical effort to determine all resulting possible WQs is NP-complete [95]. The number of possible queries to evaluate is reduced by grouping WQ to the building blocks which refers to the concept of QCAs. Still, the number of possible QCA-sets answering a UQ rises exponentially with a rising numbers of QCAs.

To cope with this problem, the authors propose a three-staged approach. In the first stage, an algorithm sorts out data sources that offer a generally bad quality. In the second stage, Leser has discussed an algorithm that efficiently identifies only those QCA-sets that would also answer the UQ [79]. In the third stage, a comparison algorithm determines the QoS of all remaining queries and applies a ranking that allows the identification of the optimal QCA-set.

Comparing this approach with the selection problem reveals that the basic principle of the both is the same: First, a query or execution plan is determined and then the QoS for this plan is aggregated. At third, the resulting QoS is put into a ranking to identify which execution plan results in the optimal QoS. For both, the combinatorial problem lies in the generation of valid query or execution plans. However, the query optimisation is different in two points and these make clear that the selection problem requires a different approach:

- **Overlapping Data Sources.** The algorithm that determines valid sets of QCAs considers the case, for example, that a data source can deliver the same information as two other data sources. This results in two different QCAs and thus the algorithm evaluates one data source against two composed others. The selection problem as discussed here does not consider this case, because it is presumed that the decision about this would have been taken in the matchmaking process of the composition (cf. Section 7.2).

- **Parallel Queries.** The mediator is allowed to execute different queries to the data sources resulting from a QCA set entirely in parallel. As for the MCKP which essentially handles all services executed sequentially, this would mean that all services are executed in parallel. Considering that this problem could be expressed as a integer programming problem, if the relevant QoS characteristics allow, then, the problem expression would cover only one structural case, namely the parallel arrangement. In other words, the problem model given for the QoS-based query planning by Leser and Naumann does not capture any specific structural arrangements as they occur in service compositions.

If an instance of a selection problem arranges all tasks in an AND-AND parallel pattern, then this instance can be transformed into the problem of finding the optimal query set. In accordance with this special case, Figure 5.9 shows the reduction relation between the two problems. However, it must be noted that the QoS-based query planning involves also the identification of feasible query sets in advance. Thus, the selection can be reduced only to a part of the problem which is described by Naumann and Leser.

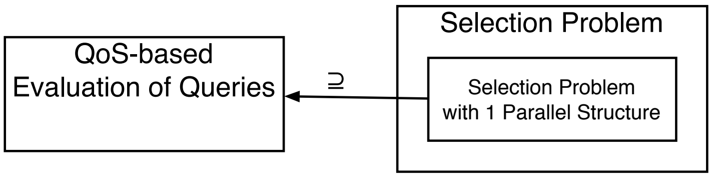


Figure 5.9: Relation between the QoS-based evaluation of queries and the selection problem.

5.3.4 Routing in the Internet based on Quality-of-Service

Routing in the Internet today has the goal to identify feasible paths for sending packets among different networks from one location to another. The routing is performed by routers that implement a routing algorithm. The routing in the Internet follows in most cases a best-effort strategy, meaning that every router tries to deliver the best service possible according to a single metric. As metrics, routing algorithms can consider the cost for transferring the data or the number of routers to pass, which is considered as the number of hops. The resulting algorithmic issue is to identify the shortest path in a graph where nodes represent routers and the edges represent their connection, weighted according to the considered metric. This routing scheme has proven its reliability in the past. However, it does not address the demands of QoS-dependent services, such as telephone applications or video broadcasts over the Internet. Such applications require a routing mechanism that is capable of ensuring a particular QoS level. For example, broadcasting video data requires a constant amount of bandwidth on all connections between involved routers in order to transmit a continuous amount of data. Otherwise, the user will experience interruptions when watching the video and will likely reject the service.

The Internet Engineering Task Force (IETF) has published a document that discusses a general framework for the QoS-based routing (cf. RFC number 2386 [20]). This documents defines the following primary goals: $a)$ enhancing the standard routing process by providing mechanisms to ensure the given requirements with respect to different QoS characteristics, $b)$ utilising the given resources more efficiently, and $c)$ recognising QoS violations and establishing compensation activities. The QoS-based routing plays a complementary role to (signalling) protocols that offer resource reservation in order to express demands for a particular QoS level. While resource reservation covers static agreements between sender and receiver, QoS-based routing dynamically determines feasible paths in order to provide the requested QoS.

Therefore, the combinatorial issue of QoS-based routing is to identify a feasible path in a graph with involving multiple QoS constraints. In this field, the literature distinguishes between additive QoS, where the individual values are added, and multiplicative QoS, where the aggregation of values is performed by a multiplication [139]. A QoS characteristic that is neither additive nor multiplicative but requires identifying the lowest value is named concave, e.g. as for the bandwidth. Wang and Crowcroft present a proof that finding feasible paths is NP-complete when either two additive or two multiplicative constraints are considered [137]. In addition, they proof that a combination of one additive and multiplicative constraint is NP-complete as well. If the application of the logarithmic function to multiplicative constraint statements is considered as discussed at the beginning of this chapter, such constraints become also additive, which was pointed out by Yang [139].

In addition, Wang and Crowcroft discuss the routing problem with one additive constraint in conjunction with the concave bandwidth [137]. For this constellation, the authors propose an algorithm with quadratic worst-case effort that finds the optimal solution. As Yang also has pointed out, an algorithm can cover non-additive characteristics as the bandwidth by first sorting out unfeasible paths in a pre-processing step. Then, the algorithm must optimise for only one QoS constraint for which well-known finding-shortest-path algorithms can be applied. Thus, the problem issue with QoS-based routing in IP network is similar to the QoS-based selection of services: A combinatorial problem arises if multiple QoS characteristics must be considered at once. However, the problem of multi-criteria QoS-based routing shows different aspects when considering the following points:

- **Sequences Only.** One obvious difference lies in the considered structure for IP networks. Contrary to a composition, at the beginning the relation between the nodes does not require a direction and thus no precedence relation among the involved nodes is given. The algorithm has the goal to identify precedence statements between the nodes that resemble a routing path. The precedence relations of the selection problem are defined by a graph. Consequently, the graph is a part of the problem formulation and not part of the solution as with the QoS-based routing.

Moreover, if such a routing path is found, each involved node has exactly one incoming path and one outgoing path, meaning that a solution is not allowed to show any parallel forking and joining elements. In other words, the considered structure for the routing does not cover the structure of service compositions.

- **Different Number of Hops.** Like the MCKP, the QoS-based selection requires that the number of selected services stays constant because of the predefined number of tasks. The number of selected routers can vary depending on the given QoS. An algorithm that attempts to solve a given MCKP or QoS-based selection can presume from the beginning that, from certain sets of services, at least one must be selected; an algorithm that identifies optimal paths among the router candidates cannot make such presumptions.

As stated for the QoS-based query planning, the problem of the QoS-based routing does not match the selection problem regarding the problem model and solution strategies.

5.3.5 Computational Complexity

Considering the model introduced in Section 5.2.3, the possible combinations result from the Cartesian product of the candidate sets in $\mathbb{U}$:

$$\mathbb{S}_1 \times \ldots \times \mathbb{S}_a$$

This product results in a set of a-tuples where the number of elements represents the number of possible combinations:

$$|\mathbb{S}_1 \times \ldots \times \mathbb{S}_a| = |\mathbb{S}_1| \cdot \ldots \cdot |\mathbb{S}_a| = \prod_{i=1}^{a} |\mathbb{S}_i|$$

This indicates the resulting exponential effort for a rising number of tasks if an algorithm potentially evaluates all possible combinations. Thus, the computational complexity of the naive algorithm can be formulated using the big-O notation as $O(m^n)$ where n represents the number of tasks and m denotes the maximal number of candidates for a task.

The exponentially rising number of combinations poses the question whether an algorithm is required to evaluate all combinations to find the optimal solution. The resulting consequence would be that the selection problem requires an algorithm that always shows an exponentially rising effort. For business processes of 25 tasks with five candidates each, the resulting number of combinations is $5^{25} \sim 9 \cdot 10^{13}$. Processes of 25 tasks seem to be exceptionally large. However, if the structure must be transformed before, using the node duplication technique as discussed in Section 4.5 of the previous chapter, such sizes are likely to occur.

NP-Hardness of the Selection Problem

For this discussion, the concept of a polynomial reduction between encoding alphabets associated with a problem is used (cf. Garey and Johnson [37, section 2.5]). Let two languages L_A and L_B represent the encoding schemes of the two problems A and B. If language L_A can be transformed into L_B then each instance of a problem A can also be expressed as a problem instance of B. The conclusion from a possible reduction is then that problems of L_B are at least as hard as problems of L_A. The concept of the polynomial reduction requires that a transformation between the two languages can be achieved by an additional machine which also performs within a polynomial bounded amount of time. Problems with a polynomial upper bound show the characteristic that combining them in sequence will result in a new problem with a polynomial upper bound as well. Thus, if a language L_A associated with a polynomial problem A can be transformed with polynomial effort into L_B, B is a problem requiring at least polynomial worst case effort.

This relation applies also to problems that can be solved by a non-deterministic touring machine in a polynomial bounded amount of time, so-called NP-problems. Because of this argumentation it can be determined if a problem is NP-hard, meaning at least as hard as another NP-problem. A language L_B is named NP-hard, if

$$L_A \leq_p L_B \quad \text{for} \quad L_A \in NP$$

where the relation $\leq_p$ denotes that L_A transforms polynomial to L_B and NP denotes the class of NP-hard problems.

The polynomial transformability between the knapsack problem and an MCKP has been discussed by Pisinger [108, pages 107-108]. Any knapsack problem represents a special case of an MCKP: Every knapsack problem instance can be transformed into an MCKP instance by setting the number of elements within one class to $b_i = 1$. This can be easily understood by considering the formulation given for the MCKP in the previous Section 5.3.1. If $b_i = 1$ then the inner sum becomes obsolete and the selection variable will be $p_{ij} = 1$ in all cases. As a result, the MCKP-compliant formulation of the instance is equal to the formulation of the knapsack problem. And regarding the knapsack problem, a proof exists that it is a NP-hard problem (cf. Garey and Johnson [37, section A6]). A transformation from an MCKP problem formulation to the formulation of the selection problem can be given in the following way 5.2:

- The set of classes is modelled as $\mathbb{T} = \{t_1, \ldots, t_a\}$ where a denotes the number of items.
- Each $\mathbb{S}_i$ representing the elements of a class in the MCKP sense is put into the set $\mathbb{U}$.
- The value of each item v_{ij} is set to one element of the referring candidate

vector $\vec{s}_{ij}$, and the weight c_{ij} is analogously set to the other element:

$$\vec{s}_{ij} = \begin{pmatrix} v_{ij} \\ c_{ij} \end{pmatrix}$$

- The set $\mathbb{W}_O$ of optimisation functions contains for the MCKP case just one statement ($|\mathbb{W}_O| = 1$):

$$\mathbb{W}_O = \left\{ f(\mathbb{U}_1) = \sum_{i=1}^{a} \sum_{\vec{s}_{ij} \in \mathbb{S}_i} p_{ij} \vec{s}_{ij_1} \right\}$$

- The set $\mathbb{W}_C$ of constraint functions also contains for the MCKP case just one statement ($|\mathbb{W}_C| = 1$):

$$\mathbb{W}_C = \left\{ c_2 \geq \sum_{i=1}^{a} \sum_{\vec{s}_{ij} \in \mathbb{S}_i} p_{ij} \vec{s}_{ij_2} \right\}$$

- The structure $\mathbb{K}_x$ that arranges all task nodes arbitrarily in a sequence.

- The solution is a selection which conforms to the optimisation function given by $\mathbb{W}_O$ and the constraint given by $\mathbb{W}_C$, which is the same integer programming formulation as for the MCKP (cf. Section 5.3.1).

It is obvious that this transformation can be performed with polynomial bounded effort. Then, any instance of an MCKP can be transformed into an instance that conforms to the presented model of the selection problem. Contrary to the Figure 5.1 presented at the beginning of this chapter, the selection problem outlined in this figure denotes all variants that were discussed so far. The conclusion of the reducibility of the MCKP to the selection problem is that the selection problem is NP-hard. The resulting direction of the reduction is depicted in Figure 5.10. Due to this discussion, it becomes clear that the given model of the selection problem generalises the MCKP with regard to two aspects: First, it covers not only sequences but also parallel, conditional or combined structures. In addition, it covers multiple optimisation or constraint criteria.

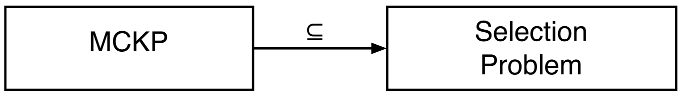


Figure 5.10: Reduction of the MCKP to the selection problem.

5.4 Heuristic Algorithms

In the selection process, an available candidate is assigned to each task. The simplest approach represents an algorithm with a greedy behaviour which evaluates the assignment from a local perspective: Such an algorithm determines each assignment for each task individually and thus ignores possible assignments of other tasks in combination. However, the previous discussion has pointed out that such an approach would not result in the optimal solution. The question is how much worse such an approach would be when compared to an algorithm that always finds the optimal solution.

Building onto the analogies between the combinatorial problems and the selection problem as explained in the previous section, the approach of this work is to discuss and evaluate the application of heuristics. In the following subsections, different approaches are explained and compared. A representation in pseudo-code notation of the discussed algorithms is given. The presentation of the algorithms in pseudo-code provides a more transparent discussion about the resulting computational complexity. The used pseudo-code notation requires some remarks:

- The inputs and outputs of the algorithms are given only in a schematic way: For each algorithm, the relevant data from a selection problem formulation according to the model is given. An implementation would require particular input and output parameters of potential operations, which are omitted if they do not improve understanding of the algorithm.
- The variable a represents the number of tasks; the counters i, j have the same meaning as used in the preceding discussion.
- The initialisation of variables as required in most programming languages is omitted for the same reason as with the input and output parameters.
- Variables are not typed. Thus, integer or floating point numbers and other data types share a common notation.

5.4.1 Greedy-based Selection

For a greedy selection, a value v_{ij} computed by function $f_{saw}(\vec{s}_{ij})$ represents the selection criteria. This function can implement the SAW method to assign a score for each candidate $s_{ij} \in \mathbb{S}$. The QoS characteristics relevant for the SAW calculation are denoted by the optimisation statements denoted by $f_n \in \mathbb{W}_O$. The algorithm starts with calculating the value for each candidate. Then it assigns the candidate with the highest value for each task. Thus, the graph G is not considered. The algorithm is described in listing 1.

The effort for this algorithm is $O(n) + O(n \log n) + O(n) = O(n \log n)$. The algorithm shows two for-loops which would indicate a quadratic effort, but in fact, the algorithm just iterates through all candidates. Assuming that b denotes the

```
    Data: $\mathbb{T}, \mathbb{U}, \mathbb{W}_O, \mathbb{W}_C$
    Result: array of candidates $l[\,], |l[\,]| = a$
    begin
1       foreach $\mathbb{S}_i \in \mathbb{U}$ do
2           foreach $\vec{s}_{ij} \in \mathbb{S}_i$ do
3               $v[i][j] \leftarrow f_{saw}(\vec{s}_{ij}, \mathbb{S}_i)$
            end
        end
4       $l[i] \leftarrow \vec{s}_{ij},\ \vec{s}_{ij} \mid f_v(\bar{s}_{ij}) == max(v[i][1], \ldots, v[i][j])$
    end
```

Algorithm 1: Greedy heuristic applied to the selection problem.

number of candidates, the effort is linear depending on the input size. The effort for the sort operation is added and the effort for assigning the candidates is considered linear. The sort results in the significant effort of $O(n \log n)$. For this greedy implementation, the use of a sort algorithm with the worst-case effort of $O(n \log n)$ is assumed. Such algorithms usually require extra memory to achieve this logarithmic class of effort, otherwise the worst case effort of common sorting algorithms is $O(n^2)$. The best case effort is $\Omega(n \log n)$ as well, because the algorithm performs the sort in any case.

The clear downside of this approach is that it is not possible to consider a global constraint while optimising the overall QoS. As an alternative, a greedy-based algorithm can also support one constraint criteria without optimising for the QoS. The algorithm starts sorting the candidates not by the score but by the relevant constraint criteria denoted by a function in $c_n \in \mathbb{W}$. Then, the algorithm assigns, for each task, the candidate that offers the best value of the constraint criteria. If a solution exists that complies with the constraint, it is found with this step. Clearly, this approach does not optimise the overall QoS of the compositions. In the remainder of this work, the two resulting QoS optimising algorithms will be named “Greedy” and the single-constraint aware algorithm “Constraint”.

5.4.2 Discarding Subsets

This algorithm represents a backtracking-based approach. It uses a search tree which consists of nodes, each representing a possible pair of a candidate and a task. Each level of the tree holds pairs of a particular task only, resulting in a tree having the same number of levels as tasks. Each path, starting from the root and ending at a leaf, represents a possible assignment of candidates. The advantage of this algorithm, in comparison to a straight global selection, lies in the idea to cut sub-trees representing unfavourable combinations to save computational efforts.

Such an approach normally identifies the optimal solution and therefore cannot be regarded as a heuristic. When applied to the selection problem, as described this work, this approach results in a heuristic, because the cutting rule must be based on an estimation. The reason is explained as follows: Considering the time, the cutting rule is clear. If a complete combination has already been determined that shows a lower (better) time as the partial combination processed at some moment, the algorithm cuts the corresponding sub-tree. Each additional candidate would worsen the time in any case.

Contrary to that, for categories where the aggregation calculates the arithmetic mean, a rule cannot determine whether the QoS gets worse or better. The reputation and the availability represent examples for such a QoS characteristic. Applied to the selection problem, the discarding subsets algorithm considers two cutting rules:

1. A partially evaluated combination already violates a constraint. Thus, the algorithm cuts the sub-tree.

2. The algorithm compares the aggregated QoS of the partial combination with the aggregated QoS of already processed complete combinations. However, for this comparison the QoS categories involving a mean-based aggregation are ignored. This strategy does not find the optimal result necessarily, because the cutting rule represents an estimation.

The algorithm operates on a tree structure, which results from the structure $\mathbb{K}_x$ that can contain tasks and further substructures. It performs the following steps as given in the listing 2. The listing includes functions which represent the following functionality: f_{con} tests a proposed solution to determine whether the required constraints hold or not, f_{agg} aggregates the QoS resulting from a proposed solution, and f_{saw} performs a SAW-based comparison (cf. Section 5.2.4) with a number as an output: If the number is greater than zero, the right argument represents a better QoS, otherwise vice versa. It must be noted that the comparison could be based either on one aggregated QoS characteristic or on many; either way this does not require a modification to the algorithm. The same applies to the check if a constraint is violated or not: This check can involve either one or more constraints.

The worst-case effort of this approach is the same as for the brute-force strategy to evaluate all possible combinations, thus $O(m^n)$. The best-case effort occurs if the first combination found represents the optimal combination and all other sub-trees are cut. Then, the algorithm walks through all levels of the tree and evaluates at least all tasks of the current level of the tree. The resulting effort for this would be n resulting in the effort class of $\Omega(n^2)$: On each task level, the algorithm would evaluate combinations resulting from applying other candidates. The number of task-candidate-pairs could be represented by a 2-dimensional matrix.

```
**Data**: $\mathbb{T}, \mathbb{U}, \mathbb{W}_O, \mathbb{W}_C, \mathbb{K}_x$
**Result**: array of candidates $l[\,], |l[\,]| = a$
initialisation: $pos \leftarrow 0$
discarding( $\mathbb{K}_x, test[\,], pos$)
**begin**
    **if** | $test[\,]$ | == $a$ **then**
        **if** | $l[\,]$ | == 0 **then**
            **if** $f_{con}( test[\,] )$ **then**
                $l[\,] \leftarrow test[\,]$
            **end**
        **else**
            $\overline{y} \leftarrow f_{agg}( l[\,] )$
            $\overline{x} \leftarrow f_{agg}( test[\,] )$
            $z \leftarrow f_{saw}( \overline{y}, \overline{x} )$
            **if** $z > 0$ **then**
                $l[\,] \leftarrow test[\,]$
            **end**
            return
        **end**
    **end**
    **foreach** $element \in \mathbb{K}_x$ **do**
        **if** $element == t_i,\ t_i \in \mathbb{T}$ **then**
            **foreach** $\overline{s}_{ij} \in \mathbb{S}_i$ **do**
                $test[\,] \leftarrow test[\,] + \overline{s}_{ij}$
                $\overline{y} \leftarrow f_{agg}( l[\,] )$
                $\overline{x} \leftarrow f_{agg}( test[\,] )$
                $z \leftarrow f_{saw}( \overline{y}, \overline{x} )$
                **if** $z > 0$ **then**
                    **if** $f_{con}( test[\,] )$ **then**
                        discarding( $\mathbb{K}_x, test[\,], pos + 1$ )
                    **else**
                        return
                    **end**
                **end**
            **end**
        **else**
            discarding( $\mathbb{K}_x, test[\,], pos + 1$)
        **end**
    **end**
**end**
```

Algorithm 2: Discarding heuristic applied to the selection problem.

5.4.3 Bottom-Up Approximation

Looking at the similarity of the selection problem to RCPSPs, it turned out that several approaches for solving RCPSPs work on a precedence model which does not allow the application to the selection problem. The resulting problem is that a solution space is created which allows the rearrangement of tasks. The resulting bounding rules cannot be applied efficiently to the selection problem.

In previous research work, the application of an heuristic algorithm for solving a RCPSPs, which was introduced by Yang et al. [153], was discussed (cf. Jaeger et al. [61]. The approach presumes that constraint and optimisation criteria form a trade-off couple, i.e. the quicker a task is performed the more it will cost. Moreover, it supports only one constraint criteria. It will first sort the candidates by the constraint criteria. Then it will improve the QoS of individual tasks by replacing the original candidate with the candidate that offers the next worse constraint value. Considering the trade-off relation, the idea is that the QoS is improved by worsening the constraint value until it reaches the limit.

The algorithm is given in the listing 3. At first, it sorts the candidates by the (one) constraint QoS characteristic. In other words, it performs a constraint-based selection. Then, it assigns, for each task, the candidate with the next worse constraint. A proposed solution is kept, if the constraint still is not violated (cf. line 10). Figure 5.11 outlines how this algorithm proceeds, based on the tasks and candidates of the example given in Section 5.3.1.

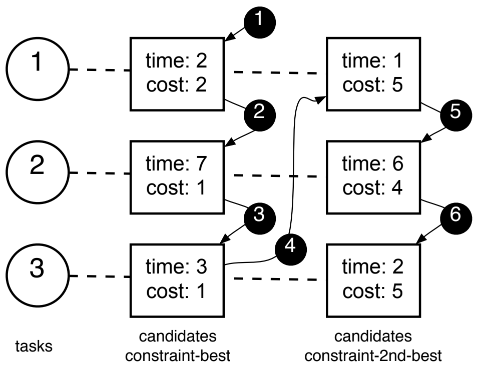


Figure 5.11: Example processing order of bottomup heuristic.

The algorithm tries to improve the QoS until it has gone through all tasks and did not find any improvement, when the counter z has reached the number of tasks (cf. line 13). Then, the algorithm stops the approximation to the optimum. Thus, it does not necessarily find the optimal solution: It might have approached a local maximum in this case. Theoretically, the algorithm has a worst case effort of $O(m \cdot n)$, resulting in the effort class $O(n^2)$. In this case, it might evaluate all combinations denoted by the two-dimensional matrix. The best case effort is at

```
Data: $\mathbb{T}, \mathbb{U}, \mathbb{W}_O, \mathbb{W}_C$
Result: array of candidates $l[\,], |l[\,]| = a$
begin
1   $m \leftarrow 0$
2   foreach $\mathbb{S}_i \in \mathbb{U}$ do
3       $cand_i[\,] \leftarrow f_{sort}(y, \mathbb{S}_i)$
4       $l[i] \leftarrow cand_i[m]$
    end
    repeat
5       $m \leftarrow m + 1$
        for $i = 0$ to $a$ do
6           $test[\ i\ ] \leftarrow cand_i[\ m\ ]$
7           $\overline{y} \leftarrow f_{agg}(\ l[\,]\ )$
8           $\overline{x} \leftarrow f_{agg}(\ test[\,]\ )$
9           $x \leftarrow f_{saw}(\ \overline{y},\ \overline{x}\ )$
10          if $x > 0\ \wedge\ f_c(\ test[\,]\ )$ then
11              $l[\,] \leftarrow test[\,]$
12              $z \leftarrow 0$
            else
13              $z \leftarrow z + 1$
            end
        end
    until $z == a$
end
```

Algorithm 3: Bottomup heuristic applied to the selection problem.

$\Omega(n \log n)$, because it at least performs a constraint-selection, which is basically a greedy algorithm.

5.4.4 Pattern-wise Selection

To address the particular problem occurring in parallel arrangements as explained at the beginning of this chapter, a selection algorithm is proposed that takes advantage of the fact that the composition can be modelled by using the composition patterns. As explained before, the parallel cases in a composition are one aspect of the selection problem. Thus, the proposed heuristic algorithm evaluates possible combinations for each identified parallel pattern. This algorithm has already been introduced in previous work about the use of QoS when composing services (cf. Grønmo and Jaeger [39]).

A pseudo-code implementation of the algorithm is given in the listing 4. This algorithm acts from a pattern-perspective. It walks recursively into the structure and identifies pattern elements that do not contain any sub-patterns (cf. line 5). For all tasks within this element, all sets of candidate assignments are evaluated. If the optimal solution for a particular pattern is determined (cf. line 8), the algorithm walks one level upwards to evaluate the assignment within the super-pattern. The aggregated QoS of contained sub-patterns is taken as a fixed value (cf. lines 5 to 7). The pattern-wise optimisation and aggregation are performed until the entire composition is covered and one aggregated QoS is returned. For determining the optimal solution, the listing mentions the function $doGlobalSearch$ which performs a naive evaluation of all possible combinations.

```
Data: $\mathbb{T}, \mathbb{U}, \mathbb{W}_O, \mathbb{W}_C, \mathbb{K}_x$
Result: array of candidates $l[\,], |l[\,]| = a$
pattern( $\mathbb{K}_x, \mathbb{W}_O, \mathbb{W}_C$ )
begin
1   $m = |\mathbb{E}|$
2   foreach $\mathbb{K}_y \in \mathbb{K}_x$ do
3       if $f_{sym}(e) == t_i,\ t_i \in \mathbb{T}$ then
4           $\mathbb{U}' \leftarrow \mathbb{S}_i$
        else
5           $\overline{x}_k \leftarrow$ pattern( $\mathbb{K}_y, \mathbb{W}_O, \mathbb{W}_C$ )
6           $\mathbb{S}'_k \leftarrow \overline{x}_k$
7           $\mathbb{U}' \leftarrow \mathbb{S}'_k$
        end
    end
8   $l[\,] \leftarrow doGlobalSearch(\mathbb{T}, \mathbb{U}', \mathbb{W}_O, \mathbb{W}_C)$
end
```

Algorithm 4: Pattern heuristic applied to the selection problem.

Since this algorithm operates on each pattern element, this approach cannot guarantee to meet global constraints. Furthermore, the fact that the algorithm takes previously aggregated sub-patterns as fixed values also decreases the potential for optimisation for these cases. Thus, this algorithm can only be regarded as heuristic. The main motivation for the algorithm lies in the assumption that the number of tasks within a pattern element will grow more slowly with an increasing total number of tasks. It is assumed that larger compositions will rather likely contain more pattern elements. Therefore, the algorithm might scale better than a brute-force approach, which will be analysed in an evaluation presented in the next chapter.

However, the resulting worst-case effort for this algorithm is $O(m^n)$ as well. The example given in Figure 5.3 of Section 5.1 shows a composition of nine tasks: Assuming that for each task 5 candidates have been found, the number of combinations that the algorithm will evaluate is $5^3 + 5^3 + 5^3 = 375$ vs. all possible combinations which is $5^9 = 1,953,125$. The effort is basically reduced to the sub-element containing the highest number of tasks: $O(m^{n_{sub-max}})$, which has not effect on the worst case effort.

The best case for this algorithm occurs, if each pattern element consists of a single task. The pattern-wise selection would perform a comparison of the candidates by their overall QoS and assign the resulting best candidate. Then, the theoretically lowest effort possible is equivalent to a sort operation which is considered $O(n \log n)$.

5.4.5 Comparison of the Algorithms

Table 5.2 lists the basic attributes of the proposed heuristic algorithms from the previous section. These basic attributes are: whether it supports one or many global constraints, the best and worst case computational complexity and if it always finds the optimal solution.

Algorithm Name	Supports Constraint	Effort Best case	Effort Worst case	Guaratees Optimum
Greedy Selection	No	$\Omega(n \log n)$	$O(n \log n)$	no
Constraint Selection	Yes, 1	$\Omega(n \log n)$	$O(n \log n)$	no
Discarding Subsets	Yes, ≥ 1	$\Omega(n^2)$	$O(m^n)$	no
Bottom-Up Approx.	Yes, ≥ 1	$\Omega(n \log n)$	$O(m \cdot n)$	no
Pattern-wise Selection	No	$\Omega(n \log n)$	$O(m^n)$	no

Table 5.2: Summary of introduced heuristic algorithms.

Chapter 6

Evaluation

The model of the selection problem and the algorithms as given in the previous chapters provide the basis for an implementation that simulates the QoS-based selection by the heuristic algorithms. Based on the simulation software and its setup, different simulation campaigns are introduced. These campaigns are designed to evaluate particular aspects of the simulated QoS-based selection. The campaigns will give the opportunity to explore the strengths and weaknesses of the considered heuristics under special conditions.

To determine the simulation parameters, related research work and publications in the field of service compositions are considered sources for common value ranges of the involved QoS characteristics, as well as sources for the setup of the simulated example compositions. Based on a summary of the related work, this chapter will explain how the basic parameters for the simulation are set. At the end, the results from the simulation campaigns will be presented and discussed.

6.1 Simulation Model

The basic idea of the simulation is to generate problem instances of the previously presented problem and then let the implementation of the heuristic algorithms solve these instances. In this simulation, a solution is an assignment of the chosen candidates to the tasks. The simulation will capture the time taken to compute a solution and the aggregated QoS resulting from the assignment. While best and worst case efforts of the proposed heuristic algorithms are given in Section 5.4.5, a simulation of real world conditions reveals the typical required effort. In order to evaluate the implemented algorithms under different influences, "the" simulation consists of several elements:

- **Campaigns.** The entire simulation is divided into simulation campaigns. Each campaign investigates the influence of a particular parameter on the performance of the algorithms. For example, a simulation campaign tests how the performance of the algorithms develops, if the number of tasks is increased while the other parameters are kept constant.

- **Setups.** Each campaign is divided into a number of setups. While one or more parameters are varied in a campaign in order to investigate their effect on the algorithms, a setup denotes a particular setting of parameters.

- **Runs.** A setup consists of runs. A particular setup is repeated for a number of runs. The repetition is necessary for the analysis of the results, because some parameters require a stochastic parameterisation.

For all simulations, the parameters cover the generation of the problem instances; no parameters are set for the implemented algorithms. Depending on the simulation setup, some of them are fixed, some are step-wisely increased and some are randomly set. Considering the problem model, the elements of a problem instance are generated in the following way:

- **Tasks and candidates.** If a problem instance is generated, only the amount of tasks and candidates is relevant. The simulation ignores particular functionality of possible tasks or possible composition goals, because the focus on this evaluation is the QoS of the candidates only. In the simulation, the amount of tasks and candidates is either step-wisely increased or set to fixed value.

- **Quality-of-Service.** The QoS values of the candidates are stochastically generated. Section 6.3.2 explains the generation mechanism and the value ranges.

- **Optimisation goals.** In the entire simulation, the optimisation goals remain the same and all generated problem instances have the requirement to optimise four QoS characteristics. These characteristics will be discussed further in Section 6.3.1.

- **Constraints.** Since the simulation evaluates the optimisation capabilities, a fixed constraint is set on an optional basis for the evaluation of heuristic algorithms that are capable of meeting constraints.

- **Structure.** The structure of a composition is generated stochastically. However, to evaluate the influence of the structural arrangement, the likeliness of generating particular structural elements (i.e. parallel vs. sequential) can be varied.

In summary, performing the simulation, i.e. performing a *run*, is divided into three main steps. First, a problem instance is generated, then the implemented heuristic algorithms try to solve the given problem and then the taken computation time and the resulting QoS are evaluated. Figure 6.1 outlines these steps.

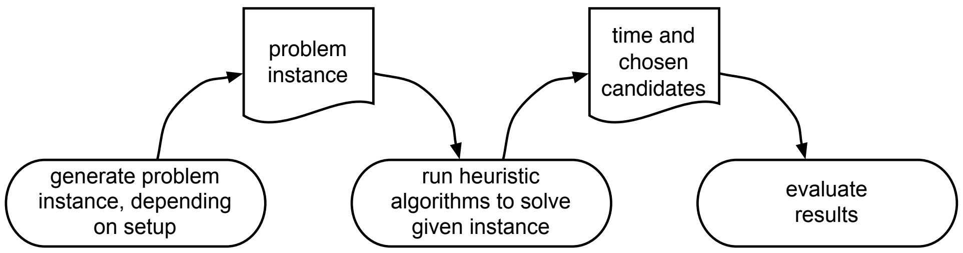

problem instance
time and chosen candidates
generate problem instance, depending on setup
run heuristic algorithms to solve given instance
evaluate results

Figure 6.1: Main steps of performing a simulation run.

6.2 Evaluation Methods and Metrics

Besides capturing the computation time of the algorithms, the simulation also compares the aggregated QoS of the composition resulting from the task-candidate assignments. A comparison makes sense, because the proposed heuristics do not guarantee finding the optimal solution. Then, a quantitative analysis of the resulting QoS shows how much worse a heuristic algorithm is as compared to others. A quantitative statement is achieved by computing a score for the resulting QoS of each selection method. For this score the SAW method is used, which has been introduced in the previous chapter (cf. Section 5.2.4). By this method, one selection method serves as a reference and resulting scores for the other methods indicate their relative QoS improvement or relative loss. For example, if the results in the upcoming sections will present a QoS ratio of 1.20 this means that a method has resulted in a "20%" better QoS when compared to the reference selection. For this comparison, the different QoS categories are aggregated considering equal weights. In summary, the concept of performance covers two measures: the *computation time* of the performed algorithm and *resulting aggregated quality* compared to a reference method.

The simulation implements the presented simulation model and performs the introduced heuristics on randomly generated problem instances. In order to compute a QoS ratio independent from a heuristic method, the simulation provides additional methods to provide references to an optimal solution and to a non-optimised result:

- **Constraint-based.** This method implements the simple case of optimising with regard to a single QoS characteristic. The algorithm shows a greedy behaviour: For each individual task, it chooses the candidate offering the best QoS with respect to the considered constraint characteristic. This algorithm can be used to meet a constraint based on a single QoS characteristic while ignoring other characteristics for optimisation.

- **Random selection.** By this selection, a candidate is randomly assigned to a task. Thus, this method completely ignores the QoS. It simulates the case in which the candidates are chosen by other criteria, such as by their organisational affiliation or by a first-found policy.

- **Global search.** The global selection evaluates all possible combinations and then determines the best resulting assignment. This method shows the worst result in terms of its computation time. However, it determines the best QoS result possible. Therefore, this method serves as a reference for how closely the heuristic algorithms approach the optimal result. This method can optimise different QoS values and it can additionally consider constraints.

In addition to these methods, the discarding subsets method has been implemented in two ways: One variant optimises for different QoS categories and another can take one or more constraints into account. In summary, considering the selection methods for reference and the heuristics as mentioned in the previous chapter, the software simulation implements the following methods:

Algorithm, *constraint capability*	
Constraint *(yes)*	Random *(no)*
Global Constraint *(yes)*	Global Optimisation *(no)*
Discarding Constraint *(yes)*	Discarding Optimisation *(no)*
Bottom-Up Approach *(yes)*	Pattern-based Selection *(no)*
	Local *(no)*

6.2.1 Statistical Measures

As mentioned in the previous section, the simulation software captures two main measures of each run: the score of the aggregated QoS relative to a reference QoS, the QoS ratio, and the computation time. The result of the simulation will be sets of samples that represent a population of these two values (QoS ratio and computation time) for each selection method. Because this work cannot list all the generated and captured data of all campaigns, the results of each campaign are presented in the following way: For each campaign, diagrams will show the average QoS and computation time of each selection method captured for each setup.

In addition to the diagrams, a table will list the results for one particular setup that represents a typical output of other setups within a campaign. For example, if a campaign tests for different setups with an increasing numbers of tasks, a separate table will present the results of one setup with a particular number of tasks. The tables will present following statistical measures derived from the captured data:[1]

- **Arithmetic mean.** The arithmetic mean is interpreted as the average of a set of values. The used definition for the arithmetic mean $\overline{x}$ is as follows: $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

[1]The definitions that follow represent basic knowledge in the field of statistics; the Handbook of Mathematics by Bronstein et al. serves as a reference [13].

- **Standard deviation.** Besides the arithmetic mean, the standard deviation is a measure for how far or how close the individual values occur around the arithmetic mean. For this work, the following definition for the standard deviation s is considered: $s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$ with x_i representing an individual sample, n the count, and $\overline{x}$ the arithmetic mean of the samples.

  In addition, the simulator performs a simple test to determine if the samples are distributed around the given arithmetic mean. For this test, the software determines the percentage of the samples contained in $1|s|$, $2|s|$ and $3|s|$ of the mean. If the distribution show a similarity to the percentage of 66%, 95% and 99% for $1s$, $2s$ and $3s$ around the mean, this indicates that the distribution is shows a concentration around the given mean value. For the interpretation of the results, this test provides an indication about the performance behaviour of a selection method. A method can either show a performance which oscillates around an average, or it can show a split performance. i.e. it performs either very badly or very well.

- **95%-Confidence interval.** It is assumed that the measured computation times and the resulting QoS represent a population parameter with a specific distribution. Thus, the calculated arithmetic mean covers only the captured part of the entire population.

  The confidence interval denotes an interval in which the arithmetic mean of the entire population resides subject to a probability of correctly captured samples. In this simulation sciences, the considered probability is set at 95%. The denotes that 5% of the values are taken as wrong values resulting from errors in the simulation setup or the method to capture the results. Different definitions for the confidence interval exist. These depend on the count of samples. Because the setup will be designed to contain between 50 and 1000 runs, the following definition of the confidence interval is used:

$$\left[\overline{x} - z\Big(1 - \frac{\alpha}{2}\Big)\frac{s}{\sqrt{n}}; \overline{x} + z\Big(1 - \frac{\alpha}{2}\Big)\frac{s}{\sqrt{n}}\right]$$

  The variable α represents the likeliness of failure. For a 95%-confidence interval the assumed failure is 5%. The term $z(x)$ refers to the quantile of the normal distribution. The other variables refer to the same definition as previously given.

6.3 Parameters and Implementation

One of the first parameters to set for the simulation is the size of a typical composition structure. The Gartner Group has presented a study about the average number of services found in companies and enterprises. This gives a rough overview about the dimensions to consider [107]. According to this study, small companies deploy about 25 services on average while very large enterprises deploy a total amount of more than 1000 services. In such very large enterprises, more than 100 clients access these services on a daily average. This study has also revealed that the services in very large enterprises are usually invoked among all units but entirely within the organisation. From these numbers, no concrete estimation is possible. It is possible to estimate that, on average, a client accesses 10 services which indicates roughly the size of possible compositions. The number of 1000 services represents a direction for how many services may be considered by a broker. However, assumptions about how many services candidates result for each task cannot be made.

Besides the numbers from this analysis, scientific publications provide indications about possible sizes of service compositions. Van der Aalst et al. describe the application of workflow modelling techniques to the process of granting a loan for a Danish bank company. In their example, 8 tasks are identified that together form a complete business process [147]. The evaluations of workflow management facilities of Heinis et al. [44] and Cranford et al. [19] mention workflows that usually consist of up to 25 individual tasks or activities. Regarding the typical size of workflows or business processes, medium sized compositions can have about ten tasks, while large compositions can be around 25 tasks in size. It must be also considered that a composition might require a transformation before it can be processed by the aggregation method. Section 4.5 has mentioned these transformations, which involve the duplication of nodes. Consequently, the resulting number of tasks to process by selection algorithms might be larger than the number found in the original business process.

6.3.1 Quality-of-Service Parameters

Regarding the response time, measurements of the invocation of Web services can be considered because Web services represent the most popular SOA implementation today. Tosic et al. have introduced an infrastructure to evaluate the provision of policy-aware Web services [132], which also provides a quantitative measurement of Web service invocation times. In their work, they have performed experiments to execute Web services while providing an adaptive solution to cover dynamic changes in the given QoS. Their experiments show that a standard setup in a local network resulted in a lowest response time of about 150 milliseconds of plain test Web services. Therefore, for this work a value of 150 milliseconds is regarded as the lower end of response time values.

Gillmann et al. have evaluated in their work the typical duration of activities in a workflow. The workflow scenario represents a realistic application of service compositions. According to their evaluation, automatic (non-interactive) activities take from around 2 to 12 seconds [38]. Chandrasekaran et al. have introduced a simulation environment for processes based on Web service compositions [18]. Their measurements show which parts of the service invocation take which amount of time: the queuing of the request, its processing or the transfer over the network. According to their measurements, the service invocations range between 400 milliseconds and 2.6 seconds.

Regarding the availability of services, the work of Gillmann et al. considers a typical downtime in the area of 20 minutes each day in their evaluation [38].[2] This would result in an availability of $0,985\%$. Kenyon presents in his book a very detailed discussion about typical availability rates in service architectures [72, p. 411]. The data is summarised in Table 6.1. Based on these values, it can be concluded that an availability value close to 99% represents a lower limit for the business minded provision of services. Values above the high-availability can be ignored from Table 6.1, because for most cases it can be assumed that the underlying software platform, computer hardware and network connection provide a lower availability.

System class	**Availability**	**Yearly Downtime**	**Daily Downtime**
Unmanaged	90.00000%	876:00 h	2:40 h
Managed	99.00000%	87.60 h	14.40 h
Well-Managed	99.90000%	8.76 h	1.44 m
Fault-Tolerant	99.99000%	52.56 m	8.64 s
High-Availability	99.99900%	5.26 m	863.99 ms
⋮	⋮	⋮	⋮
Ultra-Availability	99.99999%	3.15 s	8.64 ms

Table 6.1: Availability rates and resulting downtimes by Kenyon [72, p. 411].

Regarding the cost and the reputation, an evaluation of existing work would not result in any benefit for this simulation: Cost and reputation are individually set. Moreover, their definitions can vary as explained in Section 4.4.3. For example, the cost depends on the payment model or the considered currency. An amount of a particular currency could require a transformation into another. Regarding the reputation any scale used for setting scores can be considered. For this work, for both categories arbitrary absolute values within a reasonable range are considered. The value ranges of the other values – i.e. the response time, number of tasks and candidates, and availability – are based on the previously given discussion of existing literature. Table 6.2 summarises the value ranges for the parameters of the simulation.

6.3.2 Implementation

The simulation software generates arbitrary problem instances which includes structures and QoS values of the candidates to apply each of the different selection methods. The software considers the four QoS characteristics as described in the

[2]It must be noted that this number represents a simplified assumption of their failure model as explained in [38]. However, for the use in this work such a statement is regarded as sufficient to get an impression about typical dimensions.

Parameter, *value range*		
Number of tasks	[4 . . . 50]	*incremented by 1 or fixed value*
Number of candidates	[2 . . . 50]	*incremented by 1 or fixed value*
Response time	[150 . . . 9999]	*randomly chosen*
Cost	[0 . . . 9999]	*randomly chosen*
Reputation	[0 . . . 10]	*randomly chosen*
Availability	[0.9750 . . . 0.9999]	*randomly chosen*

Table 6.2: Parameter value ranges of the simulation.

previous section: the maximum response time, the maximum cost, the reputation based on the mean-aggregation and the availability. The generation of problem instances involves a number of steps each covering a randomly generated aspect:

1. To build up the composition structure, the software determines a root structure chosen from the composition patterns with equal probability. To build "deep structures", the software first chooses a composition pattern as a root structure. Within this root and inside further structural elements, the simulator chooses with equal probabilities to either generate a task or to generate a structural element that may contain tasks and other structural elements.

   The simulator considers only seven composition patterns from the nine introduced in Section 4.3. This represents a simplification of the composition patterns. It merges the patterns CP_5 with CP_6, and CP_8 with CP_9, to one new pattern each. Which particular element is generated is chosen with equal probability of 14.28% for each. Thus, the generated structures statistically contain more parallel arrangements, because the considered patterns consist of five parallel patterns and two sequential patterns (71.43% versus 28.57%).

2. The software generates candidate services with random QoS values. To ensure a realistic QoS variance of the candidates, the software randomly assigns, for each task, an optimal cost and response time with uniform distribution, from the following intervals:

QoS Characteristic, *value range*		
Response Time	[150 . . . 2000]	*uniformly distributed*
Cost	[0 . . . 1000]	*uniformly distributed*

   Based on the optimal value $q_{optimal}$, the actual value for each candidate is determined by adding a randomly determined percentage between 0 and 100 with uniform distribution. In short the generated QoS value q_g is:

   $$q_g = q_{optimal} \cdot (1 + x) \quad 0 \leq x \leq 1, \quad x \in \mathbb{R}$$

To form a trade-off couple between the response time and cost, the two are set as follows: The value x_1 added to the optimal response time is taken to calculate the value x_2 added to the optimal cost, with $x_1 + x_2 = 1$. Thus, the better the response time is, the worse the cost and vice versa.

3. Contrary to the previous characteristics, the software does not generate an optimal value for generating the reputation values. The reputation is generated by taking a base value of 5 and adding a value $0, \ldots, 10 \in \mathbb{N}$, which is chosen with uniform distribution, to this base value. Regarding the availability, the algorithm considers the interval $[0.975 \ldots 0.999]$ as given in Table 6.2 and chooses, with uniform distribution, a value between these two borders.

4. After the structure and candidate QoS values have been determined, a constraint is determined by running the constraint selection first. In the simulation campaigns, the cost is the considered constraint characteristic. The aggregated value is increased by a set percentage (for example by 20%) and then taken as the constraint that has to be met by the other selection methods.

After the creation of the problem instance, the software performs the selection methods on this setup. For each run, the software captures the resulting aggregated QoS and the computation time in microseconds. The computation is the time that an algorithm needs to determine the solution, based on a given problem instance. Thus, the generation of the instance including the candidates with their QoS values is not captured by the measurements.

6.3.3 Technical Details

Since the computation times are captured to compare the computational efforts of the particular selection methods, the absolute performance of the used hard- and software platform is generally irrelevant for the validity of the results. However, it was necessary to ensure that the simulation host computer kept its condition throughout all the campaigns and that no other processes running in parallel affect the measurements. Consequently, only operating system and a software run-time environment was installed on the host computer, keeping it clean at “factory settings”.

The simulation software is a custom implementation and no external software, libraries and tools were integrated. All selection methods ran in the same environment under the same conditions working on the same data structure in the computer’s main memory. Java was chosen as the implementation language for the simulation software because of its availability on many platforms. Thus, for the distribution of the software, it is ensured that it will be able to run on different hardware platforms and operating systems. Since Java 2 Standard Edition (J2SE) 5.0, the API provides the operation System.nanoTime(), which offers a more precise time measurement as the operation System.currentTimeMillis() provided by previous versions of the Java platform as the only operation of this kind. It turned out that measurements in the magnitude of milliseconds did not capture the computation time from the random, constraint and local selection methods. Therefore, the version 5.0 of the J2SE is the required run-time environment for the simulation software. Setting the parameters to create the problem instances involved the generation of random numbers. Accordingly, the pseudo-random number generator implementation of the class java.util.Random (with default seed) was used for this purpose.

The entire simulation was performed on the same computer keeping the same software platform. As a computer, standard PC hardware with the Windows 2000 operating system was used. The operating system was kept at its default state, meaning that no additional driver software was installed and no settings were made to additionally configure the computer. As a consequence, the computer did not maintain any network connection and ran in standard 680 by 480 graphics mode with 16 colors. The only software that was installed is the Java virtual machine to let the computer run the simulation software. Further technical specifications about the computer are given in the Appendix A.

6.4 Simulation Campaigns and their Results

In the following, different simulation campaigns are described in which influence of a particular parameter on the resulting performance of the algorithms is investigated. For example, it is obvious that a rising number of tasks will result in a rising computational effort. However, regarding a raising number of candidates with a fixed number of tasks, one can guess that the algorithms scale differently because of different upper bounds of complexity. Another question is how much potential for optimisation exists for a given problem instance, since this potential depends on the variance of the provided candidate QoS. To cover these considerations, the following different simulation campaigns have been designed that allow the evaluation of different performance factors:

Simulation campaigns, *short description*	
C1	Increasing number of tasks without constraint
C2	Increasing number of tasks with constraint
C3	Increasing number of service candidates with a fixed number of tasks
C4	Volatility of the QoS among the candidates
C5	Parallel vs. sequential composition structures

In the following sections, these campaigns are discussed and their results are analysed. Regarding the simulation setup, two general issues have been set that apply to all campaigns. The number of conducted runs per setup was set to 200 times. From preliminary tests, it turned out that for this number of runs it is ensured that a campaign will result in similar statistical results when run again. Because the “exponential” selection methods are indeed very time-consuming to simulate, the software skips a method if the average time has exceeded a certain limit. For the campaigns, the simulator has skipped a method if its average computation time in a run has exceeded 500 seconds.

6.4.1 Increasing Number of Tasks without Constraint (C1)

As a start, this simulation campaign has the goal to test the resulting performance with an increasing number of tasks. For this campaign, the number of candidates is kept constant. Regarding the setup of the QoS, this campaign does not set any constraints but it considers the four QoS characteristics for optimisation. As explained in the previous section, the response time and cost will form a trade-off couple, meaning that optimising one generally results in making the other worse.

The reference method for comparing the resulting QoS will be set to the constraint-based selection. Although there is no constraint to meet, this method optimises for just one QoS characteristic and thus serves as a simple QoS optimisation approach. Because, contrary to the local and pattern optimisation, the bottomup optimisation considers also constraints, this method will be skipped for this campaign and explored in the next one. In summary, this campaign uses the following setup:

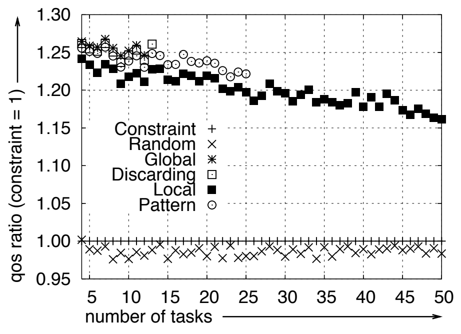
C1 Parameters and Setup, *values*	
Number of tasks	[4 . . . 50], *each setup incrementing by 1*
Number of candidates	5, *constant*
Candidate QoS	as given in Table 6.2, randomly set, uniformly distributed
Constraint	no constraint set
Involved methods	constraint, random, global, discarding, local and pattern
Reference method	constraint

Expectations

This campaign will evaluate how the different heuristics perform when faced with the problem of optimising for different QoS characteristics while two of them form a trade-off couple. A constraint is not considered. Generally, this campaign will evaluate how well the heuristics optimise a given setup when compared with the three reference methods.

Results

The results of this simulation campaign are shown in Figures 6.2 and 6.3. In addition, Table 6.3 shows the results of the setup with 12 tasks. Figure 6.2 shows the average resulting aggregated QoS performing the selection methods relative to the constraint selection. Figure 6.3 shows the average computation times of the different selection methods for problem instances with increasing number of tasks;



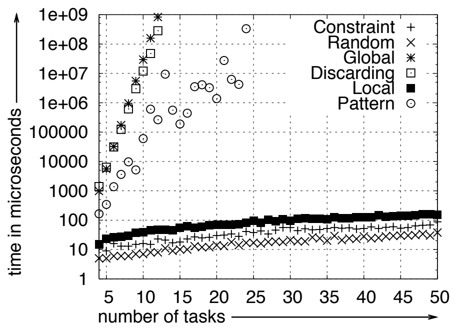


Figure 6.2: Relative QoS to constraint selection (C1, with 5 candidates).

Figure 6.3: Computation times of selection methods (C1, with 5 candidates).

this diagram has a logarithmic scale on its y-axis. Table 6.3 lists the arithmetic mean, the standard deviation and the confidence interval. Based on the standard deviation, the table shows how many percent of the samples are within one, two or three standard deviations around the mean.

Method	**Average Mean $\overline{x}$**	**Standard Deviation s**	**% in 1s**	**% in 2s**	**% in 3s**	**95%-Conf.**
Overall QoS relative to constraint selection						
Constraint	1.0	0.0	0.0	0.0	0.0	± 0.0
Random	0.9808	0.0937	76.50	96.00	98.50	± 0.0110
Glob O.	1.2463	0.0988	73.00	94.00	99.50	± 0.0116
Disc O.	1.2398	0.1022	72.00	93.00	100.00	± 0.0120
Local	1.2110	0.1068	73.00	95.50	99.00	± 0.0125
Pattern	1.2303	0.1019	71.00	94.00	99.50	± 0.0120
Computation duration, in microseconds						
Constraint	23	51	98.50	98.50	98.50	± 6.0040
Random	8	1	23.50	69.00	95.00	± 0.1177
Glob O.	831,559,679	211,098,114	69.00	95.00	100.0	± 24,485,178
Disc O.	285,281,163	184,650,968	21.00	84.00	87.00	± 21,738,258
Local	48	46	98.50	98.50	98.50	± 5.4154
Pattern	263,332	2,098,027	98.50	98.50	99.00	± 246,992

Table 6.3: QoS and times, setup with 12 tasks in C1.

Both the pattern and the local method show a resulting average QoS performance that is very close to the algorithm that always finds the optimal solution. In all runs, the pattern method shows the better resulting QoS when compared to the local method. The gap between the pattern method and the local method is relatively small. The discarding method shows almost the same QoS performance as the global method.

Regarding the computing duration, the constraint, random and local methods show a slow increase – as expected. The results for the pattern method show that, with a growing number of tasks, its computation exceeds the given time limit and thus was skipped after the setup with 25 tasks. However, its computation times climb slower than the times of the global or the discarding method. Compared to the global method, the pattern method performs significantly better: The average computation time for a setup of 12 tasks is less than 0.1% of the average computation time for the global selection.

The histogram of the computation times in Figure 6.4 explains the behaviour of the pattern method more closely. For more than 40% out of the 200 samples, the algorithm executes faster than 0.001 seconds. However, for 7% of the samples the algorithm takes more than 0.1 seconds for execution. This shows the volatility of the required computational efforts by this method. For comparison, Figure 6.5 shows the histogram of the computation duration of the global method. Although the values are larger than those resulting from the pattern method, they do not show that large variance.

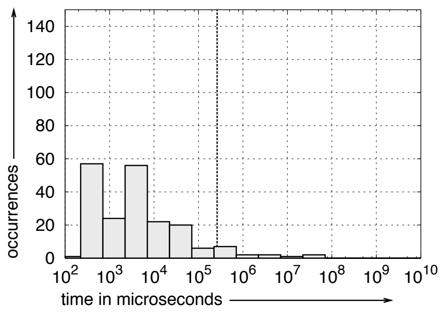


Figure 6.4: Histogram of computation times of the pattern method (C1, setup with 12 tasks).

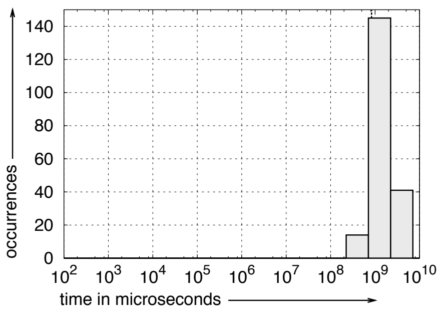


Figure 6.5: Histogram of computation times of the global method (C1, setup with 12 tasks).

Interpretation

This campaign has evaluated the performance of the three heuristics, the discarding, local and pattern methods. The results allow the following interpretation about these three:

- **Discarding.** This method results in almost the same QoS performance as the global method that always finds the optimal solution. However, this campaign shows that the bounding ability of this heuristic cannot save enough efforts to show a substantial difference. The results show that for larger number of tasks, the bounding saves more than 50% of the computational effort, but the effort still rises exponentially. Accordingly, its computational performance can be regarded as poor.

Looking at the results for very small composition sizes, the discarding method takes even longer time than the global method. Obviously, the discarding method cannot save more time by bounding sub-trees than it consumes more for testing for bounding after each step.

- **Local.** The locally optimising method shows a QoS performance that comes very close to the pattern, global and discarding methods. From the generated data of this campaign, which is not entirely presented here, the local method showed that in about 95% of the runs, this selection method results a minimum QoS of about 15% worse than the average of the optimal solution. As expected, it performs with a negligible computational effort.

- **Pattern.** The results from this campaign, especially the histogram shown in Figure 6.4, allow the conclusion that the algorithm strongly depends on the given structure of the composition. The computational effort grows exponentially for the evaluation of the combinations within a pattern. If a pattern consists of many tasks to evaluate, the number of possible combinations rises exponentially. The simulation has shown that these cases occur and they result in an extremely large computational effort. Compared with the two other heuristics tested here, this method shows a QoS performance between the local and the discarding methods.

6.4.2 Increasing Number of Tasks with One Constraint (C2)

This simulation campaign has the goal to test the resulting performance with an increasing number of tasks. For this campaign, the number of candidates per task is kept constant. The particular QoS characteristics are generated using a normal distribution. Consequently, each of the values within the range is possible with equal probabilities. For the constraint, one QoS characteristic is chosen. Then, the required constraint value is set to 20% above the best value possible. The reference method for comparing the resulting QoS will be set to the constraint-based selection. In summary, this campaign uses the following setup:

C2 Parameters and Setup, *values*	
Number of tasks	[4 . . . 50], *each setup incrementing by 1*
Number of candidates	5, *constant*
Candidate QoS	as given in Table 6.2, randomly set, with uniform distribution
Constraint	1 characteristic, 20% worse than best possible for constraint-aware methods
Involved methods	constraint, random, global, discarding and bottomup
Reference method	constraint

Expectations

The expectations are similar as in the previous campaign except that this campaign involves a constraint and thus, instead of the local and pattern heuristics, the focus lies on the discarding and bottomup heuristics. The anticipated result is a quantitative evaluation about how well each of the algorithms scales with a rising number of tasks and if the resulting QoS of particular algorithms will decrease. Obviously, a larger composition structure will include a higher number of branches, which could result in more potential to optimise. On the other hand, a larger number of tasks will reduce the optimisation level, because the QoS increase of a particular task will be relatively low compared to the aggregated QoS of the entire composition.

Simulation Results

The results of this campaign are shown in Figures 6.6 and 6.7. In addition, Table 6.4 shows the results of the setup with 12 tasks.Figure 6.6 shows the average resulting aggregated QoS performing the selection methods, relative to the constraint selection. Figure 6.7 shows the average computation times of the different selection methods for problem instances with increasing number of tasks. It must be noted that its y-axis has a logarithmic scale. Table 6.4 lists the arithmetic mean, the standard deviation and the corresponding confidence intervals. Based on the standard deviation, the table shows also how many percent of the samples are within one, two or three standard deviations around the mean.

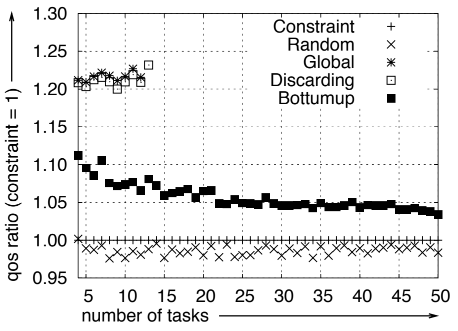


Figure 6.6: Relative QoS to constraint selection (C2, with 5 candidates).

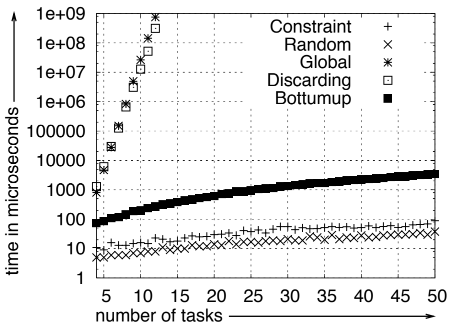


Figure 6.7: Computation times (C2 with 5 candidates).

The results show that the exponentially time-bounded methods indeed show a steep increase in the computational duration. Compared to these, the quadratic bottomup method shows a relatively slow increase. Table 6.4 shows that the discarding method has a high volatility regarding its duration, while the global seems to operate on constant level. In fact, the confidence intervals show that the given mean

Method	Average Mean $\overline{x}$	Standard Deviation s	% in 1s	% in 2s	% in 3s	95%-Conf.
Overall QoS relative to constraint selection						
Constraint	1.0	0.0	0.0	0.0	0.0	± 0.0
Random	0.9808	0.0937	76.50	97.50	98.50	± 0.0110
Glob Co.	1.2154	0.0899	73.00	94.50	99.00	± 0.0105
Disc Co.	1.2089	0.0934	72.00	94.00	99.00	± 0.0110
Bottomup	1.0655	0.0603	83.00	96.00	98.00	± 0.0070
Computation duration, in microseconds						
Constraint	23	51	98.50	98.50	98.50	± 6.0040
Random	8	1	23.50	69.00	95.00	± 0.1177
Glob Co.	750,982,559	207,036,882	68.00	95.00	100.00	± 24,373,668
Disc Co.	309,828,060	176,015,663	19.50	81.50	84.50	± 20,721,656
Bottomup	268	98	84.00	94.00	95.00	± 11.537

Table 6.4: QoS and times, setup with 12 tasks in C2.

value for the duration is not representative and the occurring values are widely spread. To examine this behaviour further, Figure 6.8 shows a histogram of the computation times for the discarding method. Each bar shows the time in microseconds. It must be noted that the x-axis in this diagram has a logarithmic scale. From this histogram it can be seen that about 50% of the runs perform faster than 100 seconds while the average is at 309 seconds due to some runs with require a high effort.

The bottomup method shows a relatively large standard deviation and confidence interval of its duration. Also when observing the complete results, which are not listed, the confidence interval decreases from $1/10$th of the mean (at 12 tasks) to $1/20$th of the mean (at 40 tasks). To provide additional insights on the resulting population, the histogram of duration from the bottomup method is given in Figure 6.9. The histogram shows an accumulation around the average. However, in some cases the values are three times larger. The computation times of the random and constraint selection are relatively low and therefore is ignored in the further discussion.

Regarding the resulting QoS, the global and the discarding selection are on the same level, indicating that the discarding method results in almost the same QoS as the global selection. The bottomup method shows a decreasing level of performance with a rising number of tasks when compared to the constraint selection. The 95%-confidence interval values, as given in the table, indicate that the average closely represents the actual performance of the selection methods.

Interpretation

Comparing the constraint selection and the global selection reveals that, on average, the overall QoS can be improved by about $25\% - 30\%$ when considering

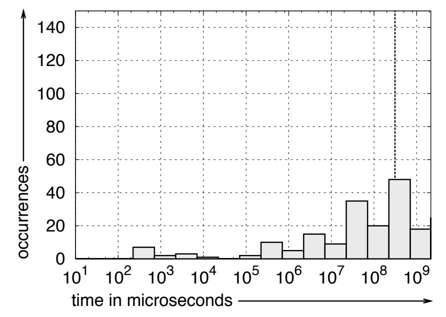


Figure 6.8: Histogram of computation times of the discarding method (C2, with 12 tasks).

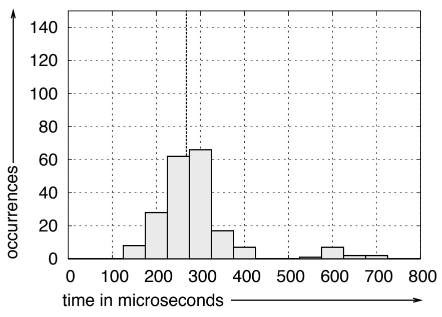


Figure 6.9: Histogram of computation times of the bottomup method (C2, with 12 tasks).

the setup of QoS values as expected. The global and discarding methods show an exponentially rising computational effort with a larger number of candidates and therefore can be regarded as unfeasible. The results allow the following interpretation about the two evaluated heuristics:

- **Discarding.** The selection by discarding subsets results in the best QoS possible, while still meeting the constraint for the setups. The heuristic aspect of the discarding subset algorithm, as discussed in Section 5.4.2, is very small; rather this method resulted in almost the same QoS as an algorithm that is guaranteed to always find the optimal solution.

  Regarding the computational duration, this selection method has an unattractive scaling. Obviously the time performance is highly volatile and seems to depend strongly on the generated example problem instances: In one third of the cases, this method solves the problem within a negligible effort while in 15% of the cases, the duration is about 100-times as high. A future simulation campaign with the focus on different structures should to examine this aspect further.

- **Bottomup.** The bottomup selection results in the second worst QoS, compared to the other heuristics. The results for the QoS values show that the bottomup method reaches about one third of the optimal performance achieved by the global selection. However, it meets the constraint and shows a low computational effort and scales well with a rising number of tasks. The bottomup method shows feasible efforts even for larger composition structures.

In summary, this campaign has confirmed that the branch-and-based approach is not guaranteed to save computational efforts in special cases. But it resulted in almost the best QoS possible. The bottomup method behaves in the opposite way: It has low computational requirements combined with a relatively low QoS performance.

6.4.3 Increasing Number of Service Candidates (C3)

For this campaign, the number of tasks in a problem instance is kept constant for each test run. Like for the previous campaign, the particular QoS categories are generated using a normal distribution - thus this means that each of the values within the range are chosen with equal probabilities. A constraint is considered for this campaign by the constraint-aware methods.

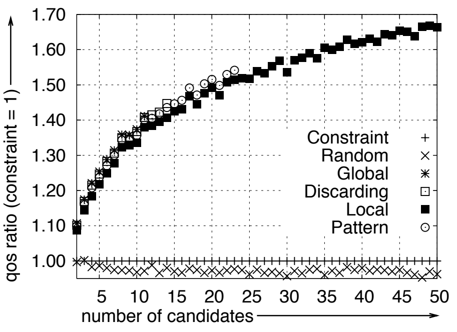
C3 Parameters and Setup, *values*	
Number of tasks	8, constant
Number of candidates	[2 . . . 50]
Candidate QoS	as given in Table 6.2, uniform distribution
Constraint	20% worse cost than best possible for constraint-aware methods
Involved methods	constraint, random, global (two), discarding (two), bottomup, local and pattern
Reference method	constraint

Expectations

The expected result is a quantitative analysis of how well each of the algorithm scales with a rising number of candidates. Compared with the rising number of tasks, increasing the number of candidates results in an effort bounded by a polynomial, because the number of tasks is kept constant. For the calculation of the worst-case effort, the number of tasks goes into the exponent of a time-bounding function, as explained in Section 5.3.5. For this campaign, the exponent is kept at 8. Therefore, the worst-case effort for this setup results in the class of $O(n^8)$ where n represents the number of candidates. A statement is expected on how the polynomial computational efforts will compare to the setup of the first two campaigns with an exponentially bounded effort.

Results

Figures 6.10 and 6.11 show the average aggregated QoS performing the selection methods relative to the constraint selection. Figures 6.12 and 6.13 show the average computation times of the involved selection methods. In addition, Table 6.5 shows the results of the setup with 11 candidates, because this setup still involves the global and discarding selection methods. For setups with more than 11 candidates, the global selection was skipped. The table lists the arithmetic mean, the standard deviation and the corresponding confidence intervals. Based on the standard deviation, the table shows how many percent of the samples are within one, two or three standard deviations around the mean.



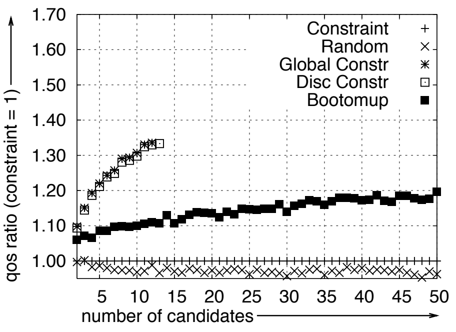
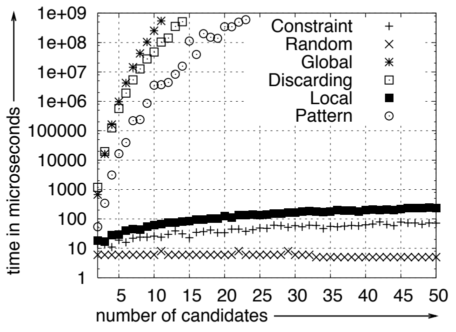


Figure 6.10: Relative QoS to constraint selection of optimisation-only methods (C3).

Figure 6.11: Relative QoS to constraint selection of constraint-aware methods (C3).



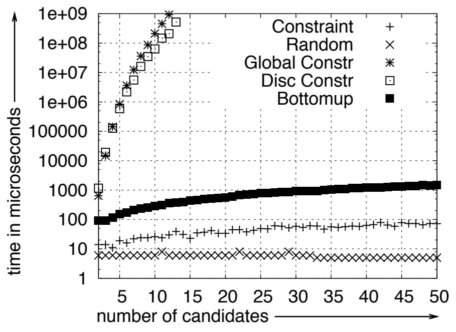


Figure 6.12: Computation duration of optimisation-only methods (C3).

Figure 6.13: Computation duration of constraint-aware methods (C3).

Regarding the QoS performance, all methods show an increase with a rising number of candidates when compared to the constraint selection. The previous campaigns have shown that an increasing number of tasks resulted in a decreasing behaviour of the QoS performance, compared with the constraint selection. In this campaign, all methods have shown an improving QoS performance for a rising number of candidates when compared to the constraint selection. The results also show a rather steep slope at the beginning, which seems to lower for larger number of candidates.

Regarding the computational efforts, the results show a similar development than in the previous two campaigns. The constraint, the random and local selection show at seldom runs a computation time of about 400 microseconds. In general

Method	Average Mean $\overline{x}$	Standard Deviation s	% in 1s	% in 2s	% in 3s	95%-Conf.
Overall QoS relative to constraint selection						
Constraint	1.0	0.0	0.0	0.0	0.0	± 0.0
Random	0.9715	0.1028	70.00	97.00	98.50	± 0.0121
Glob O.	1.4114	0.1403	75.50	94.50	98.50	± 0.0165
Glob Co.	1.3304	0.1339	75.00	93.50	99.50	± 0.0157
Disc O.	1.4074	0.1404	75.00	95.00	98.50	± 0.0165
Disc Co.	1.3243	0.1331	74.50	94.00	99.50	± 0.0156
Local	1.3801	0.1448	75.00	95.00	98.00	± 0.0170
Bottomup	1.1062	0.1050	83.50	96.50	98.00	± 0.0123
Pattern	1.4005	0.1407	74.00	95.00	98.00	± 0.0165
Computation duration, in microseconds						
Constraint	24	44	98.50	98.50	98.50	± 5.1799
Random	8	28	99.50	99.50	99.50	± 3.2963
Glob O.	544,394,247	125,620,708	68.00	93.00	100.0	± 14,788,850
Glob Co.	454,826,580	122,039,371	67.00	94.00	100.0	± 14,367,230
Disc O.	103,781,595	220,063,566	87.00	92.50	94.00	± 30,498,693
Disc Co.	147,531,815	250,731,490	81.00	89.00	95.00	± 34,748,973
Local	67	54	96.50	97.00	97.00	± 6.3572
Bottomup	304	54	85.50	94.00	94.50	± 12.125
Pattern	3,687,662	27,872,646	98.00	99.00	99.00	± 3,281,341

Table 6.5: QoS and times, setup with 11 candidates in C3.

these three methods showed a rather continuous computation duration within one setup which usually varies for a couple of microseconds. However, for a few runs, the results show an abnormality. A histogram of computation times by the local method shown in Figure 6.14 shows that most values are within the range of few microseconds. It can be assumed that these captured “break-out” times resulted from temporary anomalies of the testing platform. The referring histogram in Figure 6.15 shows that the captured values are widely distributed. Furthermore, the histogram shows the pattern of an discrete distribution. It must be noted that the y-axis of this histogram has a logarithmic scale.

Interpretation

Although the worst case effort of the algorithms is polynomial bounded, the computation time of the algorithms increases as fast as for the previous two campaigns. Since the number of possible combinations rose by n^8, the computation time for the first five setups rose quicker than for the setups with rising number of tasks and a fixed number of 8 candidates. The results show that even for relatively low composition sizes the global method results in an intolerable effort. Regarding the involved heuristics, the following conclusions can be drawn:

- **Discarding.** Like in the previous campaigns, this heuristic results in a QoS

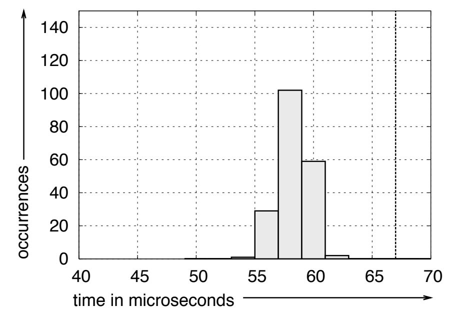


Figure 6.14: Histogram of computation times for the local method (C3, 11 candidates).

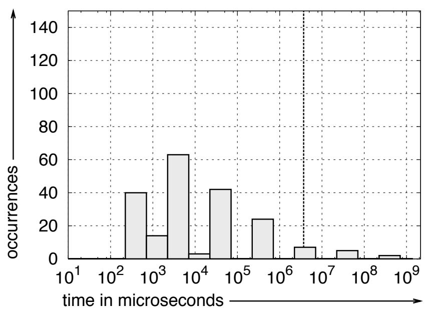


Figure 6.15: Histogram of computation times for the pattern method (C3, 11 candidates).

performance that is similar to the global selection. However, the quickly rising duration makes clear that its application is unfeasible.

- **Local.** The local method shows a resulting QoS that is very close to the global method. Considering the resulting QoS from the pattern method, this approach offers similar QoS performance while showing negligible computational efforts.

  An interesting phenomenon is the rising QoS performance with a higher number of candidates when compared to the constraint selection. With a rising number of tasks (as performed in the previous campaign), the results showed a decreasing performance: The impact when optimising a single critical task becomes relatively small with a larger total number of tasks. A fixed number of tasks provides a constant optimisation impact when improving a single critical task. In addition, a higher number of candidates seems to result in a higher potential to improve the QoS.

- **Bottomup.** The bottomup method confirms its good computational performance. Regarding the QoS performance, the improvements are clearly not as good as for the other heuristic methods; it seems to achieve $1/3$ of the performance of the other methods.

- **Pattern.** The pattern selection results in a QoS performance that is higher than that of the local selection, but not as good as that of the discarding selection. Like in previous campaigns, the pattern method shows an unfeasible effort with a rising number of candidates, even when considering that it performs by magnitudes quicker than the discarding selection. In addition, this method shows an increasing QoS performance with a rising number of candidates as well, confirming the assumption that the optimisation potential generally improves regardless of the considered method.

The pattern method showed also a distribution of captured computation times that looks similar to a discrete distribution (cf. Figure 6.15). It can be assumed that the computation times are directly dependent on the randomly generated number of tasks found within a single pattern. Considering the bars given in Figure 6.15, the rightmost could result from compositions where a pattern with 8 tasks was generated, the second-right results from a pattern with 7 tasks etc.

6.4.4 Volatility of the Quality-of-Service (C4)

For this campaign, the number of tasks and candidates of a problem instance is kept constant for each test run. Like for the previous campaign, the particular QoS categories are generated using an uniform distribution. In addition, the volatility of the QoS values of the candidates is varied: The goal is to test setups in which the QoS values show large differences among the candidates. In contrast to this, the obtained results will be compared to setups in which the QoS values among the candidates remain almost the same.

To create the variation among the generated QoS values, an additional factor, the variance factor q_v, is included when the QoS of the candidates is generated. Based on the QoS generation explained in the previous Section 6.3, the generated QoS value is:

$$q_g = q_{optimal} \cdot (1 + q_v x) \quad x = 0, \ldots, 1$$

For the simulation, the different setups were performed with different values of q_v starting from 0.1 and incremented by about 0.16 for 50 times. The Table 6.6 shows examples for generated QoS values at different values of q_v. It must be noted that each set of 10 example candidates refers to a different task with different best QoS value. This campaign is different from the previous campaigns, because the simulator repeated the simulation test setup at different levels of q_v, and not different amounts of candidates or tasks.

q_v = **0.1**				q_v = **1.0**				q_v = **4.0**			
1762.0	264.0	5.0	0.998	1844.0	1085.0	5.0	0.991	4564.0	1185.0	17.0	0.988
1742.0	267.0	5.0	0.999	1523.0	1232.0	6.0	0.998	2072.0	2949.0	12.0	0.995
1709.0	272.0	5.0	0.998	2312.0	0872.0	6.0	0.993	1648.0	3249.0	7.0	0.956
1753.0	265.0	5.0	0.999	2330.0	0864.0	10.0	0.997	1477.0	3370.0	5.0	0.954
1742.0	267.0	5.0	0.997	1491.0	1247.0	8.0	0.981	1990.0	3007.0	17.0	0.929
1714.0	271.0	5.0	0.998	2028.0	1001.0	5.0	0.976	2345.0	2755.0	18.0	0.995
1710.0	272.0	5.0	0.999	1940.0	1042.0	5.0	0.999	1107.0	3632.0	13.0	0.930
1804.0	257.0	5.0	0.997	2045.0	0994.0	9.0	0.981	4210.0	1436.0	13.0	0.978
1769.0	263.0	5.0	0.998	1668.0	1166.0	7.0	0.996	2648.0	2541.0	11.0	0.905
1704.0	273.0	5.0	0.999	2307.0	0874.0	9.0	0.990	4310.0	1365.0	13.0	0.900

Table 6.6: Examples of generated QoS values at different q_v.

C4 Parameters and Setup, *values*	
Number of tasks	8, constant
Number of candidates	5, constant
Candidate QoS	value range altered by q_v, from 0.1 to 8, by increments of 0.16, uniform distribution
Constraint	20% worse cost than best possible for constraint-aware methods
Involved methods	constraint, random, global (two), discarding (two), bottomup, local and pattern
Reference method	constraint and random selection

Expectations

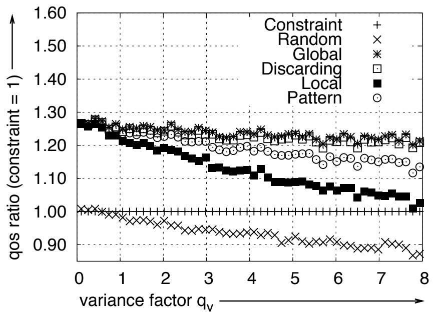
The expected result for this campaign differs much from the other. The expectation is to evaluate changes in the QoS performance of an algorithm if the volatility of the generated QoS increases. Moreover, the selection methods are expected to show individual changes in their QoS performance: A heuristic method might result in poor performance compared to the optimal solution if the setup offers a high optimisation potential.

Results

Figure 6.16 shows the average resulting aggregated QoS performing the selection methods that are capable of optimisation only (e.g. the local method) relative to the constraint selection. In the case that no constraint must be hold, the constraint selection just optimises for one QoS characteristic. Figure 6.17 shows the same for the constraint-aware selection methods (e.g. the bottomup method) with a given constraint. In addition to the these diagrams, Figures 6.18 and 6.18 show the QoS ratio relative to the random selection. In the same manner, Figures 6.20 and 6.21 show the average computation times. It must be noted that the y-axis has a logarithmic scale in both figures. Table 6.7 lists the arithmetic mean, the standard deviation s and the corresponding confidence intervals for the setup with $q_v = 2$. Based on the standard deviation, the table shows how many percent of the samples are within one, two or three standard deviations around the mean.

The results in Figures 6.20 and 6.21 show that an increasing q_v results in a slight decrease of the computation time for the methods based on a branch-and-bound approach. These methods also show a strong volatility of the computation time (cf. Table 6.7). However, except from the discarding methods, the variation of the q_v does not appear to have an impact on the times. The pattern method shows an intense oscillation around a fixed level. In comparison, the values given by Table 6.7 show a high deviation and a high confidence interval, indicating that the given average values are highly volatile. The level of typical computation times of the pattern selection appears to be two magnitudes lower than the corresponding levels of the discarding and global selection.

The results regarding the QoS performance show that most selection methods



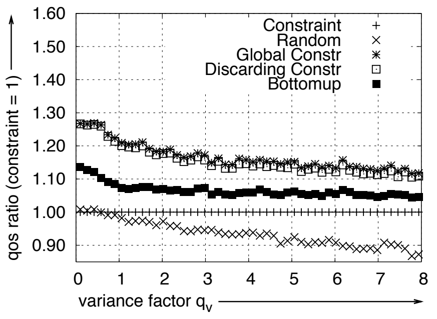
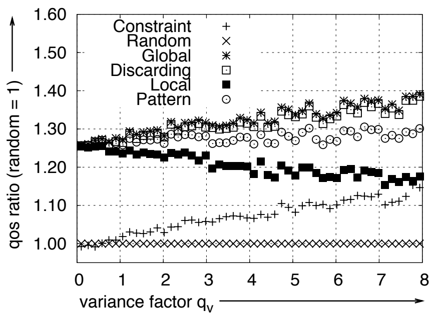


Figure 6.16: Rel. QoS to constr. selection: optimisation-only methods (C4).

Figure 6.17: Rel. QoS to constr. selection: constraint-aware methods (C4).



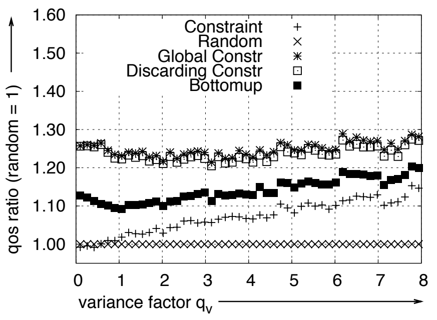


Figure 6.18: Rel. QoS to random selection: optimisation-only methods (C4).

Figure 6.19: Rel. QoS to random selection: constraint-aware methods (C4).

become worse with a rising q_v when compared to the constraint selection (cf. Figures 6.16 and 6.17). However, the decrease of the random selection indicates that the constraint selection performs exceptionally well with a rising q_v. For that reason, a relative comparison of the selection methods to the random selection is given in Figures 6.18 and 6.19. From this point of view, all selection methods except the local selection show an increasing QoS performance with a rising q_v. Among them, the constraint method shows the strongest climb. The local method shows a decrease of the QoS performance when compared with any of the other selection methods. For high values of q_v, it decreases down to the level of the constraint selection. Comparing the bottomup with the discarding method in Figure 6.19 shows that the QoS performance of the bottomup method climbs slightly stronger than the other. From the values given in Table 6.7 it can also be seen that the performance shows the smallest deviation. The pattern method shown in Figure 6.18 appears to result in a noticeable decrease of the QoS performance when compared to the

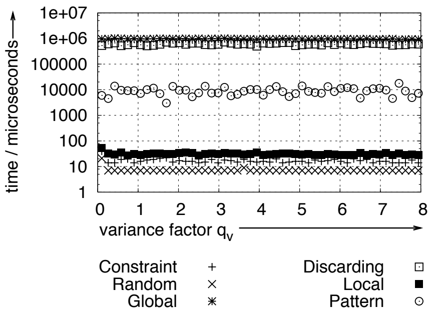


Figure 6.20: Computation duration: optimisation-only methods (C4).

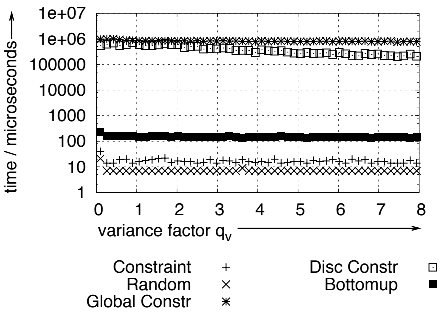


Figure 6.21: Computation duration: constraint-aware methods (C4).

global and discarding selection.

Interpretation

Compared to the random selection, the global methods (the optimising only and the constraint aware) show a rising QoS performance when the variance of the QoS values increases among the candidates. This allows the conclusion that a higher variance creates a higher potential for optimisation. Considering only one QoS characteristic, the QoS optimisation potential rises even stronger when compared to the global selection.

- **Discarding.** The discarding selection results in small savings regarding the computational efforts for small values of q_v and larger savings for a higher value of q_v, when compared to the global selection. Obviously, this method is able to skip combinations more efficiently the more the QoS is varied.

- **Local.** For small values of q_v, the local method shows a high QoS performance when considering the global selection as reference. For high values of q_v, the local selection shows no difference in the overall QoS when compared to the constraint selection, which optimises only by one characteristic.

  The simulations also show that optimising one QoS category, which is done by the constraint selection, has almost no effect on the overall QoS for low values of q_v. However, it must be noted that a constraint regarding one QoS category can be taken into consideration.

- **Bottomup.** Regarding the computation times, the bottomup method confirms the impression of the previous campaigns. It results in low efforts and keeps constant throughout the different setups. It shows a relatively low QoS performance, like in other campaigns. However, it must be noted that com-

Method	Average Mean $\overline{x}$	Standard Deviation s	% in 1s	% in 2s	% in 3s	95%-Conf.
Overall QoS relative to constraint selection						
Constraint	1.0	0.0	0.0	0.0	0.0	± 0.0
Random	0.9720	0.1052	76.00	97.00	98.50	± 0.0123
Glob O.	1.2458	0.1053	72.50	95.50	98.50	± 0.0124
Glob Co.	1.1826	0.0940	75.00	95.50	98.50	± 0.0110
Disc. O.	1.2427	0.1072	73.00	94.50	98.50	± 0.0126
Disc. Co.	1.1768	0.0933	74.50	95.50	98.50	± 0.0109
Local	1.1914	0.1207	74.00	95.50	98.50	± 0.0142
Bottomup	1.0691	0.0787	91.50	96.00	98.00	± 0.0092
Pattern	1.2266	0.1132	70.50	95.50	98.50	± 0.0133
Computation duration, in microseconds						
Constraint	17	33	99.00	99.00	99.00	± 3.8849
Radom	7	0.0	0.0	0.0	0.0	± 0.0
Glob O.	968,646	225,758	65.50	95.50	100.0	± 26,577
Glob Co.	810,440	217,805	66.00	95.00	100.0	± 25,641
Disc. O.	680,862	721,339	83.50	94.50	98.50	± 84,920
Disc. Co.	592,478	642,166	80.50	96.50	98.00	± 75,599
Local	34	47	98.00	98.00	98.00	± 5.5331
Bottomup	159	88	92.50	93.00	93.00	± 10.359
Pattern	9,488	42,495	97.00	97.00	99.00	± 5,002.7

Table 6.7: QoS and times, setup with $q_v = 2$ in C4.

pared to the other selection methods, except the constraint selection, it keeps an acceptable level.

- **Pattern.** The pattern selection shows a smaller decrease of the QoS performance for larger values of q_v when compared to the local selection. The small gap between this and the local selection that was noticed in the previous campaigns becomes larger with a higher variance among the offered QoS. This campaign has also confirmed the volatile behaviour of the pattern heuristic regarding its computational efforts.

6.4.5 Parallel vs. Sequential Composition Structures (C5)

As mentioned in the introduction on the QoS-based selection in Section 5.1, a specific characteristic of the selection problem lies in determining the combination of candidates in the parallel arrangements. This campaign will vary the structural elements used for the problem instances to evaluate the performance of the different algorithms with an increasing number of parallel arrangements.

The first setup starts with all-sequential structures. For the subsequent setups, the simulator generates parallel patterns with increasing probability. A probability value of 20% means that there is a 20% probability that a parallel pattern is created, and an 80% probability that a sequential only is created. The last setup is performed at a probability value of 100%. In this case, the generated structures consist entirely of parallel patterns.

C5 Parameter / Setup *(Value)*	
Number of tasks	8, constant
Number of candidates	5, constant
Candidate QoS	as given in Table 6.2, uniformly distributed
Constraint	20 % worse cost than best possible for constraint-aware methods
Involved methods	constraint, random, global (two), discarding (two), bottomup, local and pattern
Reference method	constraint
Structure	starting with sequential structures with rising probability of parallel structures, incrementing each setup by 2%

Expectations

The expected result from this campaign is a qualitative statement about how the QoS and computational performance of the selection methods depend on the structure of the composition. For example, for all-sequential structures, an algorithm might perform very fast when compared to parallel structures. Regarding the QoS performance, this campaign will show if any of the proposed heuristics will perform worse or better with a rising ratio of parallel patterns.

Results

The results regarding the QoS performance are shown in Figures 6.22 and 6.23. The computation times are given in Figures 6.24 and 6.25. In all figures, the dashed lines indicate the probability value that was applied in the four previous campaigns. In addition, Table 6.8 shows the results of the setup with a probability value of 100%.

The global selection and the three heuristics show an increase of the QoS performance with a rising number of parallel structures. Because the global selection results in the optimal solution, this indicates that the potential for the optimisation of the QoS increases with the ratio of parallel structures. As a qualitative statement, the local and the pattern selection show similar improvements while the pattern selection results in approx. $3 - 4\%$ better QoS performance compared to the local method. Like the global selection, the discarding selection shows a steeper increase of the QoS performance when compared to the two other heuristics.

Regarding the computation time, all three selection methods show similar results as in the previous campaign. The discarding selection requires for sequential structures almost the same efforts as the global method. For all parallel structures it performs almost three times as fast. It must be noted that after a 100% probability of generating parallel structures, no further improvements can be achieved. Therefore, the values for the 100% probability represent a maximum of the indi-

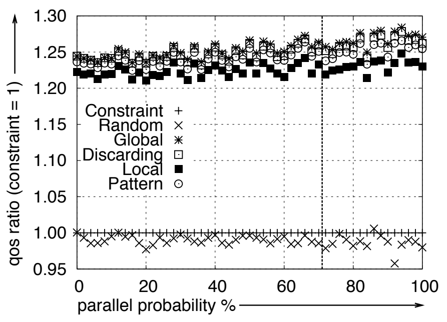


Figure 6.22: Relative QoS to constr. selection: optimisation-only methods (C5).

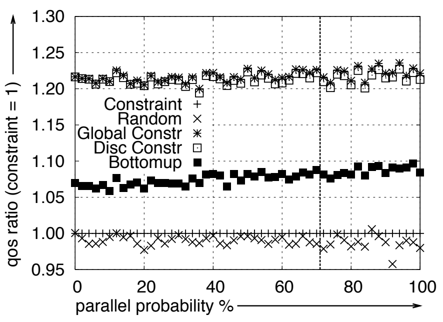


Figure 6.23: Relative QoS to constr. selection: constraint-aware methods (C5).

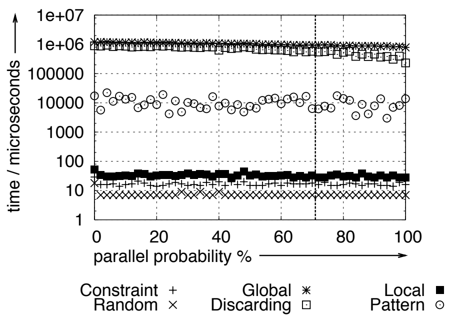


Figure 6.24: Computation duration: optimisation-only methods (C5).

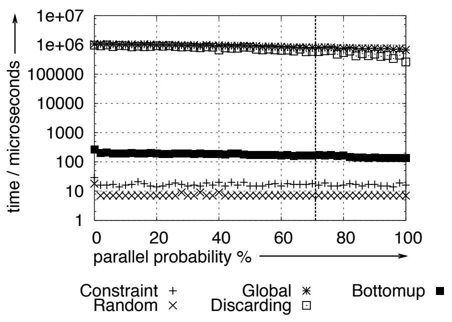


Figure 6.25: Computation duration: constraint-aware methods (C5).

cated direction. Like in the previous campaign, the pattern selection shows a larger volatility of its computational duration than the other selection methods.

Interpretation

The results show generally the larger potential for QoS optimisation if a composition structure contains parallel structures. Moreover, the three heuristics, the discarding, local and pattern selection, show also QoS performance improvements with a larger ratio of parallel structures. The constraint and the random selection show constant QoS performance. Therefore, such methods do not appear to perform well with parallel structures when compared to the other methods.

The computation time stays almost constant for all methods, except for the discarding selection, which shows poor time performance for sequential structures while its performance improves for a rising ratio of parallel structures. As in previ-

Method	**Average Mean $\overline{x}$**	**Standard Deviation s**	**% in 1s**	**% in 2s**	**% in 3s**	**95%-Conf.**
Overall QoS relative to constraint selection						
Constraint	1.0	0.0	0.0	0.0	0.0	$\pm$ 0.0
Random	0.9799	0.1005	67.50	96.00	99.50	$\pm$ 0.0118
Global O.	1.2705	0.1209	71.00	92.50	99.50	$\pm$ 0.0142
Global Co.	1.2212	0.1169	71.00	97.00	100.00	$\pm$ 0.0137
Disc. O.	1.2619	0.1261	71.00	93.00	99.00	$\pm$ 0.014
Disc. Co.	1.2127	0.1184	70.00	97.50	100.00	$\pm$ 0.013
Local	1.2300	0.1278	73.00	95.50	99.90	$\pm$ 0.015
Bottomup	1.0843	0.0868	88.50	95.00	97.50	$\pm$ 0.010
Pattern	1.2545	0.1225	71.00	92.00	99.50	$\pm$ 0.014
Computation duration, in microseconds						
Constraint	16	28	99.50	99.50	99.50	$\pm$ 3.296
Random	7	0.0	0.0	0.0	0.0	$\pm$ 0.0
Global O.	787,800	182,108	66.00	96.50	100.00	$\pm$ 21,438
Global Co.	675,092	170,954	65.50	97.50	100.00	$\pm$ 20,125
Disc. O.	232,417	348,351	85.50	94.00	98.00	$\pm$ 41,010
Disc. Co.	261,742	406,588	86.50	93.00	98.00	$\pm$ 47,866
Local	28	2	64.50	95.50	98.50	$\pm$ 0.235
Bottomup	137	70	96.50	97.00	97.00	$\pm$ 8.240
Pattern	13,902	1,407,168	95.50	98.50	98.50	7,864

Table 6.8: QoS and times, setup with a parallel probability of 100% in C5.

ous campaigns, the pattern selection shows a highly oscillating computation time, indicating a strong dependency on the composition structure.

6.5 Evaluation Conclusions

The first three campaigns have tested the heuristics with a rising number of tasks and candidates. As discussed in the previous chapter, the global selection results in quickly rising computation times. These results confirm that the straight-forward evaluation of all candidates represents an intolerable approach. The campaigns also show that an increasing number of candidates is not preferable to an increasing number of tasks for the global selection under the simulation conditions. Although an increasing number of candidates results in efforts that can be bounded by a polynomial, the results show that only small compositions can be solved in feasible time by such brute-force approaches.

The campaign that increases the deviation of the QoS values demonstrates that the more the QoS values differ among the candidates, the more increases the potential to improve the overall QoS. This potential is identified by the global, discarding, bottomup and pattern methods when compared to the random selection, which ignores the QoS. However, the heuristic local selection shows a decreasing performance which has indicated that this method performs worse the stronger the

QoS values among the candidates vary. In contrast, the heuristics reveal a good performance close to the optimal solution in cases where the QoS of candidates show relatively low differences.

The campaign that tests for the performance differences when either processing parallel or sequential structures demonstrates that, with a rising ratio of parallel structures, the optimal QoS identified by the global selection increases. Contrary to the preceding campaign, all the heuristics show a slight increase of the QoS performance when the ratio of parallel structures rises. Accordingly, the structural arrangement of the composition has in impact on the QoS performance of the different methods. However, all methods show a similar level of improvement and no method reveals a particular weakness or strength in this campaign.

Referring to the four evaluated heuristics, the following conclusions can be drawn:

- **Discarding.** The discarding selection method can be considered unfeasible. It performed about two to three times faster in comparison to the global method, however it still showed an exponentially increasing effort for the given problem instances. Obviously, the given setup of QoS values and composition structures do not allow the algorithm to bound sub-trees very often. Regarding the QoS performance, the discarding selection resulted in the same QoS as the global selection in the tests.

- **Local.** The local heuristic represents a simple approach that consumes almost no computational efforts. The required efforts are in the same range as the constraint or the random selection. Considering that an assignment must be performed in any case, the efforts for the local selection are negligible. The campaigns have shown that this method usually resulted in more than $4/5$ of the full QoS optimisation potential when considering the interval between the optimal solution and the QoS-ignoring result. Further tests have shown that this method becomes noticeably weaker the more the QoS values differ among of the candidates.

- **Bottomup.** The bottomup selection has shown a negligible computational effort. This method scales well with both a rising number of candidates and tasks. However, the QoS performance can be regarded as weak when compared to the local or pattern selection. In the campaigns, it reached about $1/2$ to $2/3$ of the QoS optimisation potential, referring to the interval between optimal and QoS-ignoring solution. Moreover, its QoS performance decreases for a rising number of tasks. However, this approach is capable of considering constraints. Thus, it has an advantage over the local and pattern selection.

- **Pattern.** Compared with the bottomup and the local methods, the pattern selection showed the best QoS performance. However, a small fraction of the performed runs has resulted in an unfeasible computational effort that makes the application of this method difficult. As discussed in the previous chapter, the algorithm performs badly if a structural pattern contains a large number of tasks. Although this constellation can be regarded as unlikely, the random setups have encountered these at significant amount of times. As a conclusion for future work, the pattern method could be combined with a local approach: Then, a combined algorithm can switch to a faster algorithm if a composition structure is identified to produce high efforts, e.g. a single pattern that contains a relatively high number of tasks.

In summary, the campaigns have shown that the heuristic have a good performance under special conditions. As for other cases, the following basic trade-off couples apply: If an heuristic performs quickly, its QoS performance is poor. If an algorithm shows a good QoS performance, it shows an unfeasible worst-case effort. As an inspiration for future work, a hybrid heuristic might be able to combine the good elements of the tested approaches. For example, such an heuristic could choose an appropriate strategy based on key parameters determined for a particular problem instance.

Chapter 7

Developing Service Compositions

As Chapter 2 has explained, different modelling languages exist: Some of them focus on modelling business processes and ignore specific SOA technologies, while others are specifically designed to cover specific technical platforms, for example Web services. In addition, Chapter 2 has defined a relation between business process models, composition models and a service composition. The development process of a service composition must cover the entire chain – from processing a business process model as an input, up to the output of an execution description that allows the service provider to run the composition.

Since the development of composition starts with a model of a process, the literature discusses methods and solutions that take advantage of the proposed *Model Driven Architecture* (MDA) introduced by the Object Management Group (OMG). The OMG represents a non-profit standardisation organisation with the goal of developing object-oriented technologies. Briefly explained, the MDA proposal bundles a set of standards to facilitate the software development with the focus on modelling languages and modelling tools. In an MDA-based development effort, not writing source code but expressing models represent the centre development effort. It is desired that the source code is automatically generated from a set of models that cover the structure and the behaviour of the software. In the context of service compositions, the MDA provides an infrastructure that allows directly taking a model of the business process as the starting point for the development of a composition of services that facilitate the process. Many software vendors and initiatives support this approach with available integrated development environments (IDEs). Examples are the software development tools of Borland, as well as EclipseUML, which is based on the Eclipse open source project; IBM offers a suite named WebSphere portfolio that supports the entire chain of development from business process modelling to a service with monitoring facilities [85].[1]

This chapter will introduce the basic elements that such a development process

[1]Borland provides the Together Suite which covers general software development efforts including the composition of Web services. A development environment that has a direct focus on the realisation of business processes from IBM is the WebSphere Integration Developer.

must involve; however, it will not cover and discuss specific development process methodologies such as the popular waterfall methodology or newer use-case-driven or extreme programming approaches. It will focus on the issues of trading when developing compositions. Because the MDA is not a part of this focus area, this chapter tries to avoid details about the MDA. It will not present a complete view on the MDA and concepts like the model hierarchy, for example, are ignored in this discussion.

The goal is to give the reader an orientation about the development of compositions and to explain how processing QoS will enhance the output of the development process. The remainder of this chapter will start with a brief introduction to the MDA and introduces service-composition-related applications from the literature. Subsequent to that, the main steps in the development of compositions will be discussed; this work is partially based on previously published research work about building compositions of Web services (cf. Grønmo and Jaeger [39], [40]).

7.1 Introduction to the Model Driven Architecture

The general objective of the MDA is to separate the specification of the system functionality from specific platform technology [96]. Figure 7.1 shows the basic separation of different categories for models that occur in this architecture. The separation is the result of an abstraction that separates the technical details in a model from the functional aspects of a system. Accordingly, one kind of model is proposed that does not refer to any technical specific artefacts. Such a model is named Platform Independent Model (PIM). Based on the PIM, a Platform Specific Model (PSM) is foreseen. Besides a PIM and a PSM, the development can optionally involve a model of the domain concepts that describe the actors and entities found in the application domain. Such a model can provide a view on the system that is independent from any functional or computational details.

The MDA approach anticipates that developers start to design their software with a PIM that does not cover any technology-specific aspects. When this model is finished, it represents the basis for deriving a PSM. Therefore, the choice for a specific technology is deferred until the PSM is created. The authors of the MDA anticipate that these transformations happen automatically and thus let the developer focus on modelling the functionality rather than writing code. Consequently, the model becomes an asset of the development process. With the adoption of the MDA, three main advantages are anticipated by the OMG [96]:

- The same system functionality, once modelled, can be reused for different target platforms. This flexibility is often required, because the IT landscape in businesses consists of many heterogeneous platforms and software systems as already explained in Section 2.4.

- A separate model of the system functionality allows a more efficient validation. If the model is free from technical details, there is less to consider for validation tools. Moreover, the software specification and development will likely result in different models at different levels of abstraction. With defined transformations between the different levels of abstraction, tools can ensure the consistency between these models.

- The creation of the system can be guaranteed to be free from technical artefacts that a specific technology would imply. Thus, a platform-independent model offers a better foundation for the desired functionality.

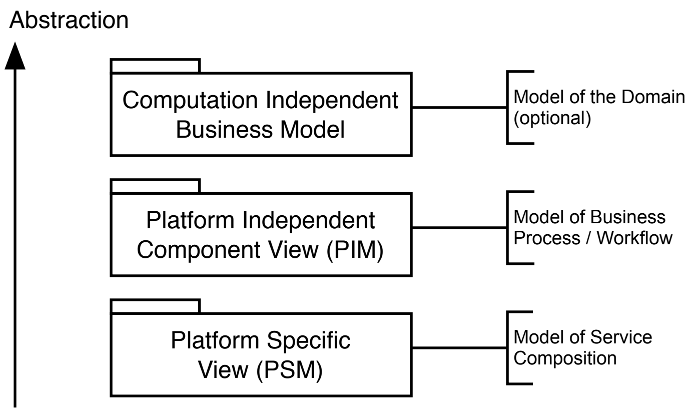

Abstraction
Computation Independent Business Model
Model of the Domain (optional)
Platform Independent Component View (PIM)
Model of Business Process / Workflow
Platform Specific View (PSM)
Model of Service Composition

Figure 7.1: Separation of the models in the MDA (based on OMG's MDA document [96, section 2.3]).

Besides the MDA proposal, the OMG has also standardised several related technologies. The Meta Object Facility (MOF, [98]) represents the most fundamental standard. It defines how to specify modelling languages. Usually, a software developer is not concerned with the MOF. Instead, it is a standard used by modelling language designers and tool manufacturers. A well-known modelling language is the Unified Modelling Language (UML, [100]). Software developers use the UML to express models of software systems or parts of a software system. And as mentioned in Section 2.1.2, a part of the UML, the activity diagram, is also considered for modelling business processes.

The main research challenge in the field of the MDA is the model transformation technology that transforms PIMs into PSMs. Furthermore transformation should provide the generation of source code or execution descriptions from PSMs. Source code or similar descriptions can be seen as textual models in this context. Several software products and tools using the UML and the MDA are already available for the creation of business processes. However, the goal that has not been reached so far is a fully automated tool chain. The vision is that software engineers can focus on the business process model and automated transformations provide the generation of the source code.

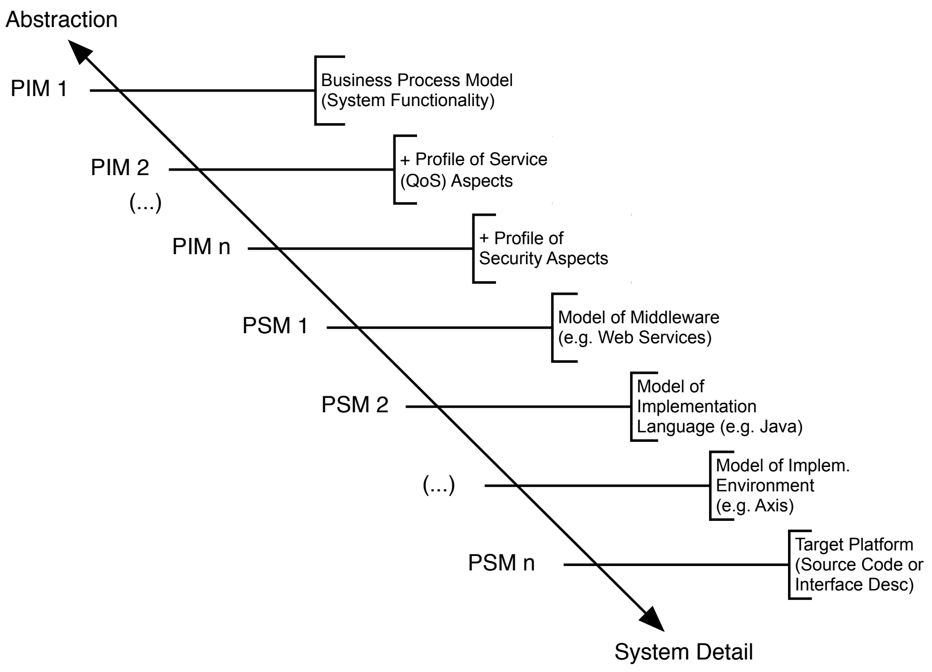

Abstraction
PIM 1
Business Process Model (System Functionality)
PIM 2
+ Profile of Service (QoS) Aspects
(...)
PIM n
+ Profile of Security Aspects
PSM 1
Model of Middleware (e.g. Web Services)
PSM 2
Model of Implementation Language (e.g. Java)
(...)
Model of Implem. Environment (e.g. Axis)
PSM n
Target Platform (Source Code or Interface Desc)
System Detail

Figure 7.2: The evolution of the models in the MDA (based on Bézivin et al. [9]).

In the context of MDA, a problem is that a single modelling language does not cover all the aspects needed to create models of the entire problem domain. This applies especially to the topic of this work, the creation of business processes with service compositions. Chapter 2 has already introduced several modelling languages, which are at different levels of abstraction. Some languages focus on modelling abstract business processes and some languages involve Web service standards which refers to PIMs. An ideal MDA environment would support the transformation of models between different levels of abstraction and it would ensure the consistency between them. Besides the transformation issues, several further aspects should be covered like security, QoS or organisational constraints. This results in different views on the system. To cover this problem, current research efforts evaluate how to mark up models with aspects based on extensions, named UML profiles. UML profiles allow extending the modelling concepts found in the UML.

Figure 7.2 outlines different levels of abstraction, with possible examples of PIMs and PSMs. This figure has been presented by Bézivin et al. [9]. It shows that a model of the business process is assigned to the group of PIMs. Additions to this model, such as the specification of QoS aspects or security constraints, result in a less abstract PIM. A model that directly refers to a specific component or SOA technology represents a PSM. A further specialisation, in the sense of being less abstract, is the coverage of specific SOA platforms. Then, possible transformations result in further detailed models up to the source code or interface descriptions.

7.1.1 Model Driven Development of Web Service Compositions

The adoption of the MDA to develop service compositions or business processes has been already covered by research work. In the following, three research efforts are introduced, which have applied the MDA for the creation of business processes with SOA technologies. These three efforts are considered to be representative for this field and to reflect the state-of-the-art. The first cited work covers the aspects of supporting Web services in an MDA-based development. The second effort plays in the same field but focusses more on how the transformations will be created. The third work covers the consistency between the models of different levels of abstraction. These three research works also demonstrate the feasibility of adopting the MDA to develop service compositions in order to develop business processes.

- **Web service compositions in UML** (Skogan et al. [120]) The authors propose UML activity models to design a Web service composition. Using MDA techniques, an UML activity diagram can be transformed into different Web service composition languages. Based on the activity models, the generation of executable models using the composition languages BPEL4WS and WorkSCo is described. WorkSCo is part of a framework for realising workflows with services.

  UML activity diagrams are used as platform independent composition models. The contribution of the authors lies in the special coverage of the Web services technology. It proposes the integration of WSDL descriptions into an UML diagram in order to generate the interface description for the resulting composition, which is also seen as a new service. Most other approaches presume that the WSDL file can or must be – depending on the application scenario – generated by an UML class diagram in order to complete the composition model. Generally, the authors propose to integrate existing Web services. Thus their WSDL descriptions can be reused.

- **B2B Applications in the context of MDA** (Bézivin et al. [9]). The work of Bézivin et al. covers the transformation of UML models into two target platforms: One is the Microsoft .NET framework and the other is a BPEL4WS execution environment for Web service compositions. As an intermediate step for the generation of BPEL4WS descriptions, the output of WSDL is provided. Further contributions covers the transformation originating from different PIMs to a Java-based Web service environment without the intermediate step of a composition model in BPEL4WS [8]. The research group of Bézivin et al. have put a special emphasis on the development of a dedicated transformation language that they name Atlas Transformation Language (ATL, [7]). The language allows definitions for transformation rules between the abstract elements of a PIM and the abstract elements of a PSM.

- **The medini platform** (Kath et al. [70]). The foundation of this research effort is the modelling environment named *medini*. Medini provides a set of tools to establish a tool chain for MDA-based software development. In addition, it features a model repository that stores the software models, which are seen as an asset of the development process. The tools provide verification functionality to ensure the consistency of the models at different levels of abstraction. The medini architecture consists of three tiers: the modelling tools, the model repositories and the transformation tools. To ensure interoperability with other development software, the elements of the medini platform are designed to conform to the MOF standards for open interfaces and model repositories.

  One application of the medini platform is the transformation of EDOC models to PSMs that target a CORBA middleware environment. EDOC stands for Enterprise Distributed Object Computing and represents a profile for the UML that covers the modelling of distributed enterprise systems. A transformation tool is capable of performing a transformation from EDOC models to PSMs suited for the CORBA Component Model (CCM, [97]). Another transformation tool covers the generation of interface descriptions of Web services. In addition to the service interface descriptions, medini also provides support for behaviour models that resemble a process. The behaviour models are EDOC choreographies on the PIM-side and BPEL4WS descriptions on the PSM-side.

7.2 Model-Driven Development of Service Compositions

The previous sections have explained that the MDA proposal by the OMG and the related research work provide methods and technologies that cover the specific characteristics of developing service compositions. The forthcoming sections will outline at what point in a model-driven development effort the trading of services, especially the QoS-based selection will take place. The trading identifies services in order to integrate them into the composition. As briefly explained in Section 1.1, it covers two main parts: One part represents the functional matchmaking of available candidate services and the other facilitates a selection among potential service candidates in order to optimise the resulting QoS of the composition.

However, this discussion does not cover the roles, the behaviour and the collaboration of actors who might participate in such a development process. Thus, this discussion cannot serve as a development methodology. The goal in this discussion is to identify at which points the trading and especially the processing of QoS enhances the output of intermediate steps in the development process. Moreover, the upcoming sections will also explain which facilities are required to perform the trading and to process QoS information. As explained in the introduction, it is assumed that the retailer of the composition also represents the modeller and the developer of the composition. As introduced in the Section 1.1, he acts also as the service importer when contacting other service brokers and 3rd party service providers.

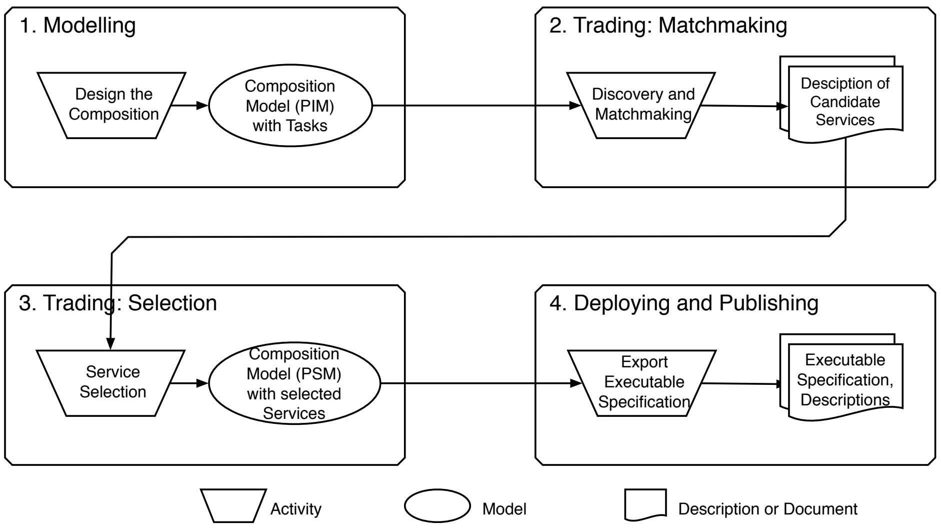

1. Modelling
Design the Composition
Composition Model (PIM) with Tasks
2. Trading: Matchmaking
Discovery and Matchmaking
Desciption of Candidate Services
3. Trading: Selection
Service Selection
Composition Model (PSM) with selected Services
4. Deploying and Publishing
Export Executable Specification
Executable Specification, Descriptions
Activity
Model
Description or Document

Figure 7.3: Main steps of developing service compositions.

Figure 7.3 outlines the tasks of an abstract development process that consists of four main steps. This figure and all the following use elements related to the flow chart notation: The trapezoid-shaped boxes indicate an activity. Activities can directly follow each other or have documents (descriptions) or models as an output or input. Documents are represented by a rectangular box that has a wave-shaped lower bound, and models are represented by ovals. Relations between subsequent activities, as well as input and output directions, are shown with arrows.

The first step covers the modelling of an abstract composition consisting of tasks. It can be left open whether this model is part of the development or represents an externally provided process model. Besides, it must be noted that the flow of information will not be elaborated in this work because this represents the main contribution of the research works presented in Section 7.1.1 and in Section 2.4.1. Very briefly, the particular steps are as follows:

- **1. Modelling.** As a first step, either the retailer or any other party defines a new service composition by creating a PIM that defines the control flow between the tasks. Then, the retailer annotates the tasks with descriptions to identify the functional requirements. In addition to the functional description, the retailer determines the required QoS for each of the identified tasks and for the entire composition. Adding the functional description and defining the QoS requirements can be done in parallel since they do not depend on each other. The outcome of the first step is an abstract composition model (a PIM) that contains all the required information for the discovery and selection of services.

- **2. Trading: matchmaking of functionality.** The second step covers the first part of the trading, which is the discovery of suitable services. This part of trading is usually based on matchmaking between the functional descriptions of services and the functional requirements. MDA techniques transformed the interface descriptions from the model and the task annotations into a textual description. This description can serve as an input for an automated search and matchmaking process evaluating services offered by other retailers or service brokers. The outcome of this step is a list of candidate services for each task.

- **3. Trading: QoS-based selection.** In this step, the set of candidate services undergo a selection, which is based on the QoS requirements. This represents the second part of trading. QoS requirements can be used in two ways: Either requirements can represent a global constraint that the resulting composition must meet or QoS categories can be used as optimisation criteria. The QoS-based selection will result in a ranked list of candidates for each task. Then, the retailer can choose a concrete service for each task and adds the corresponding description to a new software model. The resulting model is a PSM. Depending on the anticipated level of detail, this description can include interface descriptions, functional descriptions as well as QoS statements.

- **4. Assembly, deployment and publishing.** In the fourth and last step, the PSM is used to generate different descriptions: a) a document describing the interface to advertise the composition for future trading processes, b) an executable flow description to deploy the composition in an execution environment and c) a description defining the offered QoS of the entire composition.

7.2.1 Modelling the Composition

The goal of the first step is to provide the necessary information for the subsequent trading processes. This means that the modeller defines the requirements resulting from the needed tasks. Based on these requirements, services must be found that fulfil them. A detailed model of the first step is given in Figure 7.4. The development starts with an abstract model of the business process. Chapter 2 has already discussed possible languages for this purpose. Depending on the organisational circumstances, either the modeller of the composition performs the modelling or an existing business process model is taken as the basis of the composition model. In the Web service domain, the case is possible that a modeller receives a model expressed in the EPC notation of a business process. Then, based on the EPC description, an UML activity diagram must be created to continue the development in the MDA environment. Depending on the available facilities, the modeller can take advantage of existing transformations. Such transformations convert the existing model into a new one using a modelling language of the development environment (cf. Kath et al. [70]).

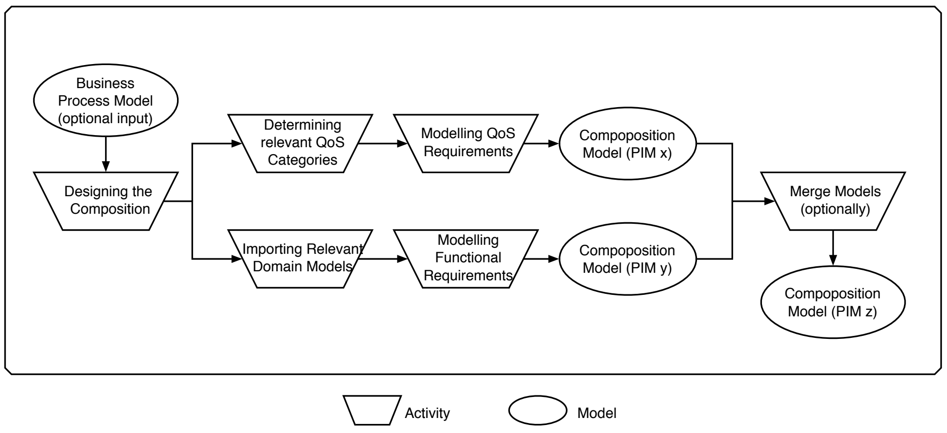

Business Process Model (optional input)
Designing the Composition
Determining relevant QoS Categories
Modelling QoS Requirements
Compoposition Model (PIM x)
Importing Relevant Domain Models
Modelling Functional Requirements
Compoposition Model (PIM y)
Merge Models (optionally)
Compoposition Model (PIM z)
Activity
Model

Figure 7.4: First step: modelling the composition.

To deliver the requirement descriptions for the trading, the existing model must be enhanced. Depending on the kind of requirements, this can cover the functional descriptions of the tasks or statements about the required QoS. The functional description can include a description of the interface or a description of the behaviour of the service. The state-of-the-art focuses on the interface description where two main approaches exists: In one approach, the interface is described on a syntactic level and in the other description of the semantics of the service elements can also be used for this purpose. Examples for a syntactic description are interface descriptions using the WSDL for Web services or the Interface Definition Language (IDL) in a CORBA environment.

The general problem with interface descriptions is that labels and data types used do not provide an unambiguous definition of the interpretation of these elements. To make such implicit information explicit, the common approach is to provide a semantic description of the service. A semantic description means that the described elements are linked to concepts that have a defined interpretation. Such a set of interpretations is named ontology; this ontology is subsumed by the concept of the domain model on top of the PIM in a MDA as presented in Section 7.1. Then, adding such a description provides a more precise definition of the task or service. This represents the main motivation for expressing the desired functionality with a semantic description.

To create a semantic description of the interfaces, the designer must first identify appropriate ontologies (domain models) that provide a definition of the concepts required to describe the service semantically. In the domain of Web services, such domain models are usually represented in textual descriptions using ontology languages, e.g. the Web Ontology Language (OWL, [89]). In the case that the MDA environment does not directly support the used language of the domain model, automated transformations can help to convert these into the required representation. In order to apply this technique to develop compositions of Web services, Djuric has the corresponding transformation methods for importing OWL ontologies into UML models [24].

A basic solution for the semantic description of the behaviour is the reference to a service type. Such a service type can be referenced to a standardised taxonomies of services. As an example, the United Nations Standard Products and Services Code (UNSPSC, [110]) provides such a directory. Using this directory, the modeller describes a service with an unambiguous key that refers to an abstract service functionality. More sophisticated approaches aim at describing conditions that must hold before, during, and after the invocation of the service. Sycara et al. have discussed the matchmaking of services involving statements about conditions [125]. Other research efforts have discussed the required interaction when invoking interface. The approach is to model patterns of interaction with process algebra or state machines. Then, a matchmaking between the patterns of offered services and the patterns of requested services can identify a compatibility on the level of possible interactions (cf. Bordeaux et al. [82], and Wombacher et al. [152]).

Besides domain models and the modelling facilities an MDA environment might provide, some dedicated proposals exist for the semantic descriptions. The currently most popular are OWL Services (OWL-S, [130]), the Web Service Modelling Ontology (WSMO, [32]) and the semantic extension to Web service interface definitions (WSDL-S, [119]). All these proposals involve Web service standards as this represents the currently most popular implementation of an SOA. In the recent past, a unification effort has started and an interest group at the W3C discusses the three proposals.[2] The advantage for the modeller of the composition is that these semantic description languages cover the specific characteristics of a service. The UML for example, is intended for the general design of software systems and thus, less suitable for modelling the semantics of a service. Moreover, research work about existing trading facilities, which will be introduced in the next section, is also based on these three proposals. However, using these languages results in additional effort regarding their integration in an MDA environment. Either transformations must exist to import and export the dedicated semantic description of services [41] or specific modelling tools must be integrated into the MDA environment.

In addition to the interface description, the modeller must determine the requirements on the QoS. The process begins with the selection of the relevant QoS

[2]This is the *Semantic Web Services Interest Group* hosted by the W3C. This group has the primary goal to share findings and activities about semantic description of Web services [12].

characteristics. This depends on the nature of the given requirements as well as which characteristics are supported by trading facilities and the service providers. As Chapter 3 has explained, the modeller can query other retailers and brokers in order to identify the characteristics that can be used to specify the requirements. An additional problem is the definition of the measures; one approach is to refer to a commonly agreed definition of the relevant QoS characteristics, for example as published by standard bodies. This particular issue has been discussed in Chapter 3 and was also mentioned as one of the prerequisites to perform the aggregation in the beginning of Chapter 4.

Currently, there is no de-facto standard present for representing QoS characteristics in the Web service domain as well as in other component middleware. When using UML in an MDA-based environment, the representation of QoS requirements in the model can follow the UML QoS-Profile [99]. This UML profile defines a set of QoS categories with their interpretation and their notation. It allows the modeller to express QoS statements as annotations in UML models. Since UML is a graphical language, the UML QoS-Profile represents a preferred choice for use in a MDA environment (cf. Grønmo and Jaeger [39]).

Figure 7.5 summarises the considered aspects of service description in a simple hierarchy. The basic separation identifies a functional and a non-functional part. In the field of non-functional descriptions, this hierarchy distinguishes between QoS and non-QoS. For non-QoS characteristics different definitions exist; for this work, it is assumed that they do not represent numeric measures, such as the country from where the service is provided (cf. the beginning of Chapter 3). The functional part has two main sub-parts, one covers the behaviour and another the interface.

The outcome is a composition model that contains all the needed semantic and syntactic description of the interface for performing service discovery. Thus, it provides the necessary information for the next step. An additional outcome is a model that covers also QoS requirements. This represents the required input for the third step in the development, the selection of services. Optionally, these two aspects can merge into one model. However, this represents a methodological consideration that is not the focus of this discussion. Besides, it is open whether the requirements are expressed in a graphical model or in a textual description. Since the MDA anticipates the use of transformations to convert between both, it is presumed that the graphical representation provided by an MDA environment represents the preferred interface for the composition modeller.

7.2.2 Trading: Matchmaking

The second step covers the discovery of services. The discovery of services is a trading process that preceeds the selection of services. The discovery has the goal to identify services that meet the functional demands. The idea is to deliver a set of services that would generally fulfil the desired task regardless of any non-functional preference. The identification process is based on matchmaking descriptions of interface and service functionality. To perform the matchmaking, it is assumed that a

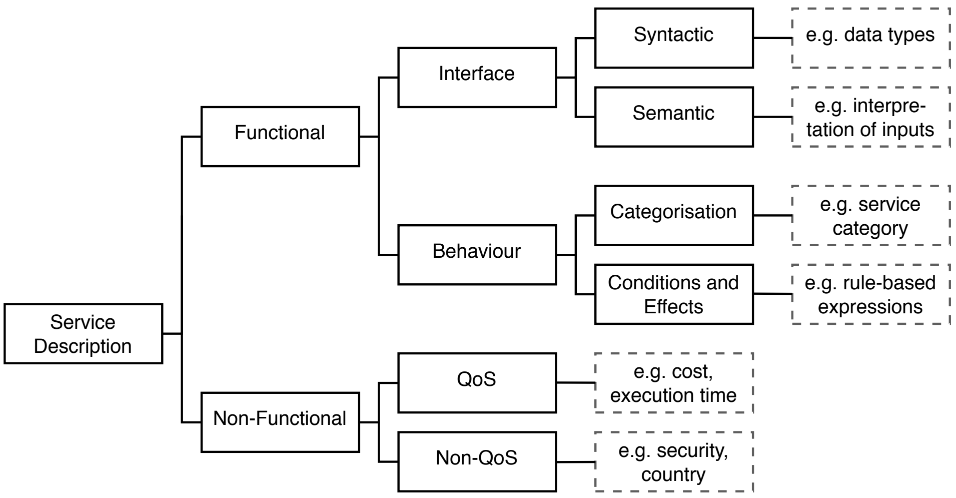


Figure 7.5: Proposed taxonomy of service descriptions.

service broker has access to the syntactic and optionally to the semantic description of the service and its elements. The necessary description can be derived from the model that the previous step has produced. Grønmo et al. have presented transformations from composition models into a semantic description language for Web services [41]. The authors explain how semantic service descriptions modelled in an UML model can be extracted to generate a description document compliant to OWL-S in order to discover Web services.

The generated documents represent semantic descriptions of requested tasks for which candidate services must be identified. The subsequent discovery process involves querying brokers to identify these candidates. For this step, syntactic and semantic interface descriptions are used. A matchmaking process compares the interface signature between the required tasks and available candidates. As mentioned in the previous section, such a description does not sufficiently describe the functionality. In any case, matchmaking must consider the syntactic (interface) description. If a service does not match the required interface on the syntactic level, it would be regarded as incompatible anyway. Thus, the matchmaking of the semantic description represents always an addition to the matchmaking of the syntactic description.

The goal of the semantic matchmaking is to identify the relation between a requested and an offered entity, which are described by concepts defined in an ontology. A matchmaking process can identify three main relations between the concept representing the required entity and the concept representing the offered entity (cf. Jaeger et al. [65, 66]):

- **No Match.** The matchmaking process cannot identify any relation between both concepts. This can have many reasons. Most likely the two concepts are indeed different. Another problem that is inherent to the matching is that the ontology used by one semantic description does show any relation to the ontology used by the other. In this case, the two concepts could match in reality. However, a match cannot be determined by the software.

- **One Concept Subsumes the Other.** If one concept subsumes the other, the matchmaking process has identified that one concept denotes a more general entity than the other does. This is similar to what can be found in object hierarchies of object oriented programming languages: From two classes, one can represent a super-class of the other representing a more general type.

  Since most services are implemented by using object-oriented programming languages, previous research work has explained that the general subsumption relation must consider the direction of the subsumption relation between service importer and exporter [65]: For the syntactic convention in programming languages, the contravariance must hold. In common object-oriented programming languages, the sub-class relation represents a contravariance because a subclass provides all characteristics of its super-class plus additional one.

  Ensuring the contravariance means that an interface is defined to expect a type that shows all needed characteristics for the operation, even if the service importer invokes the service with a more special type. This direction is applied vice versa to the returned types: The service exporter can return a type that shows the same characteristics as required by the importer requires or a more special one. This ensures that the returned type provides all required characteristics and does not violate any requirements. Other research work has also discussed that with reverse subsumption relations, a matched service could still be used. In this case, the partially matched services can be used with additional services that together ensure that the given functional requirements are met (cf. the plug-in case described by Li and Horrocks [81]).

- **Match.** The matchmaking process identifies that the concept representing the offered entity is equivalent with the concept representing the required entity. Of course, this represents the desired result of a matchmaking process.

In the field of Web services, available proposals cover the mentioned description languages OWL-S, WSMO and WSDL-S. For the OWL-S language, several works have introduced matchmaking algorithms [66, 65, 81, 106], as well as extensions to trading infrastructure namely the UDDI discovery services [1, 103, 124]. Mainly these works focus on the semantic description of the parameter types and matching service categories (i.e. WSDL-S), while the semantic description of the service behaviour is partially covered by expressing conditions based on rule expressions (in OWL-S and WSMO).

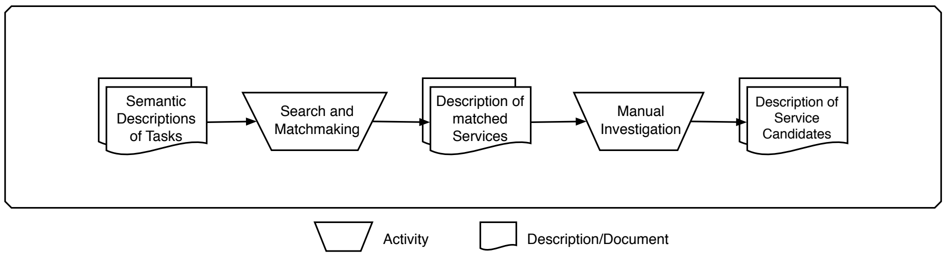

Semantic Descriptions of Tasks
Search and Matchmaking
Description of matched Services
Manual Investigation
Description of Service Candidates
Activity
Description/Document

Figure 7.6: Second step: matchmaking.

However, if a matchmaking algorithm identifies clear matches for all the aspects, a seamless interoperation of the involved candidates cannot be guaranteed. To cope with this issue, further reasoning is necessary. The matchmaking of syntactic and semantic interface descriptions will improve the precision of the discovery, but it will not remove the need for manual investigation of the discovered services to assure the compatibility. Otherwise, the accurate interoperation of the service in the composition cannot be guaranteed. Figure 7.6 summarises this step: Based on the semantic descriptions of the required tasks, a matchmaking process tries to identify service candidates. The state of the research shows that these service candidates must undergo an additional review in order to ensure their functional suitability. The anticipated outcome of this step is at least one service candidate per task.

7.2.3 Trading: Quality-of-Service-based Selection of Candidates

In this step, the selection of the services takes place; the goal of this step is to identify the best selection among the candidate services for the tasks of the composition with respect to selection criteria. This work considers the QoS as selection criteria. Because this step represents the main topic of this work, it was covered in the previous chapters. This section will just discuss the basic activities of this development step, as outlined in Figure 7.7.

Based on the identified set of candidates from the previous step, the QoS of the candidates must be processed. Either the QoS of each candidate is part of the available descriptions, or the QoS must be retrieved from the service providers. Depending on what has been defined in the first step, the QoS requirements can deal with two aspects which were also parts of the problem model for the QoS-based selection: constraints and optimisation goals. A constraint requirement can, for example, cover statements like the maximum response time of the entire composition. The other aspect covers optimisation criteria. An example is to reduce the amount of resources for executing the composition.

The selection process determines the best possible selection of services depending on the requirements and the offered QoS of the considered candidates. The simplest approach that was discussed in this thesis for performing the selection

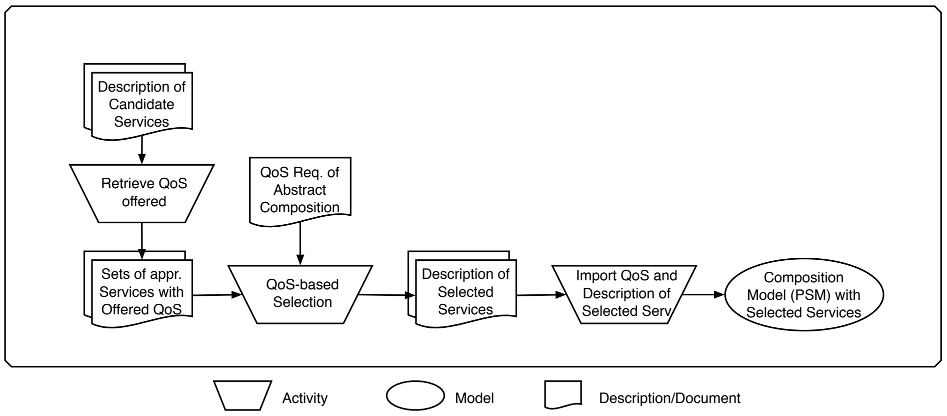

Description of Candidate Services
Retrieve QoS offered
QoS Req. of Abstract Composition
Sets of appr. Services with Offered QoS
QoS-based Selection
Description of Selected Services
Import QoS and Description of Selected Serv
Composition Model (PSM) with Selected Services
Activity
Model
Description/Document

Figure 7.7: Third step: QoS-based selection of candidates.

operates on a local perspective: Such an algorithm selects a candidate by comparing the QoS among all candidates for a particular task and chooses the candidate that provides the best QoS. This case represents the state-of-the-art of available broker implementations and trading functionality. The previous chapters have also explained that a better QoS is obtained with an algorithm that considers the entire composition or sub-parts of the composition, rather than optimising the QoS for each task in isolation. Thus, the QoS-based selection takes place at this stage in the development process.

After the QoS-based selection has been performed, the modeller can add the QoS of the selected services to form a new version of the model – a platform-specific model of the composition. Based on the QoS of the individual services, an algorithm can compute a resulting QoS statement for the composition by using the aggregation method that was introduced in Chapter 4. The selection process might not only choose the optimal service among the candidates, it can also determine the ranking of the available candidates based on the relevant criteria. Depending on the return policies, the process can result in more than one ranked service for each task. Choosing more than one service for each task could increase the reliability of service executions by providing a redundant arrangement (cf. Jaeger and Ladner [58, 59]). The outcome of this step is the generated platform-specific model involving the chosen services. This model can be enhanced by importing the offered QoS values of each chosen service and the aggregated QoS for the resulting composition.

7.2.4 Advertisement and Deployment

The fourth step builds upon the output of the previous phases, which are the assignment of at least one service to the tasks, their semantic description and the QoS resulting from the assignment. The modeller can import all information for

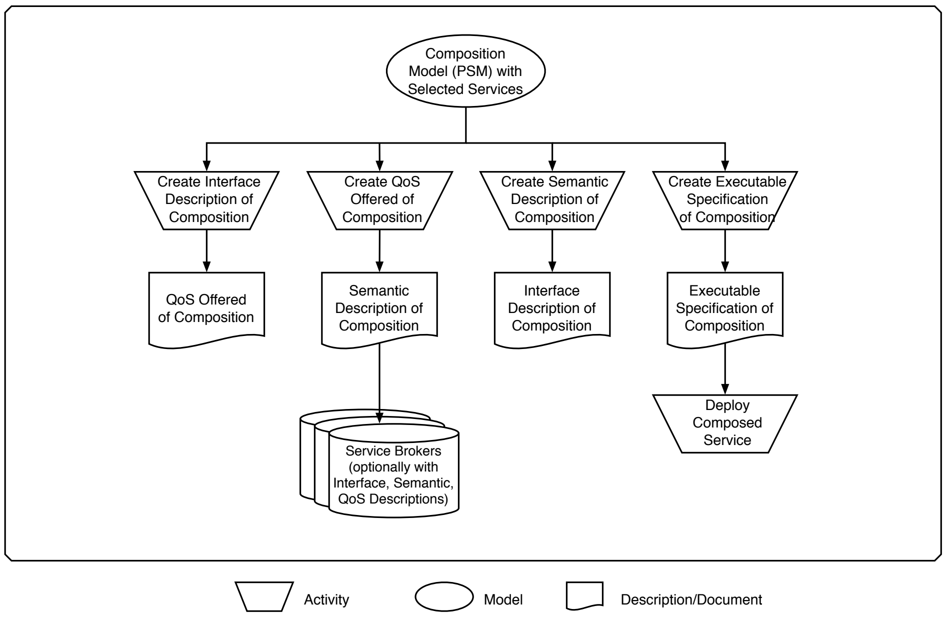

Composition Model (PSM) with Selected Services
Create Interface Description of Composition
Create QoS Offered of Composition
Create Semantic Description of Composition
Create Executable Specification of Composition
QoS Offered of Composition
Semantic Description of Composition
Interface Description of Composition
Executable Specification of Composition
Deploy Composed Service
Service Brokers (optionally with Interface, Semantic, QoS Descriptions)
Activity
Model
Description/Document

Figure 7.8: Fourth step: advertisement and deployment.

the particular models onto one unifying model by using software transformations. This step has the goal to create the necessary information to advertise the composition as a new service. Another goal is to generate an executable specification in order to perform the composition.

Figure 7.8 outlines the general activities of this step: Based on the existing PSM and optional additional descriptions, four main descriptions are created. Since they do not depend on each other, these activities can be performed in arbitrary order or in parallel. These descriptions are the following:

- **Interface Description.** Based on the design of the composition, the overall input and output parameters can be defined and exported. If the interface description uses a standardised description format, the composition can be advertised and executed as a service for the consumers. In the field of Web services this would also cover a WSDL description. An example for the MDA-based generation of WSDL files from service compositions has been introduced by Grønmo et al. [42].

- **QoS Description.** The previous step has already mentioned that, based on the QoS of the involved services, the QoS of the overall composition can be aggregated. Such a description can use the same languages which the 3rd party providers use for their service advertisements.

- **Semantic Description.** A semantic description of the service interface can be derived from the semantic descriptions of each task as created in the first step of the development. The part that covers the behaviour of the service must involve additional modelling because the composition represents a functionality that was not available before. Thus, this part cannot be derived from the description of the involved services.

Regarding the semantic description of the interfaces, Grønmo et al. have already discussed an approach based on transformations between UML and OWL Services [41].

- **Executable Specification.** Automated transformations can create an executable specification from the control specification expressed in the PSM. Runtime environments can use this executable specification to invoke the available services in the defined order. Such a specification could use WS-BPEL or BEPL4WS in order to define compositions of Web services. Kath et al. [70] and Skogan et al. [120] have explained the generation of BPEL4WS descriptions to execute compositions of Web services. Examples of runtime environments that are capable of BPEL4WS are the IBM WebSphere Server [85] or the Oracle BPEL Server [117].

Finally, the composed service composition can be deployed and published. The generated descriptions can be passed to service brokers in order to advertise the service. At the end, the retailer can offer the service ready for the invocation by consumers.

7.2.5 Development as an Iterative Process

Like other design and creation processes in the engineering and computer sciences, the creation of the composition is an iterative and heuristic process that assumes the ability to go back and forth to the different activities. The heuristic nature is as such that by undergoing these procedures in iterations, the result becomes better with each iteration. Different reasons exists for why the development process requires an iterative procedure, among them are the following:

- **No matched services.** It can happen that the matchmaking process does not identify suitable service candidates. In this case, the retailer has two options: The development process can be interrupted and the required service can be implemented. Alternatively, he can modify the composition design in order to adapt the required services to the available services that were previously excluded. Then, the modeller must redesign the composition.

- **Discovery as inspiration.** During the matchmaking and reengineering process, the retailer can discover that his abstract composition is not the optimal design. Most likely, the retailer can experience this situation during the assessment of suitable service candidates. For example, a new kind of service might inspire the composition modeller to restructure the existing tasks of the composition. Then, the development starts over with the redesign of the composition as well.

- **Too restrictive QoS.** The selection can reveal that the matched candidates cannot satisfy the requested QoS. Alternatively, some of the selected candidates do not provide any statements about the considered QoS characteristics. In both cases, the result is that no services that meet the QoS requirements are available. In the first case, the retailer needs to adjust the statement of the corresponding QoS characteristic, if loosening the requirements is possible. Then, the selection process can be performed again among the candidate services. If the retailer cannot weaken the requirements or abandon any QoS characteristic, the situation is similar to the first point: Either a new service is created or the functionality of the tasks is adapted.

- **Too many candidates.** Contrary to the previous scenarios, the selection process might reveal that too many services meet the QoS requirements. Of course, this case does not represent a problem. However, the retailer can take this as an opportunity to improve the QoS. Obviously, the candidate services would allow even stronger requirements and a better QoS can be achieved. The retailer can perform the activities of the first step again to re-define the QoS requirements. Then, the retailer can repeat the selection of the third step.

The above-mentioned issues occur during the development of the composition. Monitoring and analysing the composition during the run-time can also lead further iterations of the development process. At run-time, a composition can be monitored to measure the resulting QoS. The QoS at run-time can differ from the advertised QoS. Thus, the retailer might consider measuring the execution with respect to the relevant QoS criteria. The goal of such an analysis is to identify potential weak or critical services in the composition. The retailer has three main options when a weak or critical service is revealed:

- **Replacing.** He can replace the particular service with other identified candidates that were ranked lower than the chosen service. However, the service was originally sorted out by a selection process. Consequently, this modification results in a worse QoS prediction for the composition.

- **Acquiring new services.** As an alternative, the discovery must be repeated and the “bad” service must be excluded. The modeller can perform a new attempt a new service discovery to find services that were previously not available.

- **Redesigning the composition.** Like for the iteration in the development process, the modeller can try to redesign or rearrange the required functionality to avoid the problem of insufficient QoS. In this case, a new iteration of the development process is started.

The retailer will continuously need to optimise the composition in order to approach the optimal setup. One reason is the general volatility of the QoS, which is a characteristic of distributed systems. Another reason is that new services will appear, as well as other services will disappear. To guarantee a constant QoS level of the entire composition, a new selection must take place when the QoS of the involved services changes. To ensure that new, possibly QoS-improving services are utilised, the process steps from performing the discovery can be repeated at a regular basis. Figure 7.9 outlines the relations of the main steps with the run-time part. In this figure, the activity with the grey border represents the selection. The diagram shows that the selection is performed each time the set of involved services or their QoS has changed. A repeated selection always requires the activities of the fourth step.

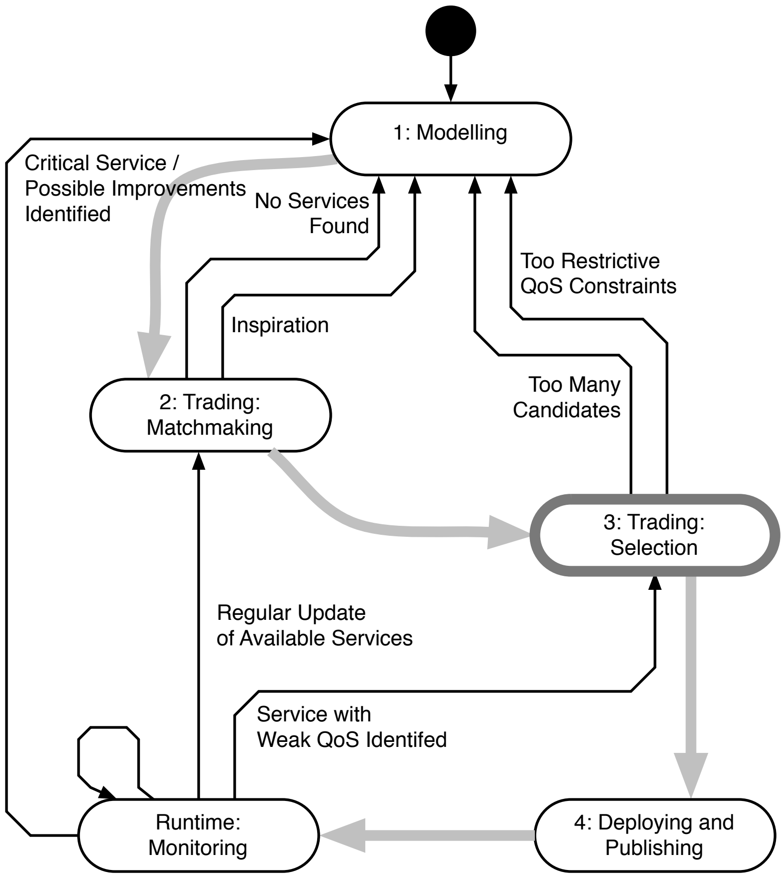

1: Modelling
Critical Service / Possible Improvements Identified
No Services Found
Too Restrictive QoS Constraints
Inspiration
Too Many Candidates
2: Trading: Matchmaking
3: Trading: Selection
Regular Update of Available Services
Service with Weak QoS Identifed
Runtime: Monitoring
4: Deploying and Publishing

Figure 7.9: The iterative development process.

Chapter 8

Conclusions

The application of heuristic algorithms to the problem of the QoS-based selection when forming service compositions was explained and analysed. In addition, the implementation of a simulation in order to analyse the performance of the considered algorithms was presented. The results from conducted simulations allow quantitative statements about the feasibility of the heuristic algorithms when compared to algorithms that guarantee to solve the problem in an optimal way.

The concept of QoS as preference criteria for the service selection covers many aspects. The definition from the ISO 9004 standard [127] on services and quality allows many possible application scenarios. To sharpen the focus of this work, the scenario of realising business processes was presented as the main application case. The motivation for this application case lies in the presented commonalities of business processes with the characteristics of service compositions (cf. Section 2.4).

The application scenario influenced the direction of the discussion on QoS in distributed systems (cf. Chapter 3). Thus, also particular QoS issues in the field of SOAs and Web services were presented. For the subsequent QoS-aggregation, existing research works on QoS issues in these fields were also assessed. Then, a uniform approach for aggregating the QoS was defined and explained for processing the QoS of individual services in compositions. The approach presumes that the composition can be described by using a specific structural model. Based on the structural model and the corresponding aggregation rules, the presented method can determine the resulting QoS of service compositions. This functionality represents the foundation for the QoS-based selection of service candidates.

After this first part, which presented the necessary foundations, an analysis of the QoS-based selection problem was given which resulted in a problem model. This model enabled an analysis on the relations to other combinatorial problems. The considered problems were the multiple choice knapsack problem, the resource-constrained project scheduling problem, the QoS-based query planning for queries to database systems, and the QoS-based routing in computer networks. In addition, the problem model served also as the foundation for a discussion about the hardness of the QoS-based selection problem. It was shown that the presented problem model for the QoS-based selection describes a problem that is NP-hard. The NP-hardness is shown by reducing the multiple choice knapsack problem to the selection problem (cf. Section 5.3.5). In addition, the comparison with the MCKP has revealed that the approaches as presented in the related work based on the MCKP do not cover all aspects of the QoS-based selection problem.

This represents the motivation to evaluate heuristic algorithms for the selection problem as an alternative to algorithms that guarantee to find the optimal solution. A selection of heuristic algorithms was explained and their advantages and disadvantages were discussed. To provide quantitative results, an evaluation of these heuristics was also presented (cf. Chapter 6). The evaluation was performed by software that simulates the application of these heuristics on randomly generated problem instances. The goal of the simulation was to capture two aspects of the performance of the heuristics: Their QoS performance and their computational time performance. Based on the quantitative results, also the weaknesses and strengths of the heuristics were revealed and discussed. In summary, the evaluation revealed the following insights about the heuristic algorithms:

- The discarding heuristic showed an unattractive consumption of the computation time under real-world conditions. This shows that a branch-and-bound strategy is not able to efficiently save computational efforts. However, this approach results in the almost optimal QoS for the composition.
- The bottomup heuristic required very few computational efforts. However, the resulting QoS reaches only $1/3$ to $1/2$ of the best QoS possible in the different simulation campaigns. Thus, its application must take a clear QoS penalty into account.
- The local heuristic, which implemented a greedy strategy to solve the given problem, resulted in very attractive QoS (about $4/5$ of the optimal) while requiring low computational efforts. However, it is not able to consider constraints. Moreover, it showed weaknesses for problem instances with very diverse QoS offers.
- The pattern-based heuristic showed a QoS performance between the local and discarding heuristic. However, it scales unattractive with larger problem instances under real-world conditions. Although, it consumes only a fraction of computation time when compared to the discarding method, it requires unattractive computation times for problem instances with a larger number of candidates or tasks.

The presentation about the abstract development process at the end has clarified the role of the QoS-based selection when creating compositions of services. By explaining this development process it was made clear at which points in the process the QoS-based selection must take place. Moreover, the development process has also explained the motivation for the following preconditions for the QoS-based selection:

- **Functional compatibility.** For the application of the QoS-based selection, the considered candidates have been proven functionally compatible to the tasks. Besides the practical benefit of this step, the advantage is that sorting out service candidates in advance results in fewer candidates to evaluate.

- **Uniformity of QoS characteristics.** The other main presumption is the uniformity of the QoS characteristics: The retailer who performs the selection process must have the same understanding of the considered QoS characteristics as the service providers who offer their candidates with QoS promises. Chapter 3 has also discussed related research work that considers this problem. Since the focus of this thesis lies on the combinatorial aspect of the problem, the QoS uniformity among the involved parties was not further discussed.

8.1 Summary of Main Contributions

The previous section has summarised the main contribution of this work which is the discussion and evaluation of the heuristic algorithms. Thus, it directly covers the problem statement given in the first chapter. Besides the evaluation results about the heuristic approaches, the following scientific contributions were presented:

- **Assessment about the QoS in service compositions.** The first part of this thesis has provided a survey about the existing research work and clarified relevant QoS concepts for the use in service compositions. The outcome is an assessment about relevant QoS characteristics, as given in Section 3 and an assessment about the resulting value dimensions when the relevant characteristics are applied in the field of services and service compositions to determine the simulation parameters (cf. Section 6.3.1).

  Moreover, Chapter 7 has explained the issues of aggregating QoS and performing the QoS-based selection in a development process for building service compositions.

- **An aggregation model.** The next contribution is a structural model an a method that enables the aggregation of QoS in service compositions, based on the QoS of individual tasks or candidates. Besides the application of this model to perform the QoS-based selection, this approach has been also applied in related work about improving the QoS by applying redundant services structures [58, 59]. The proposed approach presumes that the composition can be expressed by using this structural model. This thesis has discussed the expressiveness and limitation of this structural model. Based on an analysis of the identified limitations, the application of existing transformations (cf. Kiepuszewski et al. [73]) was proposed to cover not conforming compositions with this structural model.

- **The problem model.** Besides the structural model to perform the aggregation, another model describes the selection problem itself. The problem model uses the structural model of the aggregation method to describe the structure of the occurring problem instances. In addition, the rules of the aggregation model were used to derive statements that express constraints and optimisation goals.

  The problem model provides the foundation for three aspects: *1)* a clarification of the characteristics of the problem and its relation to existing problem models, *2)* a discussion about the effort to solve the problem and *3)* a blueprint for the implementation of the simulation in order to achieve the quantitative results.

- **Development of the simulation.** The presented performance evaluation required the implementation of a simulation. Existing works about QoS in service compositions were assessed in order to determine the parameterisation of the setup. The results from performed simulations made clear that a variation of different parameters results also in a variation of the performance. Thus, the assessment of existing work was necessary to deliver a realistic simulation setup that would result in realistic performance evaluations.

The structural model and the problem model were also the foundation for the analysis about the related work in the field of QoS aggregation (cf. Section 4.6) and in the field of QoS-based selection (cf. Section 5.1). The problem model served as a reference to show the differences to existing related work about the QoS-based selection that has considered the multiple choice knapsack problem as the appropriate problem model.

8.2 Outlook and Future Work

Because the definition of services and their composition by the ISO 9000 and 9004 standards [128, 127] is considered for this work, the discussed combinatorial problem can be applied outside an SOA and Web service infrastructures as well. The given discussion can be transferred to other domains. A possible application would be the production of goods where a product is comprised of different sub-products and services of different suppliers. For example, the production of a car represents such a scenario. The involved sub-products and services are applied in a certain order, and require a time to perform. Products cost a particular amount of money and everything operates at a certain level of reliability. The optimisation of quantifiable measures as presented in this work is also relevant for other domains which have the goal to implement a process.

Besides other potential applications of QoS-based selection in the industry, the simulations have shown in general that the optimisation of the QoS provides significant advantages in terms of cost, response time, and availability: The evaluation of the (QoS-ignoring) random and global selection methods have resulted in quantitative statements about the improvements when the QoS is optimised. All considered measures represent relevant objectives for optimisation in today's IT infrastructures when it comes to saving money and efforts (or the amount of work) and to provide systems that are more reliable.

This work has also mentioned that QoS-aware development tools are currently not widely used. Besides research prototype work (for example, presented by Grønmo and Jaeger [39]), no commercial product was identified that provides the support for the QoS of services when developing service compositions. The implemented simulator represents a solution that has proven to be capable of processing the QoS of service candidates, performing the aggregation and has demonstrated the application of different selection algorithms. This implementation serves as proof of concept for the proposed QoS-based selection. It can be further developed for the integration into software packages to create service compositions in the sense of the SOA. This thesis proposes how the discussed approaches find their way into software products, service brokers or development environments in order to create compositions as discussed in Chapter 7.

Besides the application in the software industry, the main contributions of this thesis also enable different opportunities for further research work. Possible areas of further research based on this work are:

- **Developing more heuristics.** Along with the given analysis of the four heuristics, a combined heuristic approach can be derived from further research in the field of related combinatorial problems. The result could be a hybrid approach as mentioned in Section 6.5. For example, the QoS performance of the pattern selection can be combined with the computational performance of the local selection by applying the local selection for the rare cases that pose difficulties to the pattern selection.

- **Designing campaigns.** Besides the evaluation of additional heuristics, different simulation campaigns can be also considered in order to explore more aspects of the heuristics. For example, a different setup could vary how many tasks a structural element contains on average.

- **Other approaches.** Besides the heuristics, another research opportunity would be the creation of an algorithm that derives the appropriate set of integer programming formulations according to a given instance of the selection problem. Such an algorithm could be based on defined transformations that transform each pair of QoS characteristic and composition pattern into a corresponding formula. Then, software tools for integer programming solving could be applied and the performance of such an approach can be compared with the performance of the heuristics.

- **Optimisation Prediction.** Based on the simulation results and the problem model, a functionality can be established that predicts the optimisation potential of a given composition setup. The goal of this function is to quickly predict the QoS optimisation potential. Based on this prediction, a QoS-based broker can decide whether to run a time-consuming selection algorithm or to prefer a heuristic algorithm.

Appendix A

Specification of the Hard- and Software Platform

The entire simulation presented in this work was performed on standard PC hardware running a Microsoft Windows 2000 operating system. Table A.1 lists the detailed specifications of the computer.

CPU	Intel Pentium IV CPU 2.4GHz, 512KB level-2 cache, 533MHz FSB (family 15, model 2, stepping 9)
Hardware	Intel i865 chipset with integrated graphics controller 512 MB DDR333 main memory, 40GB hard disk
OS	Microsoft Windows 2000 (version 5.0.2195, service pack 4, build 2195) performance setting “optimised for applications”
Java VM	Sun’s Java HotSpot Client VM 1.5.0_06-b05 default settings (i.e. for heap sizes)

Table A.1: Specification of the host system.

Apart from the technical specifications, the performance of the simulation host platform is also interesting in order to compare the results with future or external research. An official Java benchmark does not exist. However, different (research) groups have developed and published software that performs typical scientific calculations and captures the computational performance.

Among these efforts, “SciMark 2.0” has been chosen to measure the performance of the computer used for the simulation runs. SciMark was developed by Roldan Pozo, and Bruce Miller at the National Institute of Standards and Technology (NIST). The software is freely available at http://math.nist.gov/scimark2/. Running the benchmark tool in the command line of the host system resulted in the following output:

```
SciMark 2.0a
Composite Score: 187.41090663144433
FFT (1024): 83.56844871970135
SOR (100x100):   322.88120458089793
Monte Carlo : 24.54603686755795
Sparse matmult (N=1000, nz=5000): 127.85017738323931
LU (100x100): 378.20866560582505
java.vendor: Sun Microsystems Inc.
java.version: 1.5.0_06
os.arch: x86
os.name: Windows 2000
os.version: 5.0
```

The captured composite score of 187 compares well with the given results from other computers available on the home page of SciMark: Different other Pentium IV-based machines at 2.4Ghz clock rate resulted in the score of 187 as well.

Appendix B

List of Abbreviations

BPEL4WS	Business Process Execution Language for Web Services
BPMI	Business Process Management Initiative
BPML	Business Process Modelling Language
BPMN	Business Process Modelling Notation
BPSS	Business Process Specification Schema
CPA	Collaboration Protocol Agreement
CPP	Collaboration Protocol Profile
CPU	Central Processing Unit
CORBA	Common Object Request Broker Architecture
CCM	CORBA Component Model
ebXML	Electronic Business using eXtensible Markup Language
EDOC	Enterprise Distributed Object Computing
EPC	Event-driven Process Chain
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IDL	Interface Definition Language
ISO	International Organization for Standardization
IETF	Internet Engineering Task Force
ISO-OSI	ISO Open System Interconnection Model
J2SE	Java 2 Standard Edition
MDA	Model-Driven Architecture
MTBF	Mean Time Between Failure
MTTT	Mean Time to Transition
MOF	Meta Object Facility
MCDM	Multiple Criteria Decision Making
MCKP	Multiple-Choice Knapsack Problem

MKP	Multiple-Dimension Knapsack Problem
MMKP	Multiple-Dimension Multiple-Choice Knapsack Problem
OMG	Object Management Group
OASIS	Organization for the Advancement of Structured Information Standards
OWL-S	OWL Services
PC	Problem Carrier
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
RM-ODP	Reference Model for Open Distributed Processing
RCPSP	Resource Constrained Project Scheduling Problem
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SAW	Simple Additive Weighting
SWM	Structured Workflow Model
TINA	Telecommunications Information Networking Architecture
W3C	The World Wide Web Consortium
TINA-C	TINA Consortium
UML	Unified Modelling Language
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
UNSPSC	United Nations Standard Products and Services Code
UDDI	Universal Description, Discovery, and Integration
OS	Operating System
OWL	Web Ontology Language
VM	Virtual Machine
WSCI	Web Service Choreography Interface
WSLA	Web Service Level Agreement Language
WSMO	Web Service Modelling Ontology
WSOI	Web Service Offerings Infrastructure
WSOL	Web Service Offerings Language
WSDL-S	Web Service Semantics
WS-BPEL	Web Services Business Process Execution Language
WS-I	Web Services-Interoperability
WSDL	Web Services Description Language
WSFL	Web Services Flow Language
WfMC	Workflow Management Coalition
WorkSCo	Workflow with Separation of Concerns
XPDL	XML Processing Description Language
YAWL	Yet Another Workflow Language

Bibliography

[1] Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In *Proceedings of the Workshop on Information Integration on the Web*, pages 87–92, August 2003.

[2] Ali Shaikh Ali, Omer F. Rana, Rashid Al-Ali, and David W. Walker. UDDIe: An Extended Registry for Web Services. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 85, Orlando, Florida, USA, January 2003. IEEE Press.

[3] Cristina Aurrecoechea, Andrew T. Campbell, and Linda Hauw. "a survey of qos architectures". *Multimedia Systems, Special Section on Quality of Service Architectures for Multimedia Systems*, 6(3):138–151, May 1998.

[4] Sean Baker and Simon Dobson. Comparing Service-Oriented and Distributed Object Architectures. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'05)*, Agia Napa, Cyprus, November 2005. Springer Press.

[5] Luciano Baresi, Sam Guinea, and Pierluigi Plebani. WS-Policy for Service Monitoring. In *6th VLDB Workshop on Technologies for E-Services (TES'05)*, volume LNCS 3811. Springer Press, September 2005.

[6] Boualem Benatallah, Marlon Dumas, Marie-Christine Fauvet, F. A. Rabhi, and Quan Z. Sheng. Overview of Some Patterns for Architecting and Managing Composite Web Services. In *ACM SIGecom Exchanges*, pages 9–16. ACM Press, August 2002.

[7] Jean Bézivin, Grégoire Dupé, Frédéric Jouault, Gilles Pitette, and Jamal Eddine Rougui. First experiments with the atl model transformation language: Transforming xslt into xquery. In *2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture*, Anaheim, California, USA, 2003.

[8] Jean Bézivin, Slimane Hammoudi, Denivaldo Lopes, and Frédéric Jouault. Applying mda approach for web service platform. In *Proceedings of the Eighth IEEE International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 58–70, Monterey, California, USA, September 2004. IEEE Press.

[9] Jean Bézivin, Slimane Hammoudi, Denivaldo Lopes, and Frédéric Jouault. B2B Applications, BPEL4WS, Web Services and dotNET in the Context of MDA. In *Proceedings of the International Conference on Enterprise Integration and Modelling Technology (ICEIMT'04)*, Toronto, Canada, October 2004. Springer Press.

[10] Gregory Alan Bolcer and Gail Kaiser. SWAP: Leveraging the Web to Manage Workflow. In *IEEE Internet Computing*, pages 85–88. IEEE Press, January-February 1999.

[11] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web Services Architecture. http://www.w3c.org/TR/ws-arch/, February 2004.

[12] Carine Bournez and Hugo Haas. Semantic Web Services Interest Group Charter. http://www.w3.org/2003/10/swsig-charter, October 2003.

[13] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*.

Harri Deutsch Verlag, Frankfurt am Main, Germany, 4. edition, 1999.

[14] David Burdett and Nickolas Kavantzas (Eds.). WS Choreography Model Overview, W3C Working Draft 24 March 2004. Technical report, W3C, http://www.w3.org/TR/ws-chor-model/, 2004.

[15] Jorge Cardoso. *Quality of Service and Semantic Composition of Workflows*. PhD thesis, Department of Computer Science, University of Georgia, Athens, GA (USA), 2002.

[16] Jorge Cardoso, Amit Sheth, and Krys Kochut. Implementing QoS Management for Workflow Systems. Technical report, LSDIS Lab, Computer Science, University of Georgia, Athens, GA – USA, 2002.

[17] Jorge Cardoso, Amit Sheth, and John Miller. Workflow Quality of Service. In *Proceedings of International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02)*, Valencia, Spain, April 2002. Kluwer Academic Publishers.

[18] Senthilanand Chandrasekaran, Gregory S. Silver, John A. Miller, Jorge Cardoso, and Amit P. Sheth. XML-based Modeling and Simulation: Web Service Technologies and their Synergy with Simulation. In Jane L. Snowdon and John M. Charnes, editors, *Proceedings of the 34th Winter Simulation Conference: Exploring New Frontiers*, pages 606–615, San Diego, California, USA, December 2002. ACM Press.

[19] Jonathan Cranford, Ravi Mukkamala, and Vijayalakshmi Atluri. Modeling and evaluation of distributed workflow algorithms. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics: Information Systems Development (ISAS-SCI)*, pages 183–188, Orlando, Florida, USA, July 2001. IIIS.

[20] Eric Crawley, Raj Nair, Bala Rajagopalan, and Hal Sandick. A Framework for QoS-based Routing in the Internet. Informational RFC 2386, Internet Engineering Task Force (IETF), 1998.

[21] Thomas H. Davenport. *Process Innovation*. Harvard Business School Press, Boston, MA, USA, 1992.

[22] Remco M. Dijkman and Marlon Dumas. Service-Oriented Design: A Multi-Viewpoint Approach. *International Journal of Cooperative Information Systems (IJCIS)*, 13(4):337–368, 2004.

[23] Edsger W. Dijkstra. Go To Statement Considered Harmful. *Communications of the ACM*, 11(3):147–148, March 1968.

[24] Dragan Djuric. MDA-based Ontology Infrastructure. *Computer Science Information Systems (ComSIS)*, 1(1), February 2004.

[25] Erdogan Dogdu and Venkata Mamidenna. Efficient Scheduling Strategies for Web Services-based E-Business Transactions. In *6th VLDB Workshop on Technologies for E-Services (TES'05)*, volume LNCS 3811, pages 113–125. Springer Press, September 2005.

[26] Krzysztof Dudziński and Stanislaw Walukiewicz. Exact Methods for the Knapsack Problem and its Generalisations. *European Journal of Operations Research*, 28(2):3–21, 1987.

[27] Lill Kristiansen (Ed.). TINA-C Deliverable: Service Architecture, Version 5.0. TINA consortium website: http://www.tinac.com/, June 1997.

[28] Abdelkarim Erradi and Piyush Maheshwari. wsBus: QoS-aware Middleware for Reliable Web Services Interactions. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 634–639, Hong Kong, China, March 2005. IEEE Press.

[29] Assaf Arkin et al. Business Process Modeling Language (BPML). Technical Report Version 1.0, BPMI.org, 2002.

[30] Assaf Arkin et al. Web Service Choreography Interface (WSCI) 1.0. Technical report, W3C, http://www.w3.org/TR/wsci, 2002.

[31] Tony Andrews et al. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, IBM Corp., Microsoft Corp., http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/, 2003.

[32] Cristina Feier and John Domingue. D3.1 WSMO Primer. WSMO Final Draft, DERI International, April 2005.

[33] Daniela Florescu, Andreas Gruenhagen, and Donald Kossmann. XL: an XML Programming Language for Web Service Specification and Composition. In *Proceedings of the 11th international conference on World Wide Web (WWW'02)*, pages 65–76. ACM Press, May 2002.

[34] Helmut Frank, Norbert Gronau, and Hermann Krallman. *Systemanalyse im Unternehmen*. 3. Edition. Oldenbourg Verlag, München, Germany, October 2000.

[35] Svend Frølund and Jari Koistinen. Quality of Service Aware Distributed Object Systems. In *Proceedings in 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'99)*, pages 69–84, San Diego, California, USA, May 1999. USENIX.

[36] Dinesh Ganesarajah and Emil Lupu. Workflow-based Composition of Webservices: A Business Model or a Programming Paradigm? In *Proceedings of 6th International Enterprise Distributed Object Computing Conference (EDOC'02)*, pages 273–284, Lausanne, Switzerland, September 2002. IEEE CS Press.

[37] Michael R. Garey and David S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA, 1979.

[38] Michael Gillmann, Gerhard Weikum, and Wolfgang Wonner. Workflow Management with Service Quality Guarantees. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 228–239, Madison, Wisconsin, USA, June 2002. ACM Press.

[39] Roy Grønmo and Michael C. Jaeger. Model-Driven Methodology for Building QoS-Optimised Web Service Compositions. In *Proceedings of the 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'05)*, pages 68–82, Athens, Greece, May 2005. Springer Press.

[40] Roy Grønmo and Michael C Jaeger. Model-Driven Semantic Web Service Composition. In *The 12th ASIA-PACIFIC Software Engineering Conference (APSEC'05)*, pages 79–86, Taipei, Taiwan, December 2005. IEEE CS Press.

[41] Roy Grønmo, Michael C Jaeger, and Hjørdis Hoff. Transformations between UML and OWL-S. In *European Conference on Model Driven Architecture – Foundations and Applications (ECMDA'05)*, volume 3748 of *LNCS*, pages 269–283, Nuremberg, Germany, November 2005. Springer Press.

[42] Roy Grønmo, David Skogan, Ida Solheim, and Jon Oldevik. Model-Driven Web Service Development. *International Journal of Web Services Research (JWSR)*, 1(4), Oct-Dec 2004.

[43] Michael Hammer and James Champy. *A Manifesto for Business Revolution*. Harper Business, 1993.

[44] Thomas Heinis, Cesare Pautasso, and Gustavo Alonso. Design and evaluation of an autonomic workflow engine. In *Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*, pages 27–38, Seattle, Washington, USA, June 2005. IEEE Press.

[45] David Hollingsworth. The Workflow Reference Model. Technical Report TC00-1003, Workflow Management Coalition, Lighthouse Point, Florida, USA, 1995.

[46] David Hollingsworth. The Workflow Reference Model 10 Years On. (extracted from "Workflow Handbook 2004"), Workflow Management Coalition, Lighthouse Point, Florida, USA, February 2004.

[47] Michael N. Huhns and Munindar P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, January and February:75–81, 2005.

[48] Richard Hull, Michael Benedikt, Vassilis Christophides, and Jianwan Su. E-Services: A Look Behind the Curtain. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'03)*, pages 1–14, San Diego, California, USA, June 2003. ACM Press.

[49] Ching-Lai Hwang and K. Paul Yoon, editors. *Multiple Attribute Decision Making: Methods and Applications*, volume 186 of *Lecture Notes in Economics and Mathematical Systems*. Springer Press, March 1981.

[50] ISO/IEC. ITU.TS Recommendation X.902 — ISO/IEC 10746-1: Open Distributed Processing Reference Model - Part 1: Overview, August 1996.

[51] ISO/IEC. ITU.TS Recommendation X.902 — ISO/IEC 10746-2: Open Distributed Processing Reference Model - Part 2: Foundations, August 1996.

[52] ISO/IEC. ISO/ITU-T Recommendation X.641 — ISO/IEC 13236: Information Technology - Quality of Service - Framework, 1997.

[53] ISO/IEC. ITU.TS Recommendation X.950 — ISO/IEC 13235-1: Trading Function: Specification, August 1997.

[54] ISO/IEC. CD 15935 Information Technology: Open Distributed Processing - Reference Model - Quality of Service. (CD Ballot), October 1998.

[55] ISO/IEC. ITU-T Recommendation X.641 — ISO/IEC 13236: Information Technology - Quality of Service: Framework, 1998.

[56] ISO/IEC. ISO/IEC 15909-1: High-level Petri nets – Part 1: Concepts, Definitions and Graphical Notation. Published Standard, December 2004.

[57] ITU-T. ITU-T Recommendation X.200 — ISO/IEC 7498-1: Open Systems Interconnection – Basic Reference Model: The Basic Model, July 1994.

[58] Michael C. Jaeger and Hendrik Ladner. Improving the QoS of WS Compositions based on Redundant Services. In *The 2005 International Conference on Next Generation Web Services Practices (NWeSP'05)*, pages 189–194, Seoul, South-Korea, August 2005. IEEE CS Press.

[59] Michael C. Jaeger and Hendrik Ladner. A Model for the Aggregation of QoS in WS Compositions Involving Redundant Services. *Journal of Digital Information Management*, 4(1):44–49, March 2006.

[60] Michael C. Jaeger and Gero Mühl. Soft Real-Time Aspects for Service-Oriented Architectures. In *Proceedings of the IEEE Joint Conference on E-Commerce Technology (CEC'06) and Enterprise Computing, E-Commerce and E-Services (EEE'06)*, pages 22–29, San Francisco, California, USA, June 2006. IEEE Press.

[61] Michael C. Jaeger, Gero Mühl, and Sebastian Golze. QoS-aware Composition of Web Services: An Evaluation of Selection Algorithms. In *Proceedings of the Confederated International Conferences CoopIS, DOA, and ODBASE 2005 (OTM'05)*, volume 3760 of *LNCS*, pages 646–661, Agia Napa, Cyprus, November 2005. Springer Press.

[62] Michael C. Jaeger and Gregor Rojec-Goldmann. SENECA – Simulation of Algorithms for the Selection of Web Services for Compositions. In *6th VLDB Workshop on Technologies for E-Services (TES'05)*, volume 3811 of *LNCS*, pages 84–97, Trondheim, Norway, September 2005. Springer Press.

[63] Michael C. Jaeger, Gregor Rojec-Goldmann, and Gero Mühl. QoS Aggregation for Service Composition using Workflow Patterns. In *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 149–159, Monterey, California, USA, September 2004. IEEE Press.

[64] Michael C. Jaeger, Gregor Rojec-Goldmann, and Gero Mühl. QoS Aggregation in Web Service Compositions. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 181–185, Hong Kong, China, March 2005. IEEE Press.

[65] Michael C. Jaeger, Gregor Rojec-Goldmann, Gero Mühl, Christoph Liebetruth, and Kurt Geihs. Ranked Matching for Service Descriptions using OWL-S. In *Kommunikation in verteilten Systemen (KiVS 2005)*, Informatik Aktuell, pages 91–102, Kaiserslautern, Germany, February 2005. Springer Press.

[66] Michael C. Jaeger and Stefan Tang. Ranked Matching for Service Descriptions using DAML-S. In *Proceedings*

of CAiSE'04 Workshops, pages 217–228, Riga, Latvia, June 2004. Riga Technical University.

[67] Gerrit K. Janssens, Jan Verelst, and Bart Weyn. Techniques for modelling workflows and their support of reuse. In *Business Process Management – Models, Techniques, and Empirical Studies*, volume 1806 of *LNCS*, pages 1–15, Berlin, Germany, February 2000. Springer Press.

[68] Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles Irby, Marian Beard, and Kevin Mackey. The Xerox Star: A Retrospective. *Computer*, 22(9):11–26,28–29, September 1989.

[69] Sravanthi Kalepu, Shonali Krishnaswamy, and Seng Wai Loke. Reputation = f(user ranking, compliance, verity). In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 200–207, San Diego, California, USA, July 2004. IEEE CS Press.

[70] Olaf Kath, Andrei Blazarenas, Marc Born, Klaus-Peter Eckert, Motoshisa Funabashi, and Chiaki Hirai. Towards Executable Models: Transforming EDOC Behavior Models to CORBA and BPEL. In *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 267–274, Monterey, California, USA, September 2004. IEEE Press.

[71] Gerhard Keller, Markus Nüttgens, and August-Wilhelm Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi) 89, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[72] Tony Kenyon. *Data Networks: Routing, Seurity, and Performance Optimization*. Digital Press, 1. edition, June 15th 2002.

[73] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On Structured Workflow Modelling. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00)*, volume 1789 of *LNCS*, pages 431–445, Stockholm, Sweden, June 2000. Springer Press.

[74] Donald E. Knuth. Structured Programming with go to Statements. *ACM Computing Surveys*, 6(4):261–301, December 1974.

[75] Pinar Koksal, Sena Nural Arpinar, and Asuman Digac. Workflow History Management. *SIGMOD Record*, 27(1):67–75, 1998.

[76] Lea Kutvonen. *Trading Services in Open Distributed Environments*. PhD thesis, Department of Computer Science, University of Helsinki, Helsinki, Finland, 1998.

[77] Hendrik Ladner. Methoden zur Optimierung der Dienstgüte von Web Service Kompositionen. diploma thesis, Berlin University of Technology, Faculty of Electrical Engineering and Computer Science, October 2005.

[78] Juhnyoung Lee. Matching Algorithms for Composing Business Process Solutions with Web Services. In *Proceedings of the 4th International Conference on E-Commerce and Web Technologies (ECWEB'03)*, pages 393–402, Prague, Czechoslovakia, October 2003. Springer Press.

[79] Ulf Leser. Combining Heterogeneous Data Sources through Query Correspondence Assertions. In Fereidoon Sadri, editor, *CIKM'98 First Workshop on Web Information and Data Management (WIDM'98)*, pages 29–32, Bathesda, Maryland, USA, November 1998. ACM Press.

[80] Frank Leymann. Web Services Flow Language (WSFL 1.0). Technical report, IBM Software Group, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2001.

[81] Lei Li and Ian Horrocks. A Software Framework For Matchmaking Based on Semantic Web Technology. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 331–339. ACM Press, May 2003.

[82] Lucas Bordeaux and Gwen Salaun and Daniela Berardi and Massimo Mecella. When are two web services compatible? In *Revised Selected Papers of the 5th International Workshop on Technologies*

for E-Services (TES'04), volume 3324 of *LNCS*, pages 15–28, Toronto, Canada, August 2004. Springer Press.

[83] Heiko Ludwig. Web Services QoS: External SLAs and Internal Policies Or: How do we deliver what we promise? Keynote Speech at the WISE Workshop on Web Services Quality, December 2003.

[84] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck. Web Service Level Agreement (WSLA) Language Specification. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, January 2003.

[85] Ed Lynch and Chandra Venkatapathy. Sustaining your Advantage with Business Process Integration based on Service Oriented Architecture. White Paper, October 2005.

[86] Dirk E. Mahling, Noel Craven, and W. Bruce Croft. From Office Automation to Intelligent Workflow Systems. *IEEE Intelligent Systems*, 10(3):41–47, June 1995.

[87] Mike Marin, Justin Brunt, Wojciech Zurek, Tim Stephenson, Sasa Bojanic, and Gangadhar Gouri. Workflow Process Definition Interface – XML Process Definition Langauge, Version 1.0. Technical Report WFMC-TC-1025, Workflow Management Coalition, Lighthouse Point, Florida, USA, October 2002.

[88] E. Michael Maximilien and Munindar P. Singh. A Framework and Ontology for Dynamic Web Services Selection. In *IEEE Internet Computing*, pages 84–93. IEEE Press, September-October 2004.

[89] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Technical report, W3C, http://www.w3.org/TR/owl-features/, 2004.

[90] Daniel A. Menasce. QoS Issues in Web Services. In *IEEE Internet Computing*, pages 72–75. IEEE Press, November-December 2002.

[91] Daniel A. Menasce. Composing Web Services:A QoS View. In *IEEE Internet Computing*, pages 88–90. IEEE Press, November–December 2004.

[92] Daniel A. Menasce. Response-Time Analysis of Composite Web Services. In *IEEE Internet Computing*, pages 90–92. IEEE Press, January–February 2004.

[93] Microsoft. Enterprise UDDI Services: An Introduction to Evaluating, Planning, Deploying, and Operating UDDI Services, February 2003.

[94] Klara Nahrstedt and Jonathan M. Smith. The QoS Broker. *IEEE MultiMedia*, 2,(1):53–67, Spring 1995.

[95] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 447–458, Edinburgh, September 1999. Morgan Kaufmann.

[96] Object Management Group (OMG). Model Driven Architecture. ormsc/2001-07-01, August 2001.

[97] Object Management Group (OMG). CORBA Components. OMG formal document/02-06-65, 2002.

[98] Object Management Group (OMG). Meta Object Factility Specification. OMG formal document/2002-04-03, April 2002.

[99] Object Management Group (OMG). UML Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms. ptc/2004-06-01, June 2004.

[100] Object Management Group (OMG). Unified Modeling Language: Superstructure. OMG formal document/05-07-04, August 2005.

[101] Chun Ouyang, Marlon Dumas, Stephan Breutel, and Arthur H.M. ter Hofstede. Translating Standard Process Models to BPEL. In *Proceedings of the 18th International Conference on Advanced Information Systems Engineering*, Luxembourg, June 2006. Springer Press.

[102] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Importing the Semantic Web in UDDI. In *Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 225–236, Toronto, Canada, May 2002. Springer Press.

[103] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic Matching of Web Service Capabilities. In *Proceedings of 1st International Semantic Web Conference. (ISWC'02)*, volume 2342 of *LNCS*, pages 333–347, Sardinia, Italy, June 2002. Springer Press.

[104] Mike P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, pages 3–12, Roma, Italy, December 2003. IEEE CS Press.

[105] Chintan Patel, Kaustubh Supekar, and Yugyung Lee. Provisioning Resilient, Adaptive Web Services-based Workflow: A Semantic Modeling Approach. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 480–487, San Diego, California, USA, July 2004. IEEE CS Press.

[106] Terry R. Payne, Massimo Paolucci, and Katia Sycara. Advertising and Matching DAML-S Service Descriptions. In *Position Papers for SWWS'01*, pages 76–78, Stanford, USA, July 2001. Stanford University.

[107] Massimo Pezzini. SOA Beyond Hype and Disillusionment – A Strategic Perspective. Key Note given at the SOA Days 2005 Technology Conference, September 2005.

[108] David Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, Dept. of Computer Science, University of Copenhagen, Copenhagen, Denmark, February 1995.

[109] IONA Technologies PLC. Artix: the Extensible Enterprise Service Bus (ESB). product website, http://www.iona.com/products/artix/, 2005.

[110] United Nations Development Programme. United Nations Standard Products and Services Code. organisation website, http://www.unspsc.org/, 2005.

[111] Peter Puschner and Anton Schedl. Computing Maximum Task Execution Times - A Graph-Based Approach. *Journal of Real-Time Systems*, 13(1):67–91, July 1997.

[112] Shuping Ran. A Model for Web Services Discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.

[113] Akhil Sahai, Anna Durante, and Vijay Machiraju. Towards Automated SLA Management for Web Services. Technical Report HPL-2001-310, Software Technology Laboratory, HP Laboratories Palo Alto, Palo Alto, California, USA, 2002.

[114] Uwe Schöning. *Theoretische Informatik - kurz gefasst*. Spektrum Akademischer Verlag, Heidelberg, Germany, 3. edition, 1999.

[115] Seema Degwekar and Stanley Y. W. Su and and Herman Lam. Constraint specification and processing in web services publication and discover. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 210–217, San Diego, California, USA, June 2004. IEEE CS Press.

[116] Mohamed A. Serhani, Rachida Dssouli, Houari Sahraoui, Abdelghani Benharref, and M. E. Badidi. QoS Integration in Value Added Web Services. In *Proceedings of the Second International Conference on Innovations in Informal Technology (IIT'05)*, September 2005.

[117] Dave Shaffer and Brian Dayton. Orchestrating Web Services: The Case for a BPEL Server. Technical report, Oracle Corporation, Redwood Shores, California, USA, June 2004.

[118] Robert Shapiro, Mike Marin, Justin Brunt, Wojciech Zurek, Tim Stephenson, Sasa Bojanic, and Gangadhar Gouri. Process Definition Interface – XML Process Definition Language, Version 2.0. Technical Report WFMC-TC-1025, Workflow Management Coalition, Lighthouse Point, Florida, USA, October 2005.

[119] Kaarthik Sivashanmugam, Kunal Verma, Amit P. Sheth, and John A. Miller. Adding Semantics to Web Services Standards. In *Proceedings of the International Conference on Web Services (ICWS '03)*, pages 395–401, Las Vegas, Nevada, USA, June 2003. CSREA Press.

[120] David Skogan, Roy Grønmo, and Ida Solheim. Web Service Composition in UML. In *Proceedings of the 8th IEEE*

Intl Enterprise Distributed Object Computing Conf (EDOC'04), pages 47–57, Monterey, California, USA, September 2004. IEEE Press.

[121] Howard Smith and Peter Fingar. Business Process Fusion Is Inevitable. Business Process Trends, Columns and Articles, March 2004.

[122] Howard Smith and Peter Fingar. Workflow is just a Pi Process. Business Process Trends, Columns and Articles, January 2004.

[123] Naveen Srinivasan, Massimo Paolucci, and Katia Sycara. Adding OWL-S to UDDI, Implementation and Throughput. In *Proceedings of Semantic Web Service and Web Process Composition 2004*, San Diego, California, USA, July 2004.

[124] Katia Sycara, Massimo Paolucci, Julien Soudry, and Naveen Srinivasan. Dynamic Discovery and Coordination of Agent-Based Semantic Web Services. *IEEE Internet Computing*, 8(3):66–73, May, June 2004.

[125] Katia Sycara, Seth Widoff, Matthias Klusch, and Jianguo Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, 2002.

[126] OASIS WS-BPEL TC. WS-BPEL Specification Editors Draft. http://www.oasis-open.org/committees/download.php/12791/wsbpel-specification-draft-May-20-2005.html, December 2005.

[127] Technical Committee ISO/TC 176, Quality Management and Quality Assurance. Quality Managemant and Quality System Elements; Part 2: Guidelines for Services, 1991.

[128] Technical Committee ISO/TC 176, Quality Management and Quality Assurance. Quality Management and Quality Assurance Standards; Part 1: Guidelines for Selection and Use, 1994.

[129] Satish Thatte. XLANG - Web Services for Business Process Design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001.

[130] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical report, The DARPA Agent Markup Language (DAML) Program, http://www.daml.org/services/, 2004.

[131] Min Tian, A. Gramm, Hartmut Ritter, and Jochen H. Schiller. Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework. In *The 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pages 152–158, Beijing, China, September 2004. IEEE Press.

[132] Vladimir Tosic, Wei Ma, Bernard Pagurek, and Babak Esfandiari. Web Service Offerings Infrastructure (WSOI) – A Management Infrastructure for XML Web Services. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'04)*, pages 817–830, Seoul, South Korea, April 2004. IEEE Press.

[133] Vladimir Tosic, Kruti Patel, and Bernard Pagurek. WSOL – Web Service Offerings Language. In *Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web - WES (at CAiSE'02)*, volume 2512 of *LNCS*, pages 57–67, Toronto, Canada, May 2002. Springer Press.

[134] David Trastour, Claudio Bartolini, and Chris Preist. A Semantic Web Approach to Service Description for Matchmaking of Services. In *Proceedings of the 11th international conference on World Wide Web (WWW'02)*, pages 89–98, Honolulu, USA, May 2002. ACM Press.

[135] UDDI Spec Technical Committee. UDDI Version 3.0.1. http://uddi.org/pubs/uddi-v3.0.1-20031014.pdf, 2003.

[136] Guijun Wang, Alice Chen, Changzhou Wang, Casey Fung, and Stephen Uczekaj. Integrated Quality of Service (QoS) Management in Service-Oriented Enterprise Architectures. In *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 21–32, Monterey, California, USA, September 2004. IEEE Press.

[137] Zheng Wang and Jon Crowcroft. Quality of Service Routing for Supporting Multimedia Applications . *IEEE Journal of Selected Areas in Communications (JSAC)*, 14(7):1228–1234, September 1996.

[138] Gerhard Weikum. Towards Guaranteed Quality and Dependability of Information Systems. In *8th GI Fachtagung: Datenbanksysteme in Buero, Technik und Wissenschaft*, pages 379–409, Freiburg, Germany, March 1999. Springer Press.

[139] Wen-Lin Yang. Optimal and heuristic algorithms for quality-of-service routing with multiple constraints. *Performance Evaluation*, 57(3):261–278, January 2004.

[140] Stephen A. White. Business Process Modeling Notation (BPMN). Technical Report Working Draft (1.0), BPMI.org, August 2003.

[141] Wil M. P. van der Aalst. Why workflow is NOT just a Pi-Process. Online Edition at BPTrends, BPMI.org, February 2004.

[142] Wil M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In *Business Process Management – Models, Techniques, and Empirical Studies*, volume 1806 of *LNCS*, pages 161–183, Berlin, Heidelberg, New York, February 2000. Springer Press.

[143] Wil M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. *Jan/Feb 2003 Issue of IEEE Intelligent Systems*, pages 72–76, January 2003.

[144] Wil M.P. van der Aalst. Pi calculus versus petri nets: let us eat humble pie rather than further inflate the pi hype. BPTrends, 3(5):1-11, May 2005.

[145] Wil M.P. van der Aalst and Arthur H.M. ter Hofstede and Bartek Kiepuszewski and Alistair P. Barros. Advanced Workflow Patterns. In *7th International Conference on Cooperative Information Systems (CoopIS'00)*, volume 1901 of *LNCS*, pages 18–29, Berlin, Germany, 2000. Springer Press.

[146] Wil M.P. van der Aalst and Arthur H.M. ter Hofstede and Bartek Kiepuszewski and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[147] Wil M.P. van der Aalst and Jens B. Jørgensen and Kristian B. Lassen. Let's Go All the Way: From Requirements Via Colored Workflow Nets to a BPEL Implementation of a New Bank System. In *Proceedings of the Confederated International Conferences CoopIS, DOA, and ODBASE 2005 (OTM'05)*, volume 3760 of *LNCS*, pages 22–39, Agia Napa, Cyprus, November 2005. Springer Press.

[148] Wil M.P. van der Aalst and Kees M. van Hee and G. J. Houben. Modelling Workflow Management Systems with high-level Petri Nets. In G. De Michelis and C. Ellis and G. Memmi, editor, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31–50, 1994.

[149] Wil M.P. van der Aalst and Lachlan Aldred and Marlon Dumas and Arthur H.M. ter Hofstede. Design and implementation of the YAWL system. Technical Report FIT-TR-2003-07, Centre for IT Innovation, QUT, http://www.tm.tue.nl/it/research/patterns, 2004.

[150] Wil M.P. van der Aalst and Marlon Dumas and Arthur H.M. ter Hofstede and Petia Wohed. Pattern Based Analysis of BPML (and WSCI). FIT Technical Report FIT-TR-2002-05, Queensland University of Technology, Brisbane, Australia, 2002.

[151] Petia Wohed, Wil M.P. van der Aalst, Marlon Dumas, and Arthur H.M. ter Hofstede. Pattern Based Analysis of BPEL4WS. Technical Report FIT-TR-2002-04, QUT, Queensland University of Technology, Queensland, Australia, 2002.

[152] Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, and Erich Neuhold. Matchmaking for business processes based on choreographies. *International Journal of Web Services Research*, 1(4):14–32, October-December 2004.

[153] Bibo Yang, Joseph Geunes, and William J. O'Brien. Resource Constrained Project Scheduling; Past Work and New Directions. Technical Report Research Report 2001-6, Department of Industrial and Systems Engineering, University of Florida, 2001.

[154] Jian Yang. Web Service Componentization. *Communications of the ACM*, 46(10), October 2003.

[155] Jian Yang, Mike P-Papazoglou, and Willem-Jan van den Heuvel. Tackling the Challenges of Service Composition in E-Marketplaces. In *Proceedings of the 12th International Workshop on Research Issues in Data Engineering (RIDE'02)*, pages 125–133. IEEE, February 2002.

[156] Martin Yates, Wataru Takita, Rickard Jansson, Laurence Demounem, and Harm Mulder. TINA-C Deliverable: TINA Business Model and Reference Points. http://www.tinac.com/, May 1997.

[157] Tao Yu and Kwei-Jay Lin. A Broker-Based Framework for QoS-Aware Web Service Composition. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE'05)*, pages 22–29, Hong Kong, China, March 2005. IEEE Press.

[158] Tao Yu and Kwei-Jay Lin. Service Selection Algorithms for Web Services with End-to-End QoS Constraints. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 129–136, Hong Kong, China, March 2005. IEEE Press.

[159] 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 Transactions*, 30(5):311–327, May 2004.

Nachwort

Die Anfertigung einer Dissertation scheint die Charakteristik aufzuweisen, dass sich der Autor in die Arbeit stürzt, dazu tendiert, gesellschaftliche Aktivitäten zu reduzieren, und sich im permanenten Stresszustand befindet, weil der Eindruck besteht, dass die unerfreulichen Einsichten während der Anfertigung überwiegen. Umso mehr danke ich Paola dafür, dass sie mir mit einem Mittelweg aus Konfrontation und Verständnis begegnete und mich dazu ermahnte, mich während der Arbeit in einem vertretbaren Modus zu bewegen. Um die Ausmaße zu verdeutlichen, möchte ich erwähnen, dass mein Verhalten eine befreundete Sprachlehrerin inspirierte, eine Figur zu erfinden: Stefan, ein Statistiker, un secchione, der den ganzen Tag aufgeregt über Zahlenreihen sitzt. Er ist verlobt mit einer Italienerin, die gerne kocht . . . (vgl. Langenscheidt Sprachkalender Italienisch 2008).

Ich hatte das besondere Glück, dass mir Unterstützung nicht nur aus einer, sondern aus zwei Arbeitsgruppen (FLP und BKS) zuteil wurde. Vielen Dank daher an Chunyan, Malte, Sebastian B., Steffen, Sebastian G., Jens, Michael A., Narcisse, Helge, Matthias und Tina. Wer die Zwischenstände meiner Arbeit gesehen oder vorgetragen bekommen hat, wird sicherlich bestätigen, dass die Diskussionen und Anregungen mit und von meinen Kolleginnen und Kollegen für mich sehr hilfreich waren. Ganz besonderer Dank gebührt natürlich Gero Mühl, der mich mit großer Initiative und seinen humorvollen Darstellungen unterstützt und motiviert hat. In ähnlicher Form muss ich mich bei James und Gregor für deren außergewöhnliche Hilfsbereitschaft bedanken. Gerhard, Michael P. und Annelie danke ich für Rat und Tat in der Mathematik und im Englischen.

Bernd Mahr danke ich für die Betreuung meiner Dissertation; Kollegen schrieben zu einem ähnlichem Anlass: "*. . . zu deren Entstehung . . . Bernd mit seiner großen Begabung in sokratischer Hebammentechnik hilfreich beigetragen hat.*" Treffender kann ich es nicht formulieren. Ich möchte mich an dieser Stelle auch bei Herrn Tolksdorf bedanken, der sich bereit erklärte, die Rolle des zweiten Gutachters zu übernehmen. Ebenfalls bedanke ich mich bei Herrn Heiss, der den Vorsitz im Ausschuss übernommen hat. Last but not least, möchte Kurt Geihs meinen Dank aussprechen – bei ihm als Mitarbeiter am Lehrstuhl für verteilte Systeme an der TU Berlin hat alles angefangen.

Berlin, Dezember 2006.