Interdependent Security and Compliance in Service Selection

An der Fakultät für Wirtschaftswissenschaften der

Universität Regensburg

zur Erlangung des akademischen Grades

Doktor der Wirtschaftswissenschaften
(Dr. rer. pol.)

eingereichte

DISSERTATION

vorgelegt von

Dipl. Wirt.-Inform. Harun Fatih Karataş

Vorsitzende:	Prof. Dr. Susanne Leist
Berichterstatter:	Prof. Dr. Doğan Kesdoğan
	Prof. Dr. Bernd Heinrich

Tag der Einreichung:	05.02.2016
Tag der Disputation:	05.10.2016

This thesis is dedicated to my parents Sevil and Harun, my siblings Dilek and Mehmet Selim and last but not least to my wife Muteber.

Acknowledgments

> *"As gravity is the essential quality of matter,*
> *freedom is the basic quality of the spirit.*
> *Human freedom is, first of all,*
> *the freedom to perform creative work."*
>
> – Georg Friedrich Wilhelm Hegel

This thesis is the result of a journey which started in November 2010 when I began working on the ReSCUeIT project at Siegen University. The project was about multilateral security in supply chains and back then I thought that I would write my thesis in this field of research. Obviously the topic has changed since then. Actually this was the first and maybe most important lesson that I learned as a PhD student: The path to the truth is not straight, but rather sloped with many crossings of which some lead to dead ends. What makes navigating even more complicated is that there is neither an accurate map which one could follow nor a clear goal. Often it is the gut feeling of an experienced wanderer which tells him which path might be the right one. I am glad, that there was such an experienced wanderer, namely Prof. Dr. Doğan Kesdoğan, to whom I could turn whenever I got lost and who led me back on track. Without his guidance and fruitful discussions this thesis would certainly not have been finished. I am also indebted to Prof. Dr. Bernd Heinrich for his invaluable hints and comments as well as a series of mail discussions. His input vastly improved the quality of this work.

I would also like to thank my former colleagues at the chair for IT security management at Siegen University: Prof. Dr. Thomas Barth, Dr. Dhiah el Diehn I. Abou-Tair, Dr. Ezzaldeen Edwan, Dr. Lars Fischer, Dr. Jens Gulden, Thomas Fielenbach, Marcel Heupel, Philipp Schwarte, Ralf Dreier, Sivinay Maass, Vanessa Zimny and Beatrice Teuchert. Our working environment was characterized by focus, heated debates and lots of fun. Not to mention our working sessions on cloud computing at local hookah bars. I would like to express my special gratitude to Mohamed Bourimi for sharpening my view on the important aspects when presenting results.

As I am writing these sentences I realize that there are far too many people I could mention here for their impact on this work, either intentionally or unintentionally. I am grateful to all of them! For several fruitful discussions on optimization I would like to thank Dr. JENNY NOSSACK and Dr. DOMINIK KRESS. At one occasion, Dr. KRESS sketched an idea that would later become the initial feasibility check of COMPAGA. I would also like to thank Dr. RICARDO TESORIERO and Dr. PEDRO VILLANUEAVA (¿Cómo es Juan?) for lots of distributed work on usability and privacy topics.

Last but not least I would like to thank my family and friends for their continuous support and motivation. Especially during times of high workloads or when facing seemingly unsolvable problems they kept me going on. Özellikle aileme teşekkür etmek isterim beni her zaman kayıtsız şartsız destekledikleri için. Iyi ki varsınız!

Fatih Karataş

Abstract

Application development today is characterized by ever shorter release cycles and more frequent change requests. Hence development methods such as service composition are increasingly arousing interest as viable alternative approaches. While employing web services as building blocks rapidly reduces development times, it raises new challenges regarding security and compliance since their implementation remains a black box which usually cannot be controlled. Security in particular gets even more challenging since some applications require domain-specific security objectives such as location privacy. Another important aspect is that security objectives are in general no singletons but subject to interdependence. Hence this thesis addresses the question of how to consider interdependent security and compliance in service composition.

Current approaches for service composition do neither consider interdependent security nor compliance. Selecting suiting services for a composition is a combinatorial problem which is known to be NP-hard. Often this problem is solved utilizing genetic algorithms in order to obtain near-optimal solutions in reasonable time. This is particularly the case if multiple objectives have to be optimized simultaneously such as price, runtime and data encryption strength. Security properties of compositions are usually verified using formal methods. However, none of the available methods supports interdependence effects or defining arbitrary security objectives. Similarly, no current approach ensures compliance of service compositions during service selection. Instead, compliance is verified afterwards which might necessitate repeating the selection process in case of a non-compliant solution.

In this thesis, novel approaches for considering interdependent security and compliance in service composition are being presented and discussed. Since no formal methods exist covering interdependence effects for security, this aspect is covered in terms of a security assessment. An assessment method is developed which builds upon the notion of structural decomposition in order to assess the fulfillment of arbitrary security objectives in terms of a utility function. Interdependence effects are being modeled as dependencies between utility functions. In order to enable compliance-awareness, an approach is presented which checks compliance of compositions during service selection and marks non-compliant parts. This enables to repair the corresponding parts during the selection process by replacing the current services and hence avoids the necessity to repeat the selection process. It is demonstrated how to embed the presented approaches into a genetic algorithm in order to ease integration with existing approaches for service composition. The developed approaches are being compared to state-of-the-art genetic algorithms using simulations.

Zusammenfassung

Anwendungsentwicklung ist heutzutage gekennzeichnet durch immer kürzere Release-Zyklen und immer häufigere Änderungswünsche. Daher wecken Entwicklungsmethoden wie Service Composition in zunehmendem Maße Interesse als praktikable Alternativansätze. Obwohl die Nutzung von Web Service-Bausteinen Entwicklungszeiten rapide senkt, führt dies auch zu neuen Herausforderungen hinsichtlich Sicherheit und Compliance weil die konkrete Implementierung in diesem Fall eine Blackbox darstellt, die üblicherweise nicht der eigenen Kontrolle unterliegt. Was die Sicherheitsherausforderungen noch vergrößert ist der Umstand, dass manche Anwendungen domänenspezifische Sicherheitsziele benötigen wie z. B. Location Privacy. Ein weiterer wichtiger Aspekt in diesem Zusammenhang ist, dass Sicherheitsziele i. A. nicht unabhängig voneinander sind, sondern in interdependenter Wechselwirkung stehen. Daher beschäftigt sich diese Arbeit mit der Frage, wie interdependente Sicherheit und Compliance im Rahmen der Service Composition beachtet werden können.

Gegenwärtige Ansätze zur Service Composition betrachten weder interdependente Sicherheit noch Compliance. Die Auswahl von passenden Services für eine Komposition ist ein kombinatorisches Problem, welches bekanntlich NP-schwer ist. Oftmals wird dieses Problem durch den Einsatz genetischer Algorithmen gelöst um in vertretbarer Zeit nahezu optimale Lösungen zu erhalten. Dies ist insb. dann der Fall, falls mehrere Ziele simultan optimiert werden sollen wie z. B. Preis, Laufzeit und Verschlüsselungsstärke. Sicherheitseigenschaften von Kompositionen werden üblicherweise mittels formaler Methoden nachgewiesen. Keine der verfügbaren formalen Methoden unterstützt jedoch die Modellierung von Interdependenzeffekten oder die Definition beliebiger Sicherheitsziele. Analog unterstützt kein derzeitiges Verfahren die Einhaltung von Compliance-Eigenschaften während der Serviceauswahl. Statt dessen wird die Compliance im Nachhinein überprüft, was es möglicherweise erforderlich macht, den Auswahlprozess zu wiederholen falls eine Lösung nicht compliant ist.

In dieser Arbeit werden neue Ansätze zur Beachtung von interdependenter Sicherheit sowie Compliance im Rahmen von Service Composition vorgestellt und diskutiert. Aufgrund des Umstands, dass es keine formalen Methoden gibt, die Interdependenzeffekte im Rahmen von Sicherheit abbilden, wird dieser Aspekt im Sinne einer Bewertung abgebildet. Es wird eine Bewertungsmethode entwickelt, welche auf der Idee der strukturellen Dekomposition aufbaut um die Erfüllung von Sicherheitszielen auf Nutzenfunktionen abzubilden. Interdependenzeffekte werden als Abhängigkeiten zwischen Nutzenfunktionen modelliert. Zur Beachtung von Compliance-Anforderungen wird ein Verfahren präsentiert, welches die Compliance von Kompositionen während der Serviceauswahl überprüft und alle Teile einer Komposition markiert, die nicht compliant sind. Dies ermöglicht es, die Compliance der entsprechenden Teile der Komposition noch während des Auswahlprozesses durch Austauschen der betreffenden Services wiederherzustellen und dadurch die Notwendigkeit einer möglichen Wiederholung des Auswahlprozesses zu vermeiden. Es wird demonstriert, wie die entwickelten Ansätze in einen genetischen Algorithmus eingebettet werden können um die Integration mit bestehenden Ansätzen zur Service Composition zu erleichtern. Die entwickelten Ansätze werden mit modernen genetischen Algorithmen mittels Simulationen verglichen.

Printing Conventions

In this thesis different type styles are used for highlighting purposes.

Person, product and organization names are written in CAPITALS.

Important terms and expressions are written with *italic* font.

Terms and expressions with a special meaning for a certain context such as aspects in answering a question or parts of a concept are written with **bold** font.

Source codes and links to websites are written with `monospaced` font.

Citations and phrases from natural language are written with “double quotation marks”.

Contents

II Design & Implementation 121

List of Figures

List of Tables

List of Listings

List of Algorithms

List of Abbreviations

AHP	Analytic Hierarchy Process.
AI	Artificial Intelligence.
ASP	Application Service Providing.
AWS	Amazon Web Services.
BDSG	Bundesdatenschutzgesetz.
BPEL	Business Process Execution Language.
BPM	Business Process Management.
BPMN	Business Process Modeling Notation.
BPMS	Business Process Management System.
BSI	Bundesamt für Sicherheit in der Informationstechnik.
CIM	Computational Independent Model.
COMPAGA	Compliance-Aware Genetic Algorithm.
CORBA	Common Object Request Broker Architecture.
COTS	Commercial Off-The-Shelf.
CP	Composition Pattern.
CSP	Constraint Satisfaction Problem.
d-KP	d-dimensional Knapsack Problem.
DAG	Directed Acyclic Graph.
DCOM	Distributed Component Object Model.
DF	Degree of Fulfillment.
EAI	Enterprise Application Integration.
EPC	Event-driven Process Chain.
GA	Genetic Algorithm.
GDP	Gross Domestic Product.

GUI	Graphical User Interface.
HIPAA	Health Insurance Portability and Accountability Act.
HTTP	Hypertext Transfer Protocol.
IaaS	Infrastructure as a Service.
IBEA	Indicator-Based Evolutionary Algorithm.
ICT	Information and Communication Technology.
IFML	Interaction Flow Modeling Language.
ILP	Integer Linear Programming.
IP	Integer Programming.
ISACA	Information Systems Audit and Control Association.
ISO	International Organization for Standardization.
IT	Information Technology.
KP	Knapsack Problem.
LEO	Lyons Electronic Office.
LP	Linear Programming.
MADM	Multiple Attribute Decision Making.
MCKP	Multiple-Choice Knapsack Problem.
MEMO	Multi-Perspective Enterprise Modeling.
MIP	Mixed Integer Programming.
MMKP	Multidimensional Multiple-Choice Knapsack Problem.
MOM	Message-Oriented Middleware.
MOO	Multi-Objective Optimization.
NIST	National Institute of Standards and Technology.
NSGA-II	Nondominated Sorting Genetic Algorithm-II.
OASIS	Organization for the Advancement of Structured Information Standards.
OMG	Object Management Group.
OSN	Online Social Network.
OWL	Web Ontology Language.

PaaS	Platform as a Service.
PIM	Platform Independent Model.
PS	Personal Service.
PSM	Platform Specific Model.
QoS	Quality of Service.
RBAC	Role-Based Access Control.
RDF	Resource Description Framework.
REST	Representational State Transfer.
RMS	Root Mean Square.
RPC	Remote Procedure Call.
SaaS	Software as a Service.
SAW	Simple Additive Weighting.
SLA	Service-Level Agreement.
SOA	Service-Oriented Architecture.
SOC	Service Oriented Computing.
SOO	Single-Objective Optimization.
SPEA2	Strength Pareto Evolutionary Algorithm 2.
SSW	Service Selection Workbench.
TCP	Transmission Control Protocol.
TTR	Time-to-repair.
UDDI	Universal Description, Discovery and Integration.
UML	Unified Modeling Language.
URL	Uniform Resource Locator.
VPN	Virtual Private Network.
W3C	World Wide Web Consortium.
WebML	Web Modeling Language.
WfM	Workflow Management.
WfMC	Workflow Management Coalition.
WSDL	Web Services Description Language.
WSFL	Web Services Flow Language.

WSLA	Web Service Level Agreement.
XML	Extensible Markup Language.
XSD	XML Schema Definition.
YAWL	Yet Another Workflow Language.

List of Symbols

D	Domain model for a set of security objectives.
QA	A set of QoS attributes qa which form a QoS model.
$Q^{\alpha}_{i,max}$	Maximum value of q^{α} in S_i.
$Q^{\alpha}_{i,min}$	Minimum value of q^{α} in S_i.
Q^{α}_{max}	Overall maximum value of q^{α}.
Q^{α}_{min}	Overall minimum value of q^{α}.
S_i	Service class i.
Ω	The solution set of a MOO problem.
Φ	Aggregation function for a QoS attribute.
Φ'	Aggregation function for a normalized QoS attribute.
$\mathbf{x}$	Solution vector.
$\mathcal{HR}$	Hypervolume Ratio.
$\mathcal{HV}$	Hypervolume.
$\mathcal{NF}$	Non-dominated Front.
$\mathcal{PF}^*$	Pareto Front of a MOO problem.
$\mathcal{P}$	A set of non-dominated but not necessarily Pareto optimal solution for a MOO problem.
$\mathcal{P}^*$	The set of Pareto optimal solution for a MOO problem.
$\mathcal{Q}$	Global Quality-of-Service vector of a service composition.
$\mathcal{Q}^{'}$	Global normalized Quality-of-Service vector of a service composition.
$\mathcal{Q}^{\alpha}$	Component α of the global Quality-of-Service vector.
μ^{α}_{i}	Average deviation for q^{α} in S_i.
σ^{α}_{i}	Standard deviation for q^{α} in S_i.
e_{ij}	Effectivity vector of service alternative s_{ij}.
e^{α}_{ij}	Component α of effectivity vector e_{ij}.
p_{rep}	The probability at which COMPRepair is called.
pf_i	Protection function i.
q'_{ij}	Normalized Quality-of-Service vector of service alternative s_{ij}.

q^{α}	Component α of Quality-of-Service vector q.
q_{ij}	Quality-of-Service vector of service alternative s_{ij}.
$q_{ij}^{\prime\alpha}$	Component α of q'_{ij}.
q_{ij}^{α}	Component α of q_{ij}.
qa	A single QoS attribute in a QoS model.
s_{ij}	Service alternative j of service class S_i.

Part I

Foundations

Chapter 1

Introduction

> *"A journey of a thousand miles begins with a single step."*
>
> – LAO TZU

IN THIS INITIAL CHAPTER the work at hand is motivated at first by briefly discussing the background of service composition and by identifying research challenges (Section 1.1). Subsequently, concrete research questions are deducted from these challenges and it is sketched how these are addressed (Section 1.2). Next the structure of this thesis is illustrated (Section 1.3). The chapter closes with a presentation of published results and how research has been developed over the course of time (Section 1.4).

1.1 Motivation

Nowadays, Service Oriented Computing (SOC) [PTD+07] is widely seen as a promising approach to implement business processes of any complexity. Single tasks (i. e. process steps) are assigned to external web services which perform these tasks at a certain Quality of Service (QoS) level. The result of such an assignment is called a service composition [BSD14]. Each service composition thus implements a business process at a certain global QoS level. However, in real-world applications there usually exist global QoS constraints such as maximum response time, price, etc. The assignment of services to tasks therefore is modeled as an optimization problem and tackled with QoS-aware service selection algorithms. A common approach for modeling the service selection problem is the Multidimensional Multiple-Choice Knapsack Problem (MMKP) which was made popular by YU et al. [YZL07]. Because MMKP is NP-hard, heuristic approaches such as Genetic Algorithms (GAs) are utilized to obtain near-optimal solutions in short time.

Security and compliance are two important issues in SOC since service compositions are dealing directly or indirectly with sensitive information (i. e. critical data which is processed by the composition). As such, data theft, information leakage and compliance violations might cause tremendous damage, both financially and to a business' reputation[1]. Compliance in the context of business process management has been studied thoroughly (cf., e. g., [ALS11; BBG+07; DGM+12; GK07; GV06; LMX07; MKL+09; SGN07; YMH+06]). To our very best knowledge however, compliance issues in service selection have not been covered so far. Regarding security issues in service composition, two major groups of approaches can be identified. The first group are QoS models (cf., e. g., [LJL+03; OEH02; CDE+05; CP09; JRM04; SCD+97]) which consider security either as a single QoS property or as several mutually independent properties. The second group consists of formal methods (cf., e. g., [BDF06; CFH06; POM08; SYT+10]) which provide facilities to express security requirements either globally or on the level of single tasks. These approaches are insufficient for at least three reasons:

1. Security objectives are subject to interdependence[2].
2. Some domain-specific applications require security objectives which are usually not covered by current QoS-models and formal methods such as *location privacy* for mobile applications.
3. The implementation of QoS properties (e. g., different implementations of the same encryption algorithm) might have a significant influence on a security objective.

Thus the question is how to determine the security of service compositions with regards to above aspects? Since formal methods and security metrics are designed with single security objectives in mind, a more flexible approach is required. One approach to tackle this challenge, first introduced by Wang and Wolf, is to utilize the notion of structural decomposition in order to identify basic measurable components for intangible security objectives. For each identified component the influence on a security objective is determined based on experts' opinion. Achievement of a security objective is consequently measured as utility value which thus enables comparison [WW97]. Hence the result is not a formal security model but a security assessment method. Given such a comparable metric

[1] In Germany for instance, the average damage caused by security breaches was $199 per lost or stolen data record in 2013 [Pon13].

[2] Wolf and Pfitzmann discussed the interdependent nature of security objectives and found that security objectives either *strengthen*, *weaken* or *implicate* each other [WP00].

for different security objectives, the question arises of "how much" security is necessary? SANDHU proposed the notion of "good enough security" which demands that security measures should be employed with the application in mind and that maximum security should not be aimed at if not explicitly requested by the user [San03]. Obviously, this concept is complementary to structural decomposition as it implies defining constraints on utility values which, again, represent security assessments of single security objectives. Therefore the author believes that these two concepts are essential key factors in designing an approach for assessing interdependent security in service composition.

1.2 Research Outline

The central question addressed in this thesis is how good enough secure service compositions can be determined considering interdependent security goals and compliance requirements. This leads to a number of research questions which need to be considered. Following is an overview of these questions and how they are addressed in this thesis.

Research Question 1 *What are the requirements for a method for service composition with respect to interdependent security and compliance?*

As a prerequisite to answering this question, the current state-of-the-art in service composition is being discussed from various perspectives in Chapter 3. This discussion forms the basis for deducting concrete requirements for a service composition method which considers interdependent security and compliance. Each requirement is being compared against related works in order to clearly identify research gaps.

Research Question 2 *How can interdependent security objectives be assessed in service composition?*

The basic approach for security assessment by WANG and WULF assumes that security goals are singletons with no mutual relationships [WW97]. Hence the focus of this question is how the approach of WANG and WULF can be extended to cover the structures discovered by WOLF and PFITZMANN (cf. [WP00]). This is achieved by first discussing interdependencies in security from a high-level perspective. Based on the results of this discussion, an approach is developed which allows for modeling interdependencies between security assessment functions. In the remainder of this thesis such functions will be referred to as *protection functions*.

It should be emphasized that this thesis does not propose new mathematics but employs the notion of structural decomposition to assess the security of service compositions with regards to the structures discovered by WOLF and PFITZMANN. SANDHU's notion of "good enough security" (cf. [San03]) is captured conceptually by the presented approach in terms of function constraints.

Research Question 3 *How can compliance be verified in service composition?*

As stated, compliance has not been yet considered in the context of service composition. Hence in order to answer this question and to propose an approach, firstly a thorough literature review on business process compliance is performed. The goal is to study which aspects of compliance can be verified by different classes of approaches. Reversely, this will reveal which compliance aspects have to be considered in service composition. Subsequently, this finding is utilized to develop an approach which allows for transforming these aspects of compliance into mathematical constraints. It is furthermore shown how this approach can be integrated with GAs. Since GAs are nowadays a commonly employed technique for selecting services, the focus was to develop an approach which can be employed in existing approaches for service composition with minimal efforts.

Research Question 4 *What are the impacts of considering interdependent security and compliance on service composition?*

Since "there ain't no such thing as a free lunch" it is essential to understand the benefits and associated costs of the methods proposed in this thesis. Hence all developed approaches are being extensively evaluated employing simulations in Chapter 7. The goal is to understand the impact of different problem sizes and numbers of protection functions on service selection in general and on the methods proposed in this thesis in particular. All results with respect to the approaches presented here are being compared against state-of-the-art GAs.

To answer these research questions, a variety of research methods are employed, namely *modeling*, *simulations* and *prototyping* [WH06]. This is in accordance with FRANK's conception that the variety of problems in information systems research can hardly be solved with a single method and thus a pluralistic approach is required [Fra06]. Hence this thesis follows the paradigm of construction science which is an established scientific method in German information systems research (cf., e. g., [Fra07; ÖBF+10]). Moreover, this thesis makes use of further, mainly mathematical methods which are:

- Mathematical Modeling,
- Linear Optimization with Multiple Objectives,
- Genetic Algorithms,
- and Decision Making.

Mathematical modeling is used e. g., in formulating the basic service selection problem. This basic formulation is extended to consider interdependent security objectives. Security objectives are modeled as **linear functions** following the notion of structural decomposition. As practitioners are typically dealing with more than one security objective, **multiple objectives** need to be considered simultaneously. Problems of this class however yield the feature that their solution sets might contain an infinite number of Pareto optimal solutions. As such heuristics are utilized, i. e. **GAs**, in order to approximate the set of Pareto optimal solutions. **Decision making** techniques are required in the process of formulating security objectives as functions, more precisely to estimate the influence of single QoS attributes on a security objective.

Some related topics had to be excluded in order to sharpen the focus on the main contributions. In particular, this thesis does not cover procedures for transforming verbal process descriptions to process models. Furthermore, model-driven transformations of process models into executable forms are also not covered but only sketched to give the reader an impression of a service composition's outcome. Finally, the developed prototype has not been evaluated in terms of usability. As has been stated, the main purpose of the prototype is to proof the applicability of the contributions presented in this thesis. Hence ethnographic studies or empirical evaluations by means of design science are beyond scope. For further information regarding these topics, the interested reader is kindly referred to the corresponding standard literature.

1.3 Thesis Structure

This thesis consists of nine chapters which are organized in four parts. **Part I** introduces essential foundations which are necessary for understanding the approaches presented in **Part II**. These approaches are subsequently being evaluated in **Part III**. Finally, **Part IV** concludes the thesis. Figure 1.1 illustrates the structure of this thesis and shows in particular dependencies between chapters of each part.

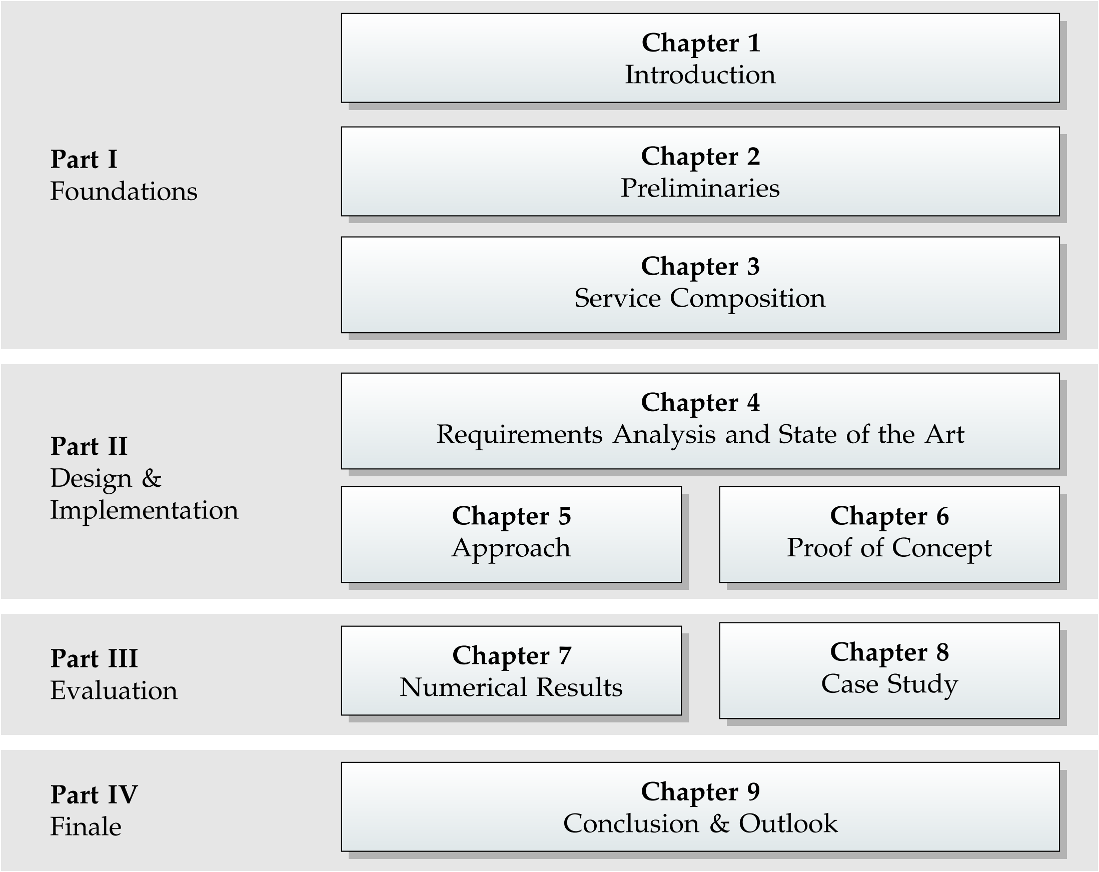


Figure 1.1: *Structure of this thesis*

Chapters ordered below each other indicate content dependencies while chapters which are placed next to each other indicate that the respective contents are in the broadest sense independent from each other. Following is a brief overview of the remaining chapters.

Chapter 2 introduces the conceptual and mathematical foundations this thesis is based upon. In particular, these are Business Process Management (BPM), SOC, interdependent security, compliance as well as models and solving approaches for mathematical optimization.

Chapter 3 provides an overview of service composition. It covers different composition topologies, the service selection problem, compliance issues and gives a brief overview about alternate composition approaches. The content provided in this chapter is essential for understanding the approaches presented in Chapter 5.

Chapter 4 presents a requirements analysis based on the discussions of the previous chapters. Elicited requirements are checked against state of the art approaches in service selection. The primary results of this chapter are thus clearly identified research gaps in service selection literature.

Chapter 5 covers the proposed approaches for addressing the research gaps identified in the previous chapter. A framework is presented for assessing the security of service compositions with regards to interdependent security objectives. Furthermore, an approach is introduced for considering compliance requirements in service selection with GAs. Latter is a drop-in solution which is illustrated by employing an existing GA.

Chapter 6 presents a graphical tool which has been developed in order to support users in applying the approaches presented in Chapter 5. The tool aims at first hand to be a proof of concept for the methods developed in this thesis. Thus it is not considered a mature software product, but a prototype for demonstration purposes.

Chapter 7 provides a thorough numerical analysis of the drop-in solution presented in Chapter 5 against related GAs. It shows the feasibility of the proposed approach in terms of algorithm runtime and consideration of compliance requirements. Furthermore, areas for further improvement are identified.

Chapter 8 reports an application of the framework for assessing interdependent security proposed in Chapter 5. The presented case study is from the EU FP7 project di.me[3].

Chapter 9 finally summarizes the main results of this thesis, discusses some open questions, shows possible directions for future research and concludes the work at hand.

1.4 Publications & Research Development

Parts of this thesis have been previously presented at international academic conferences and workshops and have also been published in peer-reviewed proceedings and international journals. This section gives an overview of the parts which have been previously published and how research has developed since publication.

The approach for considering interdependent security in service composition (cf. Section 5.2) has been initially published at the *28th Symposium on Applied Computing* (ACM SAC) [KK13a]. An extended version of this paper has been published in the Journal *Science of Computer Programming* [KFK15]. The content of Section 5.2 is a mostly unmodified reprint of parts of the latter. Some enhancements have been performed which include connecting the content of the respective article with sections from this thesis and adjusting the notation used in the article to the one employed here. An application of this approach in the EU FP7 project *di.me* has

[3] `http://www.dime-project.eu/`

been reported in a paper which was published at the *10th International Conference on Information Technology - New Generations* (ITNG) [KHB+13]. This publication formed the basis for Chapter 8.

The algorithm for compliance-aware service selection (cf. Section 5.3) has been published at the *11th International Conference on Service Oriented Computing* (ICSOC) [KK13b]. In its original version however, the algorithm did not perform any kind of initial feasibility check. Hence this step is not covered in the paper. The remaining parts of Section 5.3 are an almost unmodified reprint of this publication.

Parts of the prototype's user interface discussed in Chapter 6 have been published in a paper which was presented at the *5th International Conference on Online Communites and Social Computing* which was held as part of *HCI International 2013* [KBK13]. Particularly, the content of Section 6.2.2 on transforming optimization problems with subject to user preferences is based on this publication. The remaining parts of Chapter 6 are based on [KFK15].

Chapter 2

Preliminaries

> *"The beginning is the most important part of the work."*
>
> – PLATO

THIS CHAPTER FOCUSES on concepts, definitions and models which are essential for understanding the work at hand. At first an overview is given about Business Process Management (Section 2.1) and Service-Oriented Computing (Section 2.2) which are the conceptual foundations this thesis is built upon. What follows next is a discussion about interdependent security (Section 2.3) and Business Process Compliance (Section 2.4) which are the areas of research to which this thesis contributes. The contributions of this thesis heavily leverage optimization models and solving approaches which are discussed afterwards (Section 2.5). A short summary concludes this chapter (Section 2.6).

2.1 Business Process Management

The focus of this work is on service-oriented projections of business processes. All steps from defining over executing to analyzing business processes are summarized under the term business process management. The following sections provide an overview about the most important terms and definitions as well as the state of the art in implementing and executing business processes.

2.1.1 Overview

The outcome of each business activity is a product, either tangible or intangible. Examples include bread, insurances, software and haircuts. In order for these products to come into existence, a set of single *tasks* needs to be performed in a specific order. The order of these tasks is determined by a set of *conditions*. Each task is performed by a certain *resource*, i. e. a person, a machine or a group of persons and/or machines [AH02a]. For instance, in order to make a bread, it is mandatory to have the right ingredients at hand and mix them first. As soon as the dough takes a certain consistency, it is served to an oven. As soon as the bread reached a certain coloring, it is taken off the oven. If nothing went wrong over the course of the process, the result should be a fresh and tasty bread. While in a private household most tasks except the mixing might be performed by a human, in industrial production most if not all tasks are performed by machines with humans rarely performing tasks other than supervision and quality control. In contrast to processes which can be observed in nature (e. g. decomposition and fermentation) *business processes* are characterized by the circumstance that each single step needs to be *actively* performed by a resource. On the contrary, natural processes take place *passively* and can be controlled only conditionally. Based upon these observations, a business process is defined as follows[4]:

Definition 2.1 [BUSINESS PROCESS]. *A business process is a set of tasks which are ordered according to a set of conditions and where each task is actively performed by a certain resource in order achieve a certain outcome (i. e. product).*

The knowledge regarding business processes can be represented using a variety of graphical notations and models (cf. below). Such a graphical representation of a business process is what stipulates a *business process model*[5]:

Definition 2.2 [BUSINESS PROCESS MODEL]. *A business process model is a graphical representation of the knowledge regarding the execution of a business process.*

A business process describes an ideal case of execution, similar to PLATO'S theory of Ideas. In real-world, a process is executed multiple times in different contexts. Taking up on the introductory example about making bread, a business process correlates to the knowledge of making bread as it has been passed from master baker to master baker over the course of generations. In each period of time and at each geographical location however, the same business process is executed utilizing potentially different personnel, technical equipment and ingredients. Such a single execution of a business process in a certain context is called a *business*

[4] In the remainder of this thesis, the term *process* is used interchangeably.

[5] In the remainder of this thesis, the term *process model* is used interchangeably.

process instance[6] [Wes07]:

Definition 2.3 [BUSINESS PROCESS INSTANCE]. *A business process instance is a single execution of a business process in a certain context.*

In order to (semi-)automatize business processes employing Information Technology (IT), facilities are required to support all activities related to the lifecycle of business processes. This lifecycle consists of the following phases (cf. Figure 2.1) [AHW03; Wes07]:

- *Process design:* (Re-)Design of processes according to some process description. The outcome is a process model.
- *System configuration:* Transformation of a process model into a form which can either be executed itself or which empowers an information system to execute process instances based on a process model.
- *Process enactment:* Execution of process instances employing the configured system.
- *Diagnosis:* Analysis of operational systems to identify potential problems and to identify possibilities for improvement.

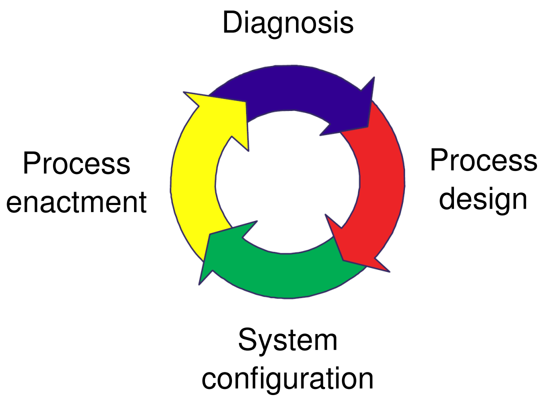


Figure 2.1: *The BPM lifecycle according to* VAN DER AALST *et al. (cf. [AHW03])*

These facilities are subsumed under the term BPM [AHW03; Wes07]. WESKE defined BPM as follows:

Definition 2.4 [BUSINESS PROCESS MANAGEMENT]. *"Business Process Management includes concepts, methods, and techniques to support the design, administration, configuration, enactment and analysis of business processes" [Wes07, p. 5].*

[6] In the remainder of this thesis, the term *process instance* is used interchangeably.

In the past, business processes were performed manually by enterprise personnel. With increasing utilization of IT in businesses, software tools were developed for supporting the enactment of business processes. A necessary requirement for this kind of tool support is a facility to represent business process in a form which can be interpreted and executed by software. For these software tools, the term Business Process Management System (BPMS) has been coined [Wes07]:

Definition 2.5 [Business Process Management System]. *Business Process Management System (BPMS) refers to software tools which support the enactment of business processes based on a process representation.*

Two other terms which are frequently encountered in the context of business processes are *workflow* and Workflow Management (WfM). According to the Workflow Management Coalition (WfMC), a workflow is a (partly) automatized business process which is characterized by an exchange of documents, information and tasks between participants [WfM99]. WfM is viewed as a synonym for workflow by the WfMC [WfM99]. This view is however not shared by the research community. A still widely adopted position is that WfM is a part of BPM but with focus on the utilization of software to support process execution [AHW03]. To put it in the words of van der Aalst et al.: "The focus of traditional workflow management (systems) is on the lower half of the BPM lifecycle" [AHW03, p. 5]. Hence in the remainder of this work, workflows will be regarded as (partly) automatized business processes and WfM as a subset of BPM.

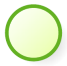
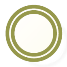
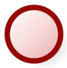
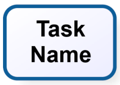
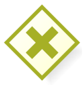
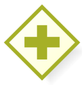
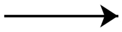
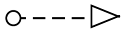
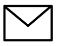
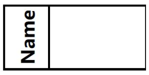
In the following, further properties of business processes will be introduced based upon a sample process for insurance claiming. This sample process is taken from [AH02a]. While in [AH02a] the authors represented the process as process diagram, it is being modeled utilizing the Business Process Modeling Notation (BPMN) [OMG11] here instead. The reason is that nowadays BPMN is widely utilized in the context of SOA and service composition while process diagrams are rarely seen. Alternative notations for business processes which are in use today are workflow nets (a class of Petri nets with certain business process-related properties) [Aal98], Event-driven Process Chains (EPCs) [Sch99] and the Multi-Perspective Enterprise Modeling (MEMO) framework [Fra98][7]. A formal approach which is frequently used to represent and analyze business processes is the π-calculus [PW05].

Table 2.1 provides an overview of the BPMN elements used in the sample

[7] An overview and a discussion of these notations (except of MEMO) is provided by e. g. [Wes07, Chapter 4].

process. This overview is by no means exhaustive but represents a minimal set of notation elements which are used throughout this thesis. For a complete overview, the interested reader is referred to [OMG11].

Table 2.1: *Overview of BPMN elements*

Element	Description	Notation
Start Event	As the name indicates, a start event marks an entry point of a process. It may be decorated with a marker such as a message (cf. below) to indicate different triggers. A process may have multiple start events.	
Intermediate Event	Intermediate events are placed between start and end events and affect the flow of a process. They may be decorated with markers such as messages (cf. below).	
End Event	In contrast to a start event, an end events marks one possible end of a process. It may be decorated with a marker such as a message (cf. below) to indicate different results. A process may have multiple end events.	
Task	A task represents an atomic activity in a process, i. e. a single process step which might not be further broken down.	
Exclusive Gateway	Exclusive gateways are utilized to indicate exclusive decision and merging points in a process flow.	
Parallel Gateway	Parallel gateways indicate forks and joins in a process model.	
Normal Flow	A normal flow indicates a path of sequence flow in a process model which is not subject to any condition.	
Message Flow	A message flow indicates sending and receiving of a message between two participants where each participant has to be represented by a pool (cf. below).	
Message	A message indicates communication between two participants which are represented by different pools (cf. below).	
Pool	A pool represents a participant in a collaborative process. It may have a content, i. e. a process model, or may be treated as a black box. Latter is usually the case when modeling non-deterministic participants such as customers.	

The sample process is triggered by an incoming insurance claim and consists of the following tasks [AH02a, p. 4]:

1. *Recording* the receipt of the claim;

2. Establishing the *type* of claim (for example, fire, motor vehicle, travel, professional);

3. Checking the client's *policy*, to confirm that it does in principle cover what has been claimed for;

4. Checking the *premium*, to confirm that payments are up to date;

5. *Rejection*, if task 3 or 4 has a negative result;

6. Producing a *rejection letter*;

7. Estimating the *amount to be paid*, based upon the claim details;

8. Appointment of an *assessor* to research the circumstances of the damage and to establish its value;

9. Consideration of *emergency measures* to limit further damage or relieve distress;

10. Provision of *emergency measures* if approved as part of task 8;

11. Establishment or revision of *amount to be paid* and offer to client;

12. Recording of client's *reaction*: acceptance or objection;

13. Assessment of *objection* and decision to revise (task 11) or to take legal proceedings (task 14);

14. Legal *proceedings*-,

15. *payment* of claim; and

16. *closure* of claim: filing.

The BPMN representation of this process is depicted in Figure 2.2. What is maybe striking most is that the tasks are not simply processed in the same order as they are enumerated above. The reason for this circumstance is that the chosen notation naturally dictates by its language elements how to express process properties such as conditions, dependencies and concurrencies. In the following, a deeper look will be taken at certain parts of the process description in order to introduce process related terms. As a visual support, the corresponding parts in Figure 2.2 will also be referenced (by enumerating the names of the involved elements in "quotation marks").

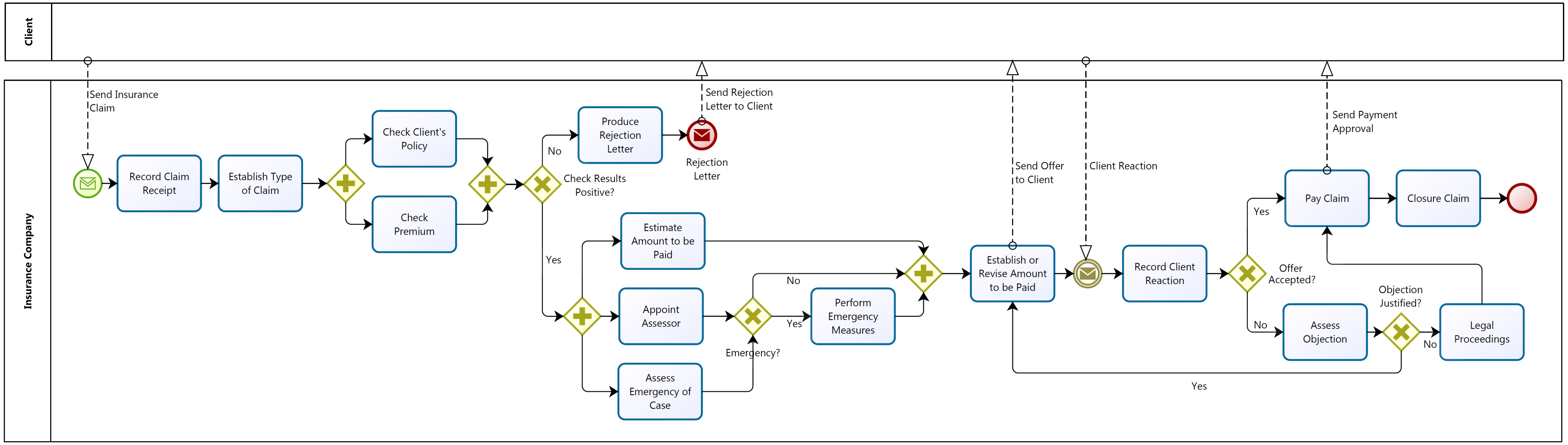


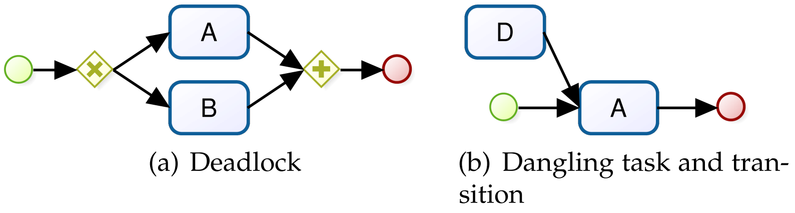
Figure 2.2: *The sample insurance claim process in BPMN*

Tasks 1 and 2 ("Record Claim Receipt" and "Establish Type of Claim") are performed in the same order as in the description. If two or more tasks need to be performed in a strict order like this, it is called a *sequence* [AH02a].

Tasks 3 and 4 ("Check Client's Policy" and "Check Premium") have no mutual dependencies and thus can be processed in *parallel*. However, the decision in task 5 ("Check Results Positive?") requires that tasks 3 and 4 are finished before. This is achieved via *synchronization* [AH02a].

The tasks 11, 12 and 13 ("Establish or Revise Amount to be Paid", "Record Client Reaction" and "Assess Objection") might be repetitive, depending on the client's reaction and the outcome of the assessment in case of an objection. Each repetition is called an *iteration* [AH02a].

As can be seen, business process models are useful to represent structured processes in a generic fashion. Improper modeling may however lead to structural errors such as deadlocks and dangling tasks and transitions. The process model in Figure 2.3(a) causes a deadlock since an XOR-split is being synchronized by an AND-join which obviously can never be fulfilled. In Figure 2.3(b), task "D" can never be performed and consequently the transition to task "A" neither as well. Hence both represent dangling elements with no effect on the process at all. While some errors such as dangling tasks and transitions may simply lead to redundant process parts, severe errors such as deadlocks may prevent business processes from ever being finished. Correctness criteria for business processes are summarized under the term *soundness* which is a combination of proper termination and non-redundancy of transitions. However, a discussion and formal presentation is out of scope of this thesis. Instead, the interested reader is referred to [Oan07; Wes07]. Since most BPMN editors today support verification of soundness criteria for given process models, it is assumed in this thesis, that considered process models are sound.



(a) Deadlock

(b) Dangling task and transition

Figure 2.3: *Some structural errors in process models*

2.1.2 Implementing and Executing Business Processes

As stated in Section 2.1.1, a BPMS is responsible for the enactment of business processes. The technical architecture of a BPMS has an impact on the way how processes are executed. As business processes are increasingly spanning several organizations and involve systems from more than one technological era, SOA (cf. Section 2.2.2) has become a first-choice architectural style for implementing and executing distributed business logic. Employing SOA, a business process is implemented as a composition of web services (cf. Section 2.2.2.3). This however raises the need to coordinate the execution of web services according to a given process model in e. g. BPMN. For this purposes, several languages have been proposed which will be shortly reviewed in the following.

Early approaches were the Web Services Flow Language (WSFL) [Ley01] by IBM and XLANG [Mic] by MICROSOFT. WSFL is a XML-based language which allows for defining service compositions as a flow model. Single process activities are represented as nodes in a graph model. The language allows for defining transition conditions for single nodes in order to capture conditional activity transitions of process models [Ley01]. XLANG is a proprietary scripting language by MICROSOFT which was developed for the company's BIZTALK server. The language has a block structure and allows for defining nested control flows [Mic].

Today, the de facto standard is the Business Process Execution Language (BPEL) [OAS07]. It is being maintained by the Organization for the Advancement of Structured Information Standards (OASIS) and can be considered as a merge between WSFL and XLANG. The language is XML-based and allows for nested expressions employing a block structure. Furthermore it allows to define links for expressing graph-like structures. The language has been thoroughly studied by academia due to its close link to BPMN[8] and several approaches for transforming process models into BPEL have been proposed. Examples include Petri nets [KM05], workflow nets [AL08], EPC [SI07] and UML [Gar03]. However, it was also found that a fully-fledged transformation from BPMN to BPEL is not possible. One major reason is that BPMN and BPEL represent different classes of languages, i. e. graph-based and block-structured [ODH+06]. Another reason is that BPMN provides a richer set of modeling constructs, some of which cannot be expressed with BPEL [RM06]. As such, fully automatic transformations can only be performed on reduced sets of BPMN elements such as presented in e. g. [ODH+06].

Another current alternative is Yet Another Workflow Language (YAWL) which was first introduced by VAN DER AALST and TER HOFSTEDE in 2002 [AH02b] and which is today maintained by the YAWL FOUNDATION. VAN DER AALST et al. observed that certain workflow patterns (e. g. those including multiple instances) cannot be realised directly in Petri nets but required additional specification ef-

[8] In fact a minor goal of BPMN was to serve as a visualization for XML-based business process execution languages such as BPEL [OMG11].

forts. As such YAWL was proposed as an evolution of Petri nets for the needs of workflow definition. The language itself provides graphical notations as well as modified Petri net semantics customized to the needs of workflow definition [AH02b]. YAWL is not as widely used as BPEL. However, in research YAWL is gaining more and more attention which is illustrated by the fact that in 2013 the first YAWL symposium took place[9].

Finally, the Web Modeling Language (WebML) was introduced in October 2000 and provides a graphical notation as well as a methodology for designing web applications. It provides different perspectives for specifying structure, presentation and personalization of web applications for users and user groups. WebML is patented and maintained by WEBRATIO [Webb]. Several approaches exist for transforming BPMN models to WebML (cf., e. g., [Bra06]) and WEBRATIO also provides a range of respective tools [Webb]. In terms of support tools and available publications however, WebML is not as widespread as BPEL and YAWL, neither in industry nor in academia.

Currently emerging alternatives for process implementation are the Interaction Flow Modeling Language (IFML) and the Jolie programming language.

IFML is based on WebML and is also developed by WEBRATIO. In contrast to WebML it was adopted by the Object Management Group (OMG) in March 2013 for publication as open standard and is, as of November 2014, in the beta phase. IFML is a generalization of WebML; where WebML focused on web applications, IFML is a general language for defining content, user interaction and control behavior for a wide range of applications such as Rich Internet Applications and desktop applications. IFML allows for binding its models to business logic in different formats such as BPMN[10] [Weba].

Jolie is an open source programming language developed especially for distributed service-oriented web applications [The13a]. Jolie's formal basis is the process calculus SOCK [GLG+06] and the syntax has certain similarities with C. Listing 2.1 shows an example sending the message "Hello, world!" to operation "println" of service "Console". This demonstrates the close syntactical connection between Jolie and the paradigm of service-orientation (cf. Section 2.2). The applicability of Jolie for implementing business processes has been demonstrated (cf., e. g., [Mon13]).

[9] `http://ceur-ws.org/Vol-982/`

[10] A thorough example of an online bookstore scenario illustrating the connection of IFML with Unified Modeling Language (UML) and BPMN models can be found at: `http://www.ifml.org/wp-content/uploads/IFML-Bookstore-Example.pdf`

Listing 2.1: *"Hello, world!" example in Jolie*

```
include "console.iol"

main
{
    println@Console( "Hello, world!" )()
}
```

2.2 Service-Oriented Computing

SOC refers to a pool of complementary concepts, the most important one being *service orientation*. Service orientation follows the paradigm of *Separation of Concerns* as discussed by Dijkstra [Dij82] and aims at building applications based on "building blocks" of application logic. Here, each building block realizes a certain task of almost any complexity such as validation of email addresses, transformation of data between certain formats and implementation of a sales process. These building blocks are called *services*. As the service concept plays a fundamental role in SOC, it is being discussed first in the following. The results of this inquiry are then utilized to define another important term in this context, namely *e-service*. With this conceptual foundation being laid, the term SOA and its buildings blocks are introduced before discussing the most recent incarnation of SOC, namely cloud computing.

2.2.1 Services and E-Services

Today the service sector is a vital part of each developed country's economy. For instance, in Germany the contribution of the service sector to the Gross Domestic Product (GDP) increased from 57% in 1980 to 71% in 2010[11]. The upcoming of Information and Communication Technology (ICT) lead to a new kind of services such as e-mail, e-commerce, video conferencing, etc. These services are usually called *electronic services* (e-services). In 2013, e-services made up 4.67% of Germany's GDP[12]. These figures illustrate the importance of services in today's economies. But what exactly is a service?

In economics, output is traditionally classified into goods and services where a service is defined as a product which "perishes in the very instant of its produc-

[11] http://data.worldbank.org/indicator/NV.SRV.TETC.ZS

[12] https://www.destatis.de/EN/PressServices/Press/pr/2014/09/PE14_306_811.html

tion"[13] or which is simply "immaterial"[14]. This classical view dates back to the late 18th century and has been utilized for about three centuries with more or less slight modifications. In fact, a publicly little noticed debate regarding the validity of this view took place behind the scenes[15]. In the second half of the 20th century, the classic service definition has been argued by HILL who pointed out that certain services (namely those affecting goods) have some similar features with goods while others (namely those affecting persons) have vastly different traits from goods [Hil77]. Based upon this observation, he found in his later work the classical dichotomy between goods and services to be insufficient as well [Hil99].

HILL argues that an important feature which constitutes a service is an economic relationship between producers and consumers based upon mutual agreement [Hil77; Hil99]. Consequently, he defined services not as a special category of goods, but as "a change in the condition of a person, or of a good belonging to some economic unit, which is brought about as the result of the activity of some other economic unit, with the prior agreement of the former person or economic unit" [Hil77, p. 318]. This definition was later extended by GADREY for time, demand and ownership aspects in order to extend the applicability of the service term to phenomenons which were not or only partly covered by HILLs original definition [Gad00].

From an information systems perspective, it is possible to simplify this view as the service term is employed in the context of IT artifacts. As such, a service definition as wide as proposed by HILL and GADREY is not necessary. This observation is reflected in proposed service definitions in the context of SOC by the fact that they usually emphasize technical aspects. Frequently the terms *activity*, *entity*, *component* and *software* are being used to describe services, along with different properties (cf., e. g., [PG03; W3C; KBS04; Erl07; PTD+07; SCK11; YCY12; WKI+12]).

While from a pure technical point of view this may be a sufficient characterization, from an economical it is not since the economical relationship between the parties stays unclear. For instance, KRAFZIG et al. define a service as "some meaningful activity that a computer program performs on request of another computer program" [KBS04, p. 14]. While surely sufficient from a technical perspective, this definition does not state the economical relationship between these two computer programs. E. g., it remains unclear if these two computer programs need some kind of former agreement and if this activity is able to exist on its own by means of a

[13] Cf. ADAM SMITH (1776), "The Wealth of Nations", Book II, Chapter III.

[14] Cf. JEAN-BAPTISTE SAY (1803), "A Treatise on Political Economy", Book I, Chapter XII.

[15] An overview of the historical development of the service term can be found in [DG92].

pay-per-use service. Even worse, this definition stresses that an activity needs to be "meaningful" in order to be regarded as a service. Now the interpretation of what is meaningful or not certainly lies in the eye of the beholder. A service for checking credit card numbers might be meaningful for most if not all service consumers. On the contrary, a trivial service which simply adds two integers and delivers the result might not be generally meaningful. As such, this definition requires a further clarification about the meaning of "meaningful" which consequently adds even more fuzz to the service term. Thus, ironically, the adjective "meaningful" leads to the point that the service definition of KRAFZIG et al. becomes quite meaningless.

HILL points out that one essential feature of a service is its transactability, i. e., the possibility to be carried out by another economic unit with the benefit being able to materialize on the consumer. As examples HILL counts housework and painting activities which might be performed by oneself or by someone else on behalf of oneself. Other activities such as eating and swimming lack transactability, i. e. nobody can eat or swim for the benefit of someone else [Hil77]. Applied to information systems, activities such as performing computations and storing data can be regarded as transactable in the sense that they can be performed remote by one entity on behalf of another. On the contrary, other activities such as allocating memory or loading values into CPU registers cannot be performed remote by one entity for another. Given this property, services are defined as follows (based on [Hil77]):

Definition 2.6 [SERVICE]. *A service is a transactable activity which is performed by an economic unit **A** (service provider) on request by another economic unit **B** (service consumer) based upon prior agreement. **A** and **B** might be identical.*

An obvious advantage of this definition is that it neither reduces the service phenomenon to some technical terms nor requires fuzzy adjectives such as meaningful. The decision if an activity is regarded as a service or not is solely reduced to the activity's potential of being provisioned and consumed by different entities. The drawback of this definition is of course that it is not generally applicable to all phenomenons which one would intuitively call a service. In the scope of this thesis however this is not relevant since the explicit goal was to establish a service definition which can be applied to information systems as general as possible. In fact, GADREY remarks, that it is impossible to formulate a service definition which can be applied to all phenomenons which one would view as a service [Gad00].

This leads to the question of how to define e-services. The common denominator of these services is that they are served via electronic networks which is regarded here as the main difference to traditional services[16]. Utilizing definition 2.6, electronic services can thus be defined as follows (based on [RK02]):

Definition 2.7 [E-Service]. *An electronic service (e-service) is a service which is provided via electronic networks.*

2.2.2 Service-Oriented Architecture

Having discussed the service term, the concept of SOA can now be introduced. Firstly this concept is being motivated from a historical and technical point of view before giving an overview of the concept itself. Afterwards, web services as a technical enabler for SOA are being introduced and their QoS properties discussed. The section closes with a discussion about the practicability of some properties that the concept of SOA is generally associated with by academia.

2.2.2.1 Motivation

> *"We [at Lyons] dreamed of some wonderful machine where all you would need to do would be to feed in paper and press buttons and get all the answers you wanted; it was all very naïve"*
>
> – John Simmons, former executive director at Lyons [CAH+97, p. 363]

The role of IT in enterprises has changed considerably from its initial applications in the first half of the 20th century. In the beginning, IT was expected to provide computation power and storage to support single business tasks. The first business computer was the Lyons Electronic Office (LEO) which was installed at Lyons company in 1951 and performed tasks such as valuation, payroll and inventory management [CAH+97]. During the second half of the 20th century however, the environment for most enterprises started to change with the globalization of national economies. Competition became increasingly tough due to lower customer loyalty and increased product complexity on the one hand as well as time and cost pressure on the other hand. Mergers, acquisitions and buy-outs of companies lead

[16] In fact, both types of services differentiate in other aspects as well, but since these differences have no impact on the content of this thesis, they will not be further discussed here. For a thorough discussion about this topic, the interested reader is referred to [Bla09].

to the necessity to integrate formerly separate and most of the time different IT systems[17]. In this regard, the notion of process-orientation became increasingly important for enterprises, i. e. cross-divisional and (semi-)automatic handling of business transactions. This allowed for quick adaptation to environmental changes such as new laws, changed product portfolios, etc. In this situation, the role of IT changed as well, namely from providing basic computing capabilities to supporting processes in distributed environments [Erl05; KBS04].

First technical approaches for distributed computing were Remote Procedure Call (RPC), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA) and later web services (cf. Section 2.2.2.3) [KBS04]. While these approaches enabled integration of distributed and diverse IT systems, the sole usage of these technologies alone leads to tightly coupled systems. As an example consider a typical scenario from a retailing company utilizing different software systems for e-shopping, inventory management, supply chain management, accounting, payroll, customer relationship management, document management and e-mail (cf. Figure 2.4). A fully meshed integration via point-to-point connections thus requires a total of

$$\frac{n(n-1)}{2} \tag{2.1}$$

connections. In this simple example this means that a total of 28 connections need to be specified and implemented. In case that one application is exchanged (due to software upgrades, licensing issues, etc.) $n - 1$ connections need to be reimplemented. In the face of ever-changing requirements on enterprise IT, a more flexible architectural style was required [KBS04].

One famous and frequently employed paradigm to face the changed role of enterprise IT is Enterprise Application Integration (EAI). EAI refers to a set of methods and middleware for integrating different physically or logically distributed applications. Integration is realized by means of enabling data exchange between applications while avoiding point-to-point integration. Widely used methods for data exchange are the following [HW03]:

- **File Transfer:** Applications exchange data by writing to and reading from a file. It is mandatory to agree in advance on name, location, format, timings for read an write operations as well as responsibility for deletion of the file.
- **Shared Database:** Applications share an identical database schema which is located in a single database. As such, data changes take effect on all involved

[17] E. g., [Erl05, Chapter 2] provides some well explained and typical case studies.

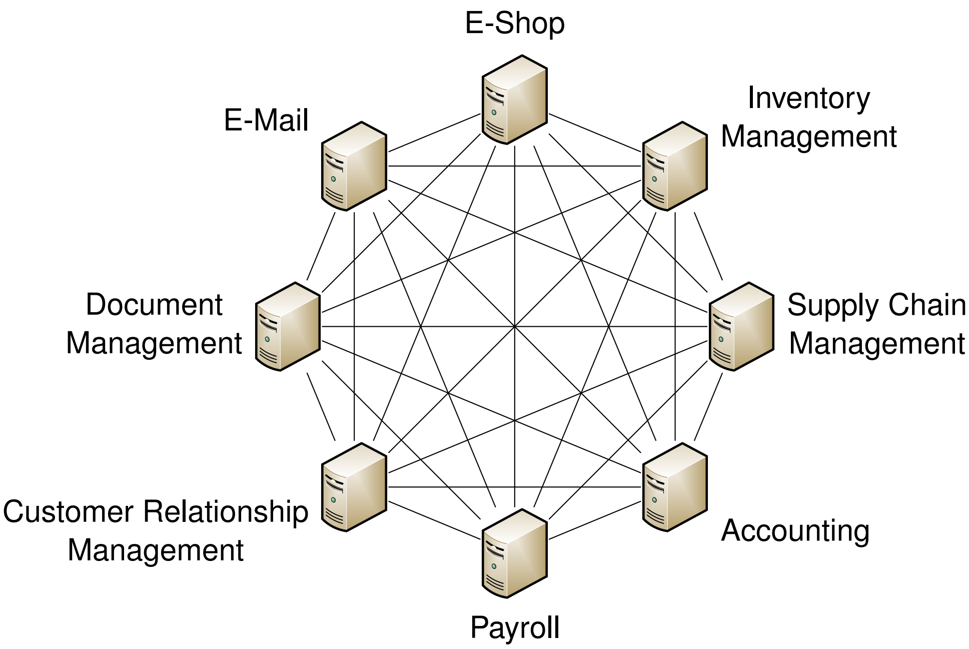


Figure 2.4: *Example scenario for a fully meshed point-to-point integration*

applications without requiring separate data transfer.

- **Remote Procedure Invocation:** Applications provide parts of their functionality for real-time and synchronous remote access.
- **Messaging:** Applications push messages to message channels while other applications read the messages from these channels. In order to enable this form of asynchronous communication, the applications must agree on a channel as well as message format.

Among these alternatives, messaging is the most widely used integration method today. Integration is performed utilizing Message-Oriented Middleware (MOM) solutions such as ActiveMQ and MSMQ. While EAI enables distributed applications to communicate, the major drawback of this approach is that it generally also leads to tightly coupled systems due to the technical nature of integration. In case of change requests, reconfiguration again needs to be performed on a fine-grained level such as files, database tables, remote procedure interfaces and message queues [HW03; KBS04].

The paradigm of SOA has been discussed since the late 1990s. The term itself was coined by the Gartner Group which described its utilization in a two-part study [SN96a; SN96b]. It was however not until the upcoming of web service standards (cf. Section 2.2.2.3) that the paradigm received a broad interest. An important factor was the observation that SOA and BPM are conceptually closely related. SOA enabled architectural flexibility which was a requirement for dynamic business processes [Bur00; LRS02]. What followed was an explosion of conferences, publications, software tools and experts for this new paradigm. In the mid-2000s the SOA hype reached its peak which was coined by books such as "Service Orient or Be Doomed!" by BLOOMBERG and SCHMELZER [BS06].

In the afterward, the time from the upcoming of the SOA paradigm until its peak can be viewed as a revival of the machine supremacy thoughts of the first half of the 20th century; early computers such as the ENIAC and the Harvard Mark I had caused much excitement due to their previously unknown computing capability and were called "electro-mechanical brains" and similar exaggerated names by media[18]. This reflected the public view of that time which was driven by the misconception that computers were omnipotent. History repeated, this time with focus on SOA. Fatal optimists like BLOOMBERG and SCHMELZER expected to solve all IT-related problems and to gain an increase in enterprise's productivity and innovation of unprecedented degree by applying SOA [BS06]. Soon however it was realized that SOA, as anything else, comes at a price. The flexibility enabled by SOA lead to technically complex solutions which in many cases caused a disproportional increase in IT costs. Furthermore, unawareness about the prerequisites of a successful SOA implementation such as a close alignment between business and IT lead to an increasing rate of failing SOA projects[19]. Finally, the SOA paradigm was even declared dead [Man09]. It was not until the upcoming of Cloud Computing (cf. Section 2.2.3) that SOA experienced a revival as a facility to integrate internal IT systems with Commercial Off-The-Shelf (COTS) components supplied by external providers.

2.2.2.2 Overview

At its core SOA advocates to encapsulate and provide IT resources as services which enables *Two-Level Programming* [DK75]. Services are intended to act as building blocks to develop software systems in *in the large*. In the context of BPM this means building service-oriented workflows by mapping single process steps to services. On the contrary, services are developed *in the small* encapsulating e. g. legacy applications running on diverse operational resources. Figure 2.5 illustrates the interplay between these different layers in a SOA.

Each service has a description which contains information regarding its func-

[18] The titles of two articles from Life Magazine are striking examples: "Overseas Air Lines Rely on Magic Brain" (August 16, 1937, p. 45) and "The Great Electro-Mechanical Brain" (January 14, 1946, pp. 73-74).

[19] A survey conducted by the BURTON GROUP on 20 companies in 2008 reports that only 20% of the projects were considered a success [McK08].

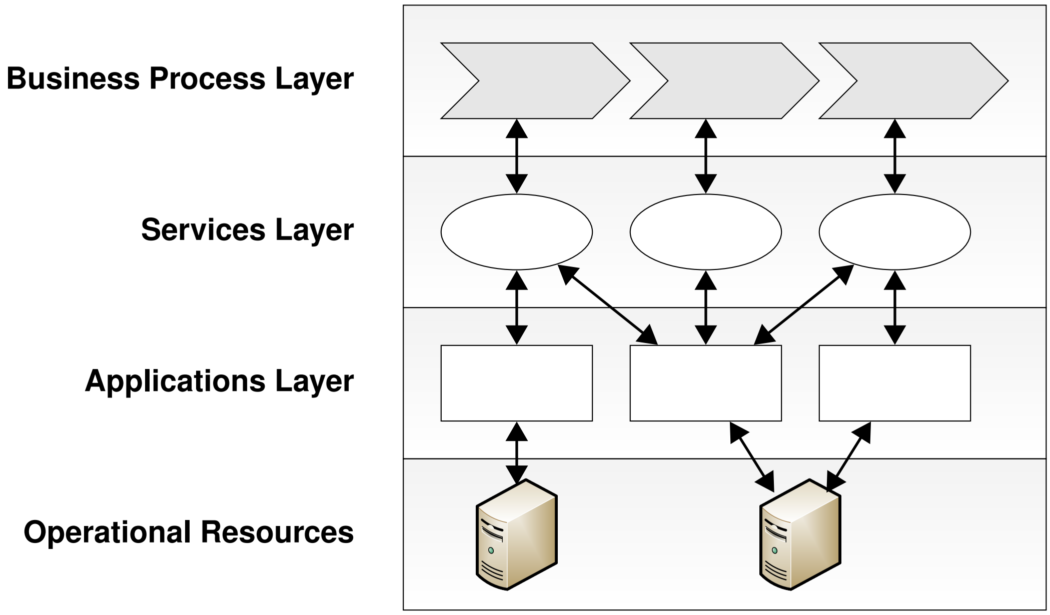


Figure 2.5: *SOA layers*

tional (e. g. input and output parameters) and non-functional properties (e. g. QoS constraints). Services may but do not have to be under the control of different domains. In contrast to approaches such as EAI, SOA does not prescribe technical solutions but provides architectural concepts to tackle integration issues [KBS04; Erl05; OAS06]. This leads to the question what a *software architecture*[20] is. In fact, this term is maybe even more ambiguous than the service term and plenty of definitions can be found in the literature[21]. Here, the term architecture will be defined as follows (based on [ISO11]):

Definition 2.8 [Software Architecture]. *Software architecture describes the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.*

Utilizing this definition of the architecture term, SOA can be defined as follows:

Definition 2.9 [Service-Oriented Architecture]. *A Service-oriented Architecture is a technology neutral architectural style to organize IT resources as services and a guiding idea to develop applications utilizing services as building blocks. These services may be under the control of different domains.*

[20] In the remainder of this thesis, the term *architecture* will be used interchangeably.

[21] A vast collection of definitions can be found e. g. here: `http://www.sei.cmu.edu/architecture/start/glossary/index.cfm`

This definition exhibits obvious similarities with component-based software engineering (cf., e. g., [HC01]). In fact, services can be viewed as a special kind of component. Where components in the traditional sense are viewed as static building blocks that can be plugged together, services are components which provide their functionality only upon request and do not require tight coupling with the application being developed.

In a SOA, a total of three roles is differentiated [Bur00; Erl05; KBS04]:

- *Service Provider:* The service provider serves one or more services. For each service, the provider publishes a service description, providing information regarding functional and non-functional properties, in a service registry.
- *Service Registry:* Similar to a phone book a service registry acts as a central instance which can be used to search and find services. It provides the service description to interested service consumers.
- *Service Consumer:* A service consumer is an entity which is seeking a service. In order to find a suiting service offer, the service consumer consults a service registry.

A typical sequence from publishing a service description to service consumption is as follows. A service provider publishes a service description at a public service registry. An interested service consumer consults the service registry and searches for a service suiting its needs. As soon as the consumer found one, the registry sends the consumer the service description which among others includes the address of the provider. The service consumer then binds this service into its SOA, either statically at development time or dynamically at runtime. Finally at each request of the consumer, the provider serves the service to the consumer. Figure 2.6 illustrates this sequence.

As stated, SOA is not a technology but in the first hand a paradigm. Thus there is not "the" implementation of a SOA but different implementations yield certain characteristics. While some characteristics are a matter of discussion, there is consensus on the following [Bal04; Erl05; MSD]:

- **Autonomy:** The encapsulated application logic is under the sole hegemony of the service. Changes made to one service have no effect on another. In a broader sense this also includes as much control over its runtime environment as possible by e. g. reducing shared access to service resources and a maximum physical isolation. An outcome of this autonomy is that the runtime behavior of each service becomes more predictable. For services which are composed of other services this means that the maximum level

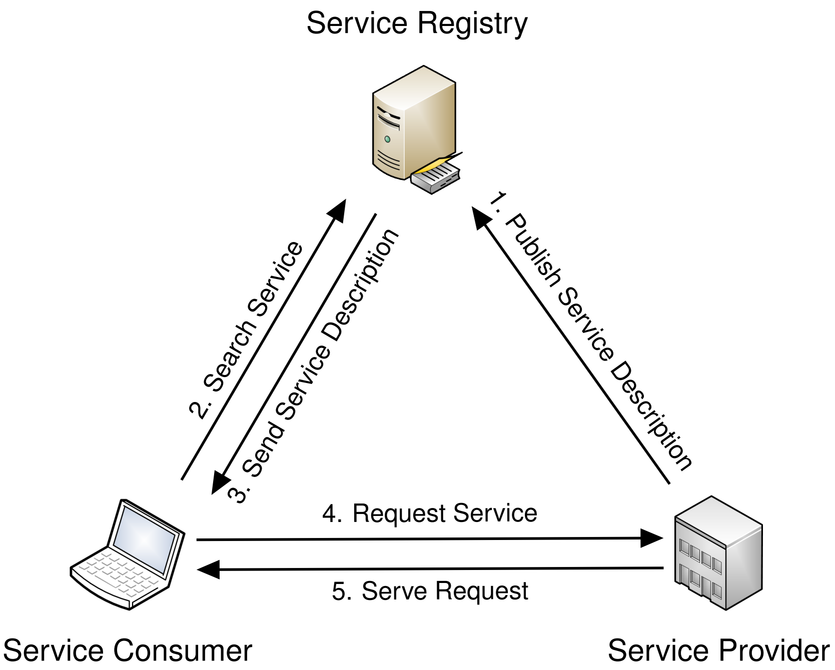


Figure 2.6: *Roles and their interaction in a SOA (based on [Bur00])*

of autonomy achievable is determined by the autonomy of the services it is composed of.

- **Loose Coupling:** There is no dependence between applications and services. In consequence this means that applications can be changed without affecting the services it is composed of and vice versa. The latter might not be as obvious, but remember that a service is accompanied by a service description which defines *what* task it solves. *How* the task is solved is another question and should have no effect on the defined task. A storage service which does not store any data it receives obviously violates its description. However, if the service is storing data on a hard disk using the NTFS or ext3 file format is usually not a matter of interest for the service consumer.

- **Reusability:** Services can be reused in different contexts. While this is a almost trivial requirement for services which perform basic activities such as storage and computations, it becomes harder for services delivering complex activities such as seat reservation in a flight booking application. For service design this means that services need to be agnostic to a) the applications they are used in and b) the technical environment they are executed in. In case of the mentioned seat reservation service, a well-designed solution would be a service which is neither bounded to a certain airline nor to a certain travel information system such as AMADEUS.

- **Composability:** Services can be composed without any knowledge about technical details of the underlying implementation. Thus a SOA allows for programming in the large where single services can be replaced with little effort to fulfill change requirements. From a conceptual point of view this requires that services are the result of a systematic decomposition of an application. The number of special services to address technical issues should be kept to a minimum to increase their reusability.

Services in a SOA can be classified according to certain properties. A widely used classification proposed by PAPAZOGLOU and GEORGAKOPOULOS differentiates between basic, composite and managed services. The employed criterion for classification is the role that a service plays in a SOA or a composition. The latter part causes conceptual inaccuracies as a composite service which is used in another composite service is suddenly considered as a basic service [PG03; PTD+07]. Another classification which was proposed by KRAFZIG et al., differentiates services only by their role in a SOA [KBS04]. The resulting classification scheme is thus more consistent and creates less confusion. In total, KRAFZIG et al. differentiate between four types of services [KBS04]:

- *Basic services:* From a functional point of view, these services form the smallest building block in a SOA. They are willingly kept simple and implement only simple data processing and/or logical steps in a process. Basic services are stateless and highly reusable.
- *Intermediary services:* Intermediary services are intended to bridge technical inconsistencies or architectural design gaps. They are stateless and usually yield low reusability as to their specific role.
- *Process centric services:* This type represents service-oriented representations of business processes. Accordingly, these services are usually highly complex and stateful. Reusability of process centric services is usually low.
- *Public enterprise services:* While the other types of services are usually employed internally, this type of service is made publicly available. Accordingly this type of service acts as a facility for cross-enterprise integration. Statefulness and complexity are depending on the task the service is intended to deliver. Usually these services exhibit a high degree of reusability. Besides of a task-specific implementation, these services require additional functionalities for e. g. billing, security and reliability.

As can be seen, SOA is at first abstract and needs to be fleshed out by utilizing concrete technologies. Nowadays frequently used are web services for implementing the building blocks and BPEL for service composition (cf. Section 2.2.2.3).

2.2.2.3 Web Services and Related Standards

The original Internet, also called Web 1.0 or "content web", was dominated by static content and participants could do little more than viewing and linking contents. With the transition to the web 2.0 (the so-called "service web"), the concept of web services was created as a facility to provide functionality over the Internet [Kno03]. Today web services are omnipresent and one of the most frequently used communication methods for applications and devices. Already in 2008 Amazon reported that the Amazon Web Services (AWS) were consuming more bandwidth than all retail sites[22]. In the consumer field, web services are used to e. g. integrate data from different sources such as mobile apps and social networks to enhance user profiling in order to customize marketing [KR11]. In the business world, web services gained a lot of attraction in the context of cloud computing which enables organizations to rent infrastructures, platforms and software on demand [MLB+11].

The web service term is not used uniformly and in fact describes at least two groups of approaches (cf. below). As such, definitions vary as well. Some definitions solely focus on technical aspects, i. e. that web services are components delivering functionality over a network such as the Internet [PG03; SCK11; YCY12]. As discussed in Section 2.2.1, this excludes the economical aspect and is thus an incomplete view. Other definitions imply the utilization of certain standards with web services [W3C04b; YCY12]. This however excludes alternate web service implementations and is thus too limited. Let us first examine existing groups of web services and analyze their features before formulating a definition.

Two major groups of web services can be distinguished:

- Web services which utilize standards based on Extensible Markup Language (XML) for interface description and communication [KBS04]. These are also called XML web services.
- Web services following an architectural style known as Representational State Transfer (REST) [Fie00]. These are also called RESTful web services.

Web services of these major groups exhibit the following features [KBS04; Fie00]:

1. **Unique Identifier:** A web service is uniquely identified. In case of XML web services, this identifier is typically an Uniform Resource Locator (URL). In case of RESTful web services, an URL identifier is mandatory.

[22] `http://aws.typepad.com/aws/2008/05/lots-of-bits.html`

2. **Interface description:** XML web services utilize the Web Services Description Language (WSDL) standard (cf. below) to provide an overview about all operations the web service supports, including input and output parameters. In case of RESTful web services, interface description is provided by the HTTP standard [FGM+99] as available operations are solely HTTP commands such as GET, POST, PUT and DELETE.

3. **Provision via the Internet:** Both XML and RESTful web services are provided via the Internet.

4. **Interaction via message exchange:** Web services interact with service consumers by exchanging messages. XML web services typically utilize SOAP (cf. below) for this task. In case of RESTful web services, service requests are simple HTTP request messages. The type of the response message depends on the requested operation and might be either HTML, JSON or XML.

With these groups and features of web services in mind, it can now be tried to capture these in a definition. Ideally, a definition should also include the economical aspects of the service term while allowing for alternate implementations. Thus Definition 2.7 for e-services as presented in Section 2.2.1 is being utilized and extended for the distinguishing features of web services identified above:

Definition 2.10 [Web Service]. *A web service is an e-service which is uniquely identified, has an interface description and is provided via the Internet. Interaction with other entities takes place via message exchange.*

In the scope of this thesis, the focus will be on XML web services. Thus, in the following relevant standards for building a SOA will be presented which are WSDL, SOAP and Universal Description, Discovery and Integration (UDDI) [LRS02]. Finally, a brief overview of BPEL as the de facto standard for service composition will be given.

WSDL

WSDL is a XML-based standard which is being maintained by the World Wide Web Consortium (W3C). Its first version has been published in September 2000 and was developed in cooperation between IBM, Microsoft and Ariba. At first, it was a joint effort to consolidate service description concepts found in earlier proposals. This initial release has been re-released in a formalized version 1.1 in March 2001 [CCM+01]. The latest version is 2.0 and became a W3C recommendation in June

2007 [W3C07b]. Here the focus will be on the previous version 1.1 since it is still more widely in use than 2.0.

Figure 2.7 shows the basic structure of a WSDL document. The root element is always named `definitions`. The child elements are divided in a so-called abstract and a concrete part. The abstract part has the child elements `types`, `messages` and `portType`. WSDL supports the basic data types defined in XML Schema Definition (XSD) such as `xs:string` and `xs:integer`. If there is further need for complex data types such as a customer data record, these can be defined in the `types` element. These complex data types along with the basic types of XSD can be used in the `messages` element to define different message formats for web service interactions. Finally, the `portType` element contains all `operations` the web service offers. WSDL supports different transmission primitives for operations which are: One-way, request-response, solicit-response and notification. Depending on the transmission primitive of each operation, the message types defined in the `messages` element can be assigned to each operation as input, output and fault message type [CCM+01].

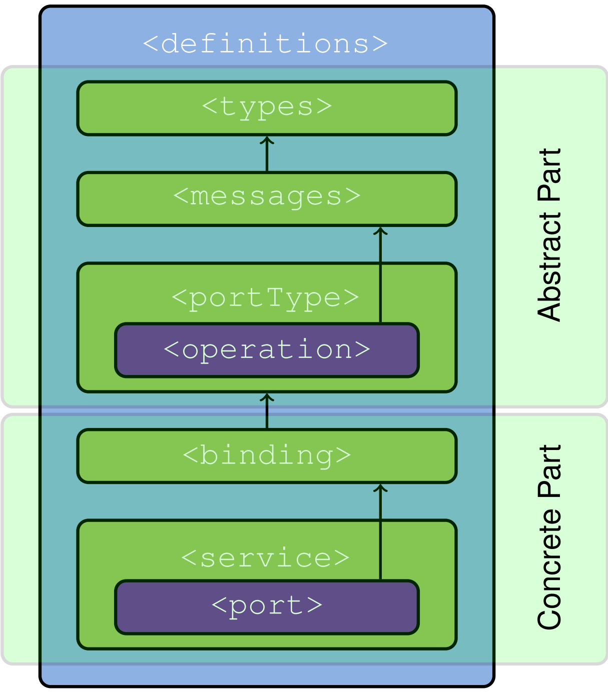


Figure 2.7: *Structure of a WSDL document*

The concrete part of a WSDL document contains the elements `binding` and `service`. The `binding` element allows for specifying the protocol as well as the message format for an `operation` defined in the `portType` element. This message format is not to be confused with the ones defined in the `messages` element. It refers to the concrete messages exchanged between endpoints utilizing the specified protocol. The abstract messages defined in the `messages` element will be transformed into these concrete messages during service invocations. An operation can have more than one binding (e. g. for different versions of the specified protocol). Finally, the service element is used to group one or more port elements. A port element assigns an address to a binding and thus specifies an endpoint of the service [CCM+01].

SOAP

SOAP is a protocol for exchanging XML-based messages. It was developed by MICROSOFT in 1998 but was later standardized by the W3C which is still the maintainer. The latest version is 1.2 which became a W3C recommendation in April 2007 [W3C07a]. Originally SOAP was an abbreviation for "Simple Object Access Protocol" [BEK+00] but this has been dropped since version 1.2 [W3C07a].

The basic form of a SOAP message is depicted in Figure 2.8. Messages are always enclosed by an envelope. This envelope has two elements, namely a header and a body. The optional header element contains meta information regarding the communication such as expiration dates, utilized encryption algorithms, etc. The mandatory body element contains the actual payload. In case of a XML web service described with a WSDL document, the body contains a message of the type associated with the invoked web service operation. SOAP can be bound to several protocols such as Transmission Control Protocol (TCP) and Hypertext Transfer Protocol (HTTP) [W3C07a]. Especially the binding to HTTP is used frequently.

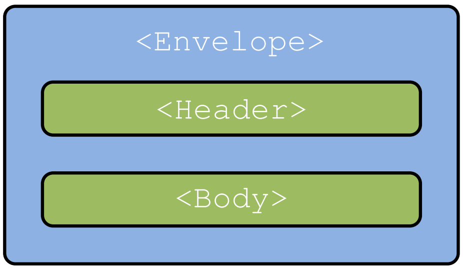


Figure 2.8: *Structure of a SOAP message*

UDDI

The idea behind UDDI was to act as a web-based registry for web services, similar to a phone book. It was actively supported by MICROSOFT, IBM and SAP which also hosted individual UDDI registries. UDDI was later standardized by the OASIS. The last published version was 3.0.2 which was made available on October 19th, 2004 [BCC+04]. On January 12th, 2006 however, MICROSOFT, IBM and SAP stopped operations of their UDDI registries [SAP05].

A UDDI registry contains information about web services and their providers in form of a XML document. This information is either published by the providers themselves or on behalf of them. Interested service consumers can search such registries for information about services they are interested in. In case that a service consumer found a service which suits her needs, the UDDI registry provides the consumer with a WSDL document of the service. For communication with a UDDI registry, the standard suggests the utilization of SOAP [BCC+04].

Utilizing these standards, a SOA can be implemented as depicted in Figure 2.9. It should be noted that this represents an ideal view which is not found in real-world SOA implementations (cf. Section 2.2.2.5).

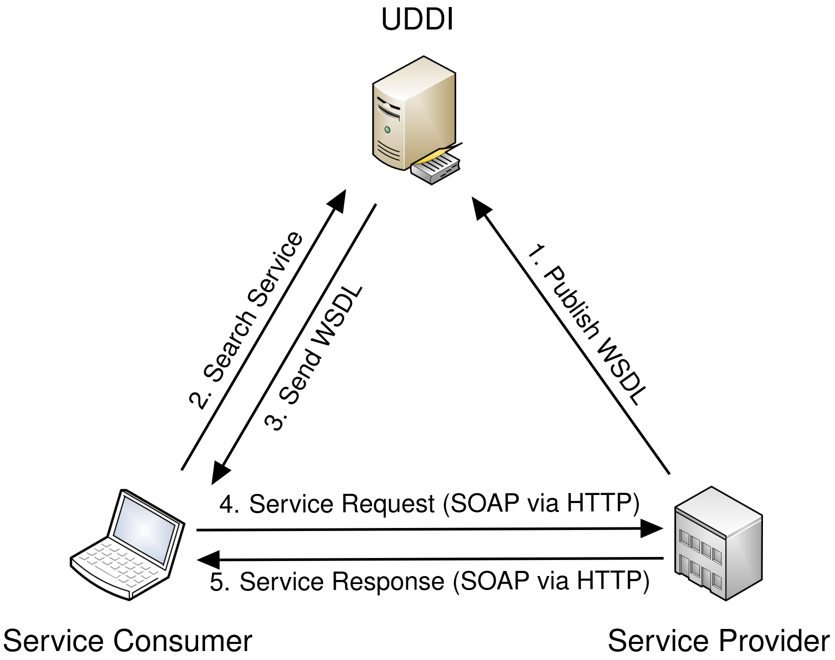


Figure 2.9: *Realization of a SOA with web services*

BPEL

BPEL is a XML-based language for web service orchestration (cf. Section 3.1). The structure of a BPEL process can be seen in Figure 2.10.

Each process definition must have a root element `process`. This element has two mandatory attributes `name` and `targetNamespace`. The `name` attribute is a simple textual representation of the process name while `targetNamespace` defines the context of the process [OAS07].

The element `partnerLinks` is used to group one or more `partnerLink` elements. Each `partnerLink` constitutes a communication channel with a partner, i. e. a web service. As previously stated, WSDL describes functionality provided

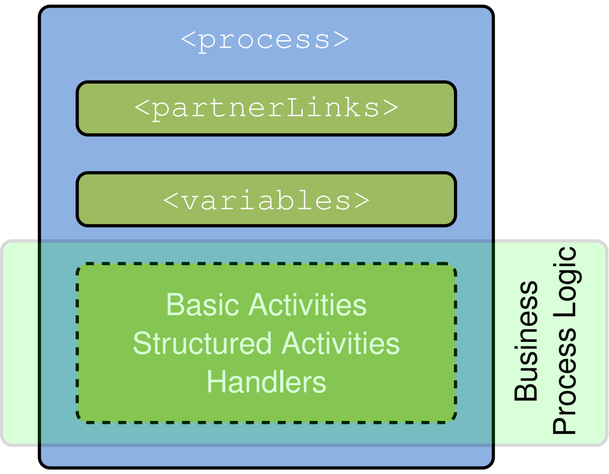


Figure 2.10: *Structure of a BPEL process*

by a web service on an abstract and a concrete level as well as the transmission type of each `operation` such as one-way or request-response. Depending on this transmission type, a different relationship is established between a business process and a web service. For instance, one-way leads to a peer-to-peer relationship such that it is also possible for a business process to be a provider and for a web service to be a consumer. A `partnerLink` element is employed to model this relationship between a business process and a partner based on the `portType` element in the respective WSDL file. Consequently, BPEL allows for defining multiple `partnerLink` elements for a single web service [OAS07].

As the name indicates, the `variables` element allows for defining one or more `variable` elements. Each `variable` represents a typed variable which can be utilized to hold state information of the process in form of messages or temporary data. When executing the business process logic (cf. below), these variables can be read, written and utilized by single activities for e. g. distinction of cases [OAS07].

The core of a BPEL process is the business process implementation which is represented as a set of activities. Two groups of activities are to be distinguished: basic activities and structured activities. Basic activities are characterized by the fact that they implement elemental steps of a process flow such as invocation and assignment operations. Frequently used activities are e. g. `invoke` which performs a web service call and `receive` which waits for the partner to invoke the business process by sending a message. Structured activities are used to model control flows and may be comprised of basic activities or other structured activities. Typical examples are `sequence` which allows to model a sequence of activities and `pick` which allows to select one of several alternative paths [OAS07].

Another important language feature of BPEL are so-called handlers. Handlers are special functions which are called automatically under certain circumstances.

They are utilized to handle message exchange, process failures and to model compensation logic in case of process failures [OAS07].

2.2.2.4 Quality of Service

As stated, a service performs an activity which yields some kind of utility for a requester. The question how this service is provided in terms of non-functional properties such as response time, uptime probability etc. is covered by the topic QoS. Thus QoS describes the modalities at which a service is constituted. Unfortunately there is no agreed upon definition of the term QoS. SABATA et al. describe QoS as "a combination of metrics and policies" [SCD+97, p. 100] whereas O'SULLIVAN et al. consider service quality as "a measure of the difference between expected and actual service provision" [OEH02, p. 125]. These two definitions agree upon that 1) QoS can be measured by means of single parameters and that 2) there is a previously agreed behavior which can thus be expected (and also controlled in terms of QoS parameters) by the service requester. A definition which precisely captures these aspects has been formulated by SCHMITT:

Definition 2.11 [QUALITY OF SERVICE]. *"QoS is the well-defined and controllable behavior of a system with respect to quantitative parameters." [Sch01, p. 4]*

Similar to the definition of the QoS term, there is no general consensus about the question which QoS parameters are relevant in the context of web services. A working group of the W3C prepared a note on this topic in 2003 (cf. [LJL+03]). As of December 2015 however, this note still did not achieve the status of an official draft. Further proposals have been made in the literature (cf., e. g., [SCD+97; MN02; Men02; OEH02; ZBD+03; MS04; YL04]). The reason for this disagreement seems to be fourfold [Ber07]:

1. Web services deliver a vast array of **different functionalities** such as computation and storage,

2. Web services can be utilized for **different purposes** such as outsourcing IT capabilities or renting required IT capabilities,

3. Web services can be utilized in **different environments** such as finance industry and public service sector,

4. Web services can be viewed from **different perspectives** such as purely functional, as software component or as network component.

Thus it is to be doubted if all these differences can be unified in a consolidated QoS model for web services. Hence in the following the most frequently used QoS properties will be enumerated and discussed:

- **Availability:** Availability describes the probability that a service is operating and available for the user. This probability is usually calculated on the basis of a maximum annual downtime hours [Men02; MN02; OEH02; LJL+03; MS04; YL04]. A metric which is also associated with availability is Time-to-repair (TTR) which expresses the period of time the provider expects to require to make the service up and running again in case of a failure [MN02; LJL+03]. Additionally to this temporal availability, O'SULLIVAN et al. also proposed to include spatial availability, i. e. where a service is available [OEH02].
- **Performance:** The performance of a web service is described by multiple quality metrics where *latency*, *response time* and *throughput* are most frequently used. Latency describes the time an empty message needs to travel from the requester to the service provider and back again, excluding any service operations. Response time describes the time a service needs to response to a request, including latency. Throughput describes the rate at which a service is able to process requests in a certain time period [SCD+97; MN02; LJL+03; MS04; Men04; YL04]
- **Reliability:** Reliability describes the degree at which a service is able to maintain its functionality and deliver an expected outcome. This degree is typically calculated based on the number of service failures for a certain time period, usually a month or a year. Reliability also includes assured and ordered message exchange between service requesters and providers [MN02; LJL+03; ZBD+03; MS04].
- **Robustness:** The degree of robustness describes the ability of a web service to function correctly and deliver deterministic results in case of invalid, incomplete or conflicting inputs [SCD+97; LJL+03; MS04].
- **Scalability:** Scalability refers to the ability of a web service to increase its computing capacity in order to serve an increased number of user requests, operations or transactions in a given time period. It refers to the number of requests and transactions that can be served and is related to performance [LJL+03; MS04].
- **Security:** Mechanisms and techniques for ensuring confidentiality and integrity are summarized under the general term security. This includes encrypting message exchange between requester and provider as well as se-

curing service constitution itself. Latter especially requires mechanisms for authentication, access control and non-repudiation [SCD+97; MN02; OEH02; LJL+03; MS04]. For MENASCÉ this also includes resilience to denial-of-service attacks [Men02].

QoS properties of a web service are being negotiated between a requester and a provider in the context of a Service-Level Agreement (SLA). For web services, two XML-based standards are available which allow for capturing SLAs formally, namely the Web Service Level Agreement (WSLA) [LKD+03] language and WS-Agreement [ACD+07].

WSLA has been designed to capture a wide range of assertions associated with SLAs. These include organizational as well as QoS perspectives. From an organizational perspective, WSLA allows for defining management actions which are exposed to other parties involved in the SLA. From a QoS perspective, the standard covers details such as what metrics are used and who is in charge of computing them. The current version has been published by IBM in 2003 [LKD+03].

WS-Agreement is a web services protocol for establishing an agreement between two parties, i. e. a provider and a consumer. It is a facility for documenting service requirements as well as guarantees and is thus similar to WSLA. However as part of the WS-* family of specifications[23], WS-Agreement does not aim at providing a standalone language but an extension to SOAP/WSDL and depends on other WS-* specifications. WS-Agreement has been published by the OPEN GRID FORUM in 2007 [ACD+07].

2.2.2.5 Discussion

SOA gained considerable attention by both academia and industry. While many theoretical properties formulated by academia are reflected in real-world SOA implementations, others proved to be not practical. In this section some of the major discrepancies are explored.

[23] The term “WS-*” refers to a family of web service specifications that have evolved around the three initial standards WSDL, SOAP and UDDI. As the initial standards are general purpose, extensions for special purposes have been covered in separate specifications which can be combined with each other as well as the initial standards. An overview can be found in e. g. [WCL+05].

Discoverability

One characteristic of a SOA frequently mentioned in the literature is discoverability (cf., e. g., [Bur00; LRS02; OEH02; PG03; Erl05]), i. e. the capability to discover single web services to subsequently utilize them automatically. UDDI had long been considered as a technical enabler for this characteristic. But as mentioned in Section 2.2.2.3, the public UDDI repositories of MICROSOFT, IBM and SAP have been shut down in 2006 with the last version of the standard being published in 2004. Consequently the question arises why neither UDDI nor any alternative technology was widely adopted to implement public service registries.

Two possible explanations can be considered. The first one was formulated by BLOOMBERG and blames the desire of businesses to establish relationships based on "a human element" rather than automatically based on the interaction of computers [SAP05]. The second possible explanation is that the technical complexity introduced by UDDI (the last version of the standard counts more than 400 pages) was considered out of proportion with the benefit to discover and automatically bind a relatively limited number of publicly available web services[24]. It was furthermore criticized that UDDI only offers a simple search function allowing to lookup services of certain providers offering specified capabilities. On the contrary, advanced matchmaking mechanisms based on e. g. similarity search were missing which are crucial in cases of e. g. non-uniform naming and ordering of input and output parameters [FFH+03].

Today, web services are usually discovered manually a priori by the future consumers. When utilizing BPEL for service composition, web services are bound statically at design-time by referencing the respective `partnerLink` in the process (cf. Section 2.2.2.3). A more dynamic approach is to change the endpoint of a `partnerLink` at runtime using the `endpointReference` element. This element can be assigned at runtime by the process, e. g. by employing a BPEL `variable` and hence allows for more flexibility [OAS07].

Reported real-world SOA implementations [KBS04; Bra07; LH07; BLN+09; BCH+10; Mur13] make use of service repositories which act as company-wide storage for service descriptions such as WSDL documents. While some of these repositories are merely more than a folder on a server, some repositories also

[24] The exact number of publicly available web services is controversial. ZHENG et al. reported in 2010 a total of 21,358 obtained web service addresses "by crawling web service information from the Internet" [ZZL10, p. 83]. On the contrary, ProgrammableWeb (`http://www.programmableweb.com/`) lists a total of 11,129 web service APIs as of March 8th 2014. Whatever the real number is, it is to be expected that this figure was considerably lower before 2006. To the authors best knowledge, official figures on the number of web services do not exist as of March 2014.

contain information about each service such as the developer, associated SLA, implemented fault handling strategies, etc. The main advantage of these repositories is in coordinating the development efforts of several distributed teams working on different services. None of the reported cases however contained a service registry supporting automatic discovery and binding of services. As such, discoverability is not considered a feature of SOA in this thesis.

Service-Orientation

Although service-orientation forms the core of the SOA paradigm, solely focusing on services does not lead to a successful SOA. The BURTON GROUP reported in 2008 that out of 20 analyzed case studies for SOA projects only 5 were successful. One of the major reasons for failure was the sole focus on technical aspects, especially on implementing services. In consequence organizations implemented many web services, each solving one integration task but lacking reusability[25]. In other words, SOA had been confused with EAI [Mee08].

In both, the case studies analyzed by the BURTON GROUP as well as the ones reported in the literature, a crucial success factor is a focus on the business view [KBS04; Bra07; LH07; Mee08; BLN+09; BCH+10; Mur13]. In most success cases, SOA introduction started from an examination of the organization's current situation in terms of conducted business processes. Service design in these cases was primarily guided by the question what the business needed instead of what was technically possible. This approach naturally lead to a high reusability of services in other business processes. Consequently, SOA introduction was found to be an evolutionary process which is hard to plan completely beforehand. An extreme example is the SOA of CREDIT SUISSE for which preparatory work started as early as 1998 and which is still evolving today. In 2013 the SOA of CREDIT SUISSE consisted of over 1,000 services which were added iteratively over the course of time [Mur13].

Summary

The SOA paradigm describes an ideal which can be technically realized in different ways as proved by several reported case studies [KBS04; Bra07; LH07; BLN+09; BCH+10; Mur13]. Regardless of employed technologies, a common result of all reported case studies was an IT infrastructure consisting of highly reusable

[25] MCKENDRICK coined the term "Just a Bunch of Web Services" or JBOWS to describe such a situation [McK05].

components. A crucial factor of success was to align the technical development with the business needs instead of exploiting all technical possibilities.

2.2.3 Cloud Computing

As discussed in Section 2.2.2.1, enterprise IT departments are under constant pressure to reduce costs. From an economical point of view, IT is no longer considered as necessary business asset but as cost factor. As such, possibilities to turn capital expenditures to operational expenditures are always welcomed by enterprises [Car05]. For several years, the term cloud computing has been present in research, news and media. Although the concept received a lot of criticism in the past from business executives such as LARRY ELLISON as well as software activists such as RICHARD STALMAN (cf., e. g., [AFG+09, p. 3]), the topic seems nowadays to have been generally accepted by the business world. Despite the global surveillance disclosures regarding the NSA which shook the world in 2013, the demand for IT services from "the cloud" is increasingly gaining in importance. By the end of 2013, about 40% of companies in Germany of all sizes and branches were using cloud services which is an increase of 3% compared to 2012 [KPM14]. The German IT industry association BITKOM estimates that in Germany revenue for cloud services reached EUR 8 billion in 2013[26].

The conceptual groundwork for cloud computing had been established in the last century under the term utility computing [FHJ+74]. In occasion of the ARPANET's birth (which should later become the Internet), KLEINROCK described the vision of utility computing as follows: "As of now, computer networks are still in their infancy. But as they grow up and become more sophisticated, we will probably see the spread of 'computer utilities', which, like present electric and telephone utilities, will service individual homes and offices across the country" [Kle69]. Today utility computing is considered to provide IT resources from computing resources such as CPU hours to complex business processes over the Internet[27]. Essential characteristics identified in the literature are shared infrastructures, scalability, reliability and the utilization of pay-per-use as well as pay-as-you-go billing schemes [Rap04; LMH+06; VWB09]. As such, utility computing is defined as follows (based on [VWB09]):

Definition 2.12 [UTILITY COMPUTING]. *Utility computing is the provision of com-*

[26] `http://www.bitkom.org/de/markt_statistik/64086_75301.aspx`

[27] In the years 2000/2001 Intel offered Intel Computing Services which provided CPU hours. The service was not successful though as participation required negotiating a contract and did not allow utilization on a per hour basis, but on longer terms [AFG+09].

puting resources, infrastructures, applications and business processes which are available in a shared, scalable and reliable environment over the Internet on a pay-per-use or pay-as-you-go basis.

Cloud computing has been discussed as a potential solution for delivering IT resources via utility computing by means of an on-demand service. The main driver behind this development was the potential to achieve large economies of scale by concentrating IT capabilities in huge data centers at low-cost electricity locations [AFG+09]. The term cloud is derived from telecommunications of the 1990s. At this time telephone companies in the US started to switch from hardwired connections to digital Virtual Private Network (VPN) solutions. This digitization led to cost reduction while keeping service quality at a similar level. The digital infrastructure, which was thus responsible for tasks such as dynamic routing and load distribution, was called *Telecom Cloud* [Kau09].

Applying the cloud metaphor verbatim from telecommunication to computing naturally caused some confusion. In some definitions it was used as synonym for the Internet or more specifically to describe the parts of an IT infrastructure which are out of the own sphere of influence [MKL09; VVE10]. Obviously these definitions are too general as they allow to define every online service as cloud service. The conceptual mistake here is that in telecommunications, a connection between two partners is the actual service. In computing however the connection is just the transport medium, not the service itself. The nowadays widely adopted NIST definition of cloud computing takes this point into consideration by defining cloud computing as a model which enables access to a shared pool of computing resources (i. e. a cloud) which exhibits certain properties [MG11]. Although a matter of discussion, the following properties are commonly viewed as characterizing a cloud [AFG+09; Gro09; MKL09; WAB+09; RCL10; VVE10; MG11]:

- *Shared Resources:* All users of a cloud share the same physical resources. These resources are centrally managed, e. g. utilizing virtualization technology. For single users this creates the illusion of exclusivity.
- *Configurability:* Provided resources can be configured by users to a certain degree. This degree depends on the virtualized resource as well as the technical implementation of the cloud.
- *Elasticity:* Provided resources can be (automatically) scaled depending on the utilization by the user. To users this creates the illusion of infinite available resources and thus potentially avoids shortages. For providers this leads to a higher utilization of available physical resources.

- *Rapid Provisioning and Release:* User requests for resource provisioning and release can be fulfilled instantly and require only minimal provider interaction.

- *Service Metering:* Resource utilization is thoroughly recorded and thus transparent for the user. Employed metrics depend on the type of service.

Considering these properties, a cloud in the context of cloud computing is defined as follows (based on [MG11]):

Definition 2.13 [CLOUD]. *A cloud is a shared pool of configurable and elastic computing resources such as networks, servers, storage, applications and services which can be metered, rapidly provisioned and released with minimal management effort or service provider interaction. Services provided by a cloud are called cloud services.*

Cloud and utility computing interrelate in the sense that cloud computing employs utility computing principles for service metering such as pay-as-you-go and pay-per-use. The main difference is that utility computing solely focuses on the properties of the provided resources. Cloud computing on the other hand also addresses the question of how these resources are accessed. The following features are frequently cited in the literature [AFG+09; Gro09; MKL09; WAB+09; RCL10; VVE10; MG11]:

- *Ubiquity:* Cloud services can be accessed from a variety of devices and are usually also location-independent.

- *Usability:* Cloud services offer sophisticated user interfaces which require little knowledge about technical details and thus can also be utilized by non-experts. This leads to lower upfront costs and increases acceptance of the service.

- *On-Demand Self-Service:* Similar to a supermarket, users can search and utilize services they require with minimal provider interaction. This is enabled by a high degree of automation in a cloud.

Employing these properties, cloud computing can be defined as follows (based on [MG11]):

Definition 2.14 [CLOUD COMPUTING]. *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a cloud based on utility computing principles.*

Many different service models have been discussed in the context of cloud computing of which three are generally accepted [AFG+09; Gro09; MKL09; WAB+09; RCL10; VVE10; MG11]:

- *Software as a Service (SaaS):* SaaS focuses on providing multiple users with an access to software which is hosted in a cloud. Users can customize this software to their needs at different granularity levels such as user interfaces and database schemes. This service model has been criticized as being a new term for Application Service Providing (ASP). The main difference is however that ASP is an approach to host COTS software in a dedicated environment and to provide users with an access to this environment. The software licenses of COTS software usually do not permit hosting in a virtualized environment. SaaS software on the other hand is designed to run in a virtualized environment. In comparison to ASP this leads to reduced costs as reserve capacities for catching up phases of over-utilization can be kept to a minimum.
- *Platform as a Service (PaaS):* PaaS empowers users to upload and run individual code in an environment which is being run and maintained by the provider. Depending on the service, this code can be either written in any programming language, a proprietary language or might require including some platform-specific libraries. Some PaaS services can also be utilized for application delivery.
- *Infrastructure as a Service (IaaS):* IaaS provides users with access to elementary IT capabilities such as storage and networks or to operating systems (usually in the form of virtual servers). The underlying infrastructure for service provisioning is run an maintained by the provider.

Figure 2.11 shows the interrelation between these service models. As can be seen these service models form a stack (the so-called cloud stack) which are usually provider-dependent and not interoperable. The idea of utilizing SOA approaches to achieve interoperability has been discussed [TSB10]. However, currently available open source cloud stacks (cf. below) are lacking this property.

In the context of cloud computing, several deployment models have been discussed. The following three are broadly agreed upon [AFG+09; Gro09; MKL09; WAB+09; RCL10; VVE10; MG11]:

- *Public Cloud:* Public clouds offer their services to the general public. As this enables economies of scale, public cloud services are usually cheaper than private cloud services (cf. below). Users and providers typically do not

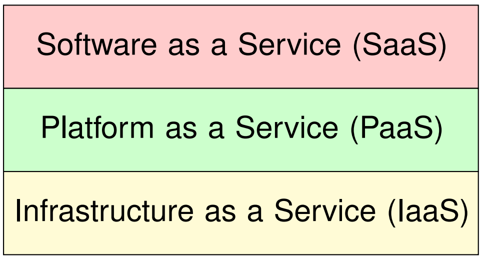

Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

Figure 2.11: *Interrelation between cloud service models*

belong to the same organization. In consequence users have minimal to no control about how their data and applications are being handled.

- *Private Cloud:* Private clouds offer services to a closed user-group exclusively. Typically users and providers belong to the same organization. Alternatively, a private cloud can be run and maintained by a 3rd party for exclusive use. This exclusivity however leads to higher costs at the benefit of a higher degree of control of data and applications.

- *Hybrid Cloud:* As the name indicates, hybrid deployment is an approach to combine the best of both worlds. This empowers cloud user on the one hand to utilize private cloud services for sensitive and critical assets and public cloud services for everything else.

Figure 2.12 shows the interrelation between these cloud deployment models. Similar to the SOA paradigm, cloud computing describes an ideal which can implemented utilizing different technologies. Besides of commercial cloud offerings there are currently several open-source implementations available which exhibit the properties discussed above. Examples are CloudStack[28], Eucalyptus[29] and OpenStack[30]. Furthermore there are several ongoing standardization efforts. Prominent examples are the Open Cloud Manifesto[31] which focuses on cloud computing in general and the Cloud Security Alliance[32] which concentrates on security issues.

[28] http://cloudstack.apache.org/

[29] http://www.eucalyptus.com/eucalyptus-cloud/iaas

[30] http://www.openstack.org/

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

[32] https://cloudsecurityalliance.org/

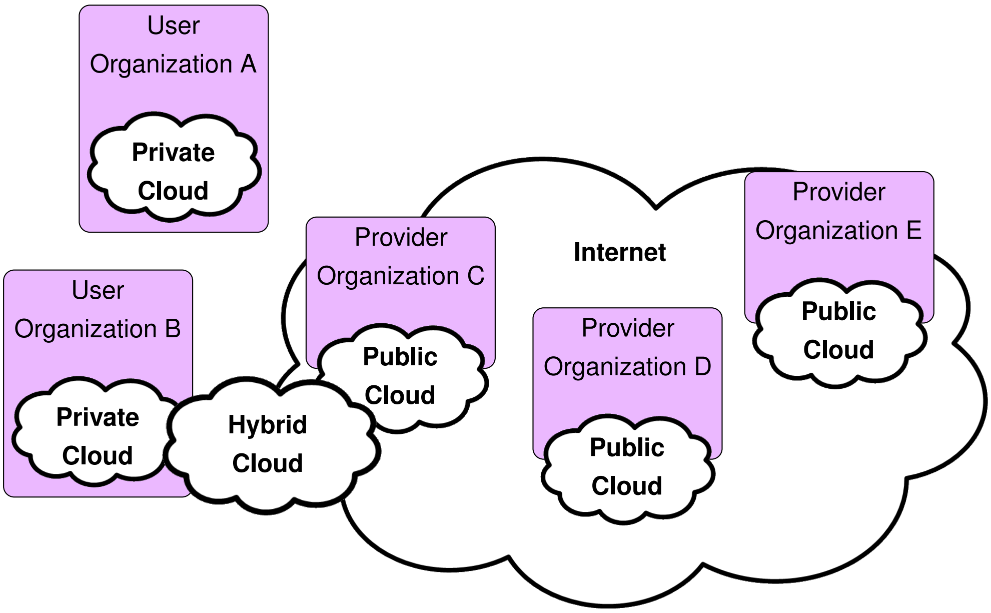


Figure 2.12: *Interrelation between cloud deployment models (based on [BKN+11])*

2.3 Interdependent Security

This section first introduces basic security terminology utilized throughout this thesis. Next, an overview about formal methods for security verification is provided. Finally, interdependence effects in security are discussed.

2.3.1 Basic Terminology

In this work, security is being considered in the context of *IT systems* which are defined as follows (translation of the definition in [Eck12, p. 3]):

Definition 2.15 [IT SYSTEM]. *An IT-System is a closed or an open, dynamic technical system which has the capability of storing and processing information.*

In this context, a closed system is considered as a proprietary system which is

- built on the technology of a single manufacturer,
- incompatible to products of other manufacturers,
- reserved to a closed circle of users,
- usually homogeneous in terms of hard- and software and
- centrally administered.

On the contrary, an open system is characterized as a physically distributed system which is

- composed of networked subsystems where each subsystem can communicate with other systems on its own,
- making use of standards for exchanging information with other systems,
- usually heterogeneous in terms of hard- and software and
- not centrally administered.

IT systems are subject to a wide array of *threats*. A threat aims at exploiting one or more *vulnerabilities* or *weaknesses* of an IT system in order to unauthorizedly acquire or modify *information assets*. In this context, the term *information* is employed in a broad meaning and includes data, programs as well as hardware structures [Pfi00]. Examples for threats are malicious users, hard- and software faults and user mistakes. Classifications of threats are covered thoroughly in the literature [BK02; Bis05; Eck12]. Each threat is directed against one or more *security objectives*. A usual classification of security objectives in the context of IT systems is as follows (cf., e. g., [VK83; GR95; Bis05; ALR+04]):

1. **Confidentiality:** Access to information is exclusively granted to authorized entities.
2. **Integrity:** Information is not being modified or deleted in an unauthorized and undetected manner.
3. **Availability:** Information is available to authorized entities, regardless of location and time.

This classification is also referred to as the "CIA model". More fine grained classifications have been proposed by e. g. KESDOĞAN which allow for subdividing these aforementioned security objectives into more specific concerns. For instance, *anonymity* is being viewed as part of confidentiality and *accountability* as part of integrity [Kes00]. Alternative classifications have been proposed by e. g. [Gov93; BK02; Eck12; ISO08].

The distinction of what is allowed from what is not with respect to information assets is being expressed by *security policies*. In other words, a security policy specifies which actions are *secure* and *non-secure* respectively and can be either formal or informal. Security policies are enforced utilizing one or more *security mechanisms*. A security mechanism might be technical (e. g. a method or a tool) or nontechnical (e. g. organizational procedures) and aims at preventing or detecting an attack or to recover from an attack [Bis05]. For instance, restricting access to information by means of confidentiality can be technically enforced utilizing cryptography (cf., e. g., [Den82; MOV97]).

Figure 2.13 shows the interrelation of the terms introduced in this section. It should be emphasized that the terminology presented here is by no means exhaustive but represents a minimal set which is necessary for the understanding of this thesis. For a thorough presentation the interested reader is referred to the corresponding standard literature such as [Bis05; Eck12].

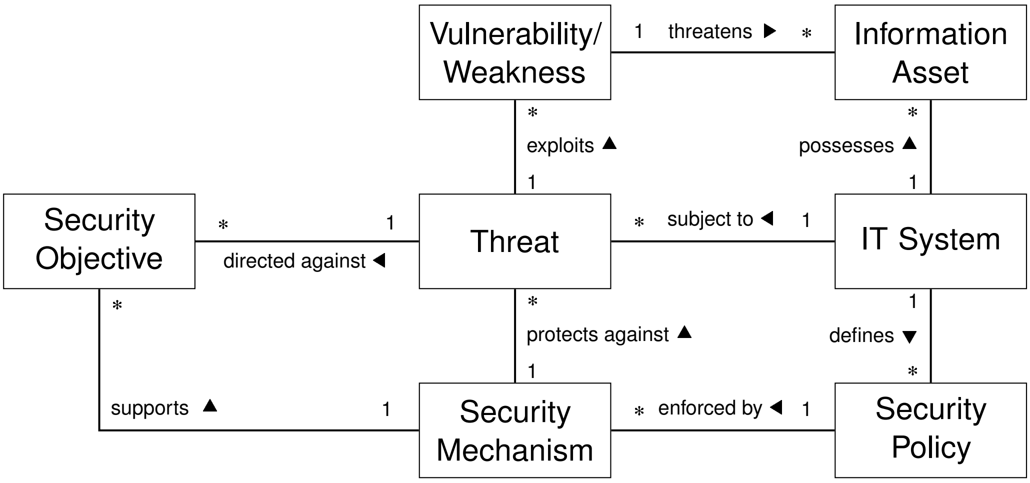


Figure 2.13: *Interrelation of Security Terms*

2.3.2 Formal Methods

In a perfect world, every developed IT system would exactly behave as specified and correctly implement its security policies. This is however rarely the case. In fact, delivering functionality as expected via *legitimate channels* only is a fundamental security property. According to PFITZMANN this corresponds to total correctness [Pfi00]. The question is thus how to ensure this property of IT systems. Formal methods are a prominent approach to support this goal and can be applied at different phases of the systems development life-cycle (e. g. specification and implementation) and for different aspects of an IT system (e. g. hardware, software and protocols). They aim at modeling, analyzing and verifying (parts of) IT systems [WLB+09; KTV+10; GG13].

A formal method is characterized by a sound mathematical basis which provides the foundations to precisely define and reason about notions such as soundness. It also serves to prove certain properties of an IT system based on a model. This model is specified in some (formal) specification language. Another essential characteristic is a methodology which describes how to effectively apply a formal method. This also includes a clear definition of the problem scope and the target user group the method is designed for. Some formal methods also provide tool support [Win90; WLB+09; GG13]. Due to this wide scope, it is naturally difficult to formulate a definition which captures all nuances of formal methods. Hence the definition of GARAVEL and GRAF is employed which the author believes is well-suited for the scope of this thesis:

Definition 2.16 [FORMAL METHODS]. *"Formal methods in a broad sense are mathematically well-founded techniques designed to assist the development of complex computer-based systems; in principle, formal methods aim at building zero-defect systems, or at finding defects in existing systems, or at establishing that existing systems are zero-defect" [GG13, p. 15].*

As the adverb "in principle" in Definition 2.16 indicates, it is the sublime goal of formal methods to prove that systems are zero-defect. This goal is however accompanied by several challenges which reach from fundamental results in computability theory to communication problems in general [Win90; And08; GG13]:

- *Undecidability:* Formal methods aim at verifying certain properties of systems. However, *Rice's theorem* states that deciding if a program yields a certain non-trivial property can be reduced to the halting problem which is known to be undecidable [Ric53]. Hence, there can be no algorithm which can automate this task. This issue can be potentially circumvented by restricting a formal method's scope. For instance, the expressiveness of the utilized specification language can be restricted to problem classes which are decidable. Another possibility is to restrict tool support such that a formal method produces output which needs to be manually verified by human beings.
- *High-Complexity:* Assuming that a formal method considers a problem scope which is (semi-)decidable, the problem can still yield a high complexity. Particularly, if a problem is NP-complete, exact solutions are not generally feasible, but only for problem instances up to a certain size. While clever tool design can certainly support avoiding high complexity, this challenge remains a real obstacle for the analysis of large-scale systems.
- *Model inaccuracies:* Since a model is just a simplified projection of the real-world, it can never capture all facets. This is especially true for the environment where a system is intended to run. As a system rarely runs in isolation, it can be affected by its environment in ways which were not anticipated during its formal specification and verification. Particularly, some aspects

such as human errors and natural hazards are impossible to model. Another crucial factor are unrealistic (or outdated) assumptions regarding the environment. Such model inaccuracies lead to formally verified systems which might fail in real environments.

- *Inadequate specification:* A challenge which can occur even in a fully verified system is that the initial specification proves to be inadequate. The reasons for this mismatch between requirements and specification might be multiple but can be typically traced back to either inconsistent or misinterpreted requirements. The result will thus be a system which is formally verified but which does not deliver what was intended. This kind of challenge does not exclusively occur when applying formal methods but is a general issue which frequently occurs in systems development. According to the STANDISH GROUP, only 39% of IT projects were classified as "successful" in 2012 which among budget and time constraints also includes completeness in terms of features and functions [The13b].

As a result of these challenges, formal methods are employed for comparably small systems or for small parts of systems respectively. According to a survey conducted by WOODCOCK et al. in 2009, out of 62 projects which utilized formal methods, most had a size of up to 100,000 lines of code. Most of these projects were systems from transportation and financial sectors. Predominant system types were real-time applications, distributed applications, transaction processing and high data volume. 30% of these projects were furthermore applying for certification [WLB+09].

Since formal methods cover a vast field, a comprehensive survey it not possible here. Hence this section will concentrate on two major classes of formal methods which are frequently employed in security, namely *model checking* and *theorem proving*. This decision has been taken since the concepts introduced here are also employed in the research fields of compliance and secure service composition and hence support understanding related works discussed in Sections 2.4 and 3.3 respectively. For a more exhaustive overview of formal methods in general and for security, the interested reader is referred to [Mon03; GG13].

The common denominator of the approaches discussed here is that security of an IT system is verified with regards to some *security model*. Frequently encountered security models in this context are the models of BELL-LAPADULA, BIBA and CLARK-WILSON. BELL-LAPADULA is a secrecy model which aims at preventing information leakage from high to low levels in a hierarchy. This is achieved employing no read-up and no write-down rules [BL73]. Similar, but aiming at integrity, is the model of BIBA which defines the no read-down and no write up rules [Bib75]. Another, but less formal, model for integrity is the CLARK-WILSON model which employs a set of nine rules of practice [CW87].

Model Checking

Model checking has been firstly described by CLARKE et al. in 1983 [CES83] and involves the construction of an abstract model $\mathcal{M}$ as well as a specification formula f which represents some desired property. $\mathcal{M}$ is defined using state-transition graphs while f is defined using temporal logic. Model checking then aims at finding all states $s \in \mathcal{M}$ which satisfy f, i. e. $\mathcal{M}, s \models f$. A major advantage of model checking approaches is that they do not require constructing correctness proofs. However, since model checkers perform exhaustive state explorations, *state space explosion* is a perilous soft spot of this kind of approaches [Cla08].

The predominant application area for model checking in security research is analysis of cryptographic protocols (cf., e. g., [MCF87; Mea96; Bla01; BMV05]). These approaches usually build upon the DOLEV-YAO model where the network is represented as a set of abstract participants who are mutually connected via bidirectional communication channels. Participants can exchange messages without authorization. At any given time new participants can enter the network and current users can leave it. The adversary in this model is assumed to be capable of overhearing, intercepting and synthesizing any message in the network. The only limitation for the adversary are cryptographic operators which are assumed to be "perfect", i. e. no information is leaked. In other words, the only way an adversary can read encrypted messages is to have the proper decryption key [DY81].

Available works on cryptographic protocol analysis are mostly about protocol analyzing tools written in Prolog. First approaches in this direction are the INTERROGATOR and the NRL PROTOCOL ANALYZER which employ multiple state machines for representing protocol participants. The states of these machines correspond to protocol interactions where certain states are defined to be insecure. In this model a protocol is secure if insecure states are proven to be unreachable [MCF87; Mea96]. A similar, but light-weight approach has been proposed by BLANCHET which makes use of several abstractions in order to avoid state space explosion [Bla01]. Another work which aims at circumventing state space explosion is the OFMC MODEL CHECKER by BASIN et al. which builds the state space in a demand-driven fashion. This enables to significantly reduce the search space and thus allows for analyzing bigger problems [BMV05].

Other application areas for model checking in security include (but are not limited to) verifying security properties of control flow graphs (cf., e. g., [BJL+01]) and business processes (cf., e. g., [ACP+11]).

Theorem Proving

Theorem proving in its current state can be traced back to two fundamental works by HOARE and DIJKSTRA which were published in 1969 and 1975 respectively [Hoa69; Dij75]. The goal is to prove the general validity of a formula f (also called *theorem* in this context), i. e. $\models f$. This is achieved by showing that f is the logical consequence of a set of other formulas by employing *proof inference* over a set of *inference rules* and *axioms* [Ber02]. An inference rule is given in the following form:

$$\frac{P_1, \quad P_2}{C} \tag{2.2}$$

Here, P denotes a premise (with "," denoting concatenation) and C a conclusion. An inference rule states that if all premises are derivable, then the conclusion is guaranteed to hold. Inference rules without any premises are called axioms. A proof obtained by theorem proving has the form of a *derivation tree*. The nodes of this tree represent formulae with the root node representing f. Every node of the tree is at the same time the conclusion of an associated inference rule while its child nodes are premises of this inference rule. Leave nodes are associated with axioms. A theorem is proven valid if it can be logically deducted from the leave nodes to the root (cf. Figure 2.14).

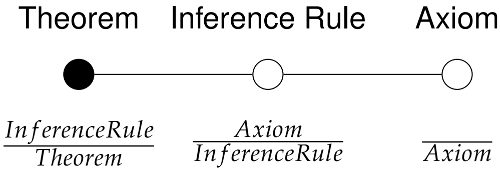

Theorem
Inference Rule
Axiom
$\frac{InferenceRule}{Theorem}$
$\frac{Axiom}{InferenceRule}$
$\frac{}{Axiom}$

Figure 2.14: *Example of a derivation tree (black node representing the root node)*

In contrast to model checking approaches, theorem provers do not require to exhaustively search the state space of a program and hence can even handle problems of infinite size. However, theorem proving often requires interaction with a user to guide the process (*interactive theorem proving*) while some approaches can work autonomously on rather simple problems (*automatic theorem proving*)[33].

Similar to model checking approaches, a major application area for theorem proving is reasoning about cryptographic protocols. Early works such as [DS97; Pau97] employed general-purpose provers for this task. DUTERTRE and SCHNEIDER embedded CSP[34] in the PVS prover. This combination allowed them to verify several authentication protocols [DS97]. PAULSON employed the ISABELLE/HOL

[33] Cf. [Ber02] for a detailed comparison of model checking and theorem proving.

[34] A formal language for describing interactions between processes developed by HOARE.

prover to demonstrate his approach to reason about authentication protocols using induction [Pau97]. More recently, MOEBIUS et al. utilized the interactive theorem prover KIV[35] to verify application-specific security properties of cryptographic protocols. This approach uses an application's UML model to generate a tailored formal specification [MSR10].

Other major lines of security research involving theorem proving deal with verifying the correctness of cryptographic protocol implementations (cf., e. g., [BFG+08]) and information flow properties in software systems (cf., e. g., [DHS05]). Furthermore theorem proving techniques have been used as building blocks of several security-related verification approaches. For instance, NECULA developed a framework called Proof-Carrying Code which formally proves that a program meets certain safety and correctness requirements. These proofs are generated employing a theorem prover and are subsequently attached to the program. An entity receiving this program can validate the proof and thus ensure that a program actually fulfills its claimed safety and correctness requirements prior to running it [Nec98].

Summary

Utilizing formal methods can ensure to a certain degree that an investigated IT system behaves as specified. However, formal methods come along with some challenges as discussed above. Hence they are usually employed for rather small (sub-)systems. Employing them in greater systems requires a certain degree of abstraction which usually increases with increasing system size. This leads to the question to what degree formal verification results can still be viewed as reliable if achieved at a high abstraction level. It also needs to be explicitly stated that utilizing formal methods is no guarantee that an IT system is bug free since the absence of bugs can never be proved [Dij72]. Furthermore, there are to the best of our knowledge no formal methods which are able to cover interdependence effects between security goals (cf. Section 2.3.3).

2.3.3 Interdependence Effects

Security mechanisms are usually assumed to be independent of each other. However, as early as 1995 interdependence effects have been discussed in the context of technical and organizational mechanisms by the National Institute of Standards

[35] A dedicated prover to reason about correctness of software systems, cf.: `http://homepages.inf.ed.ac.uk/wadler/realworld/kiv.html`

and Technology (NIST). The NIST handbook on computer security discusses several such effects for different types of mechanisms and emphasizes that careful selection might lead to synergy effects. On the contrary, if the scope and the effects of mechanisms are not fully understood, they might undermine each other. It is however also stated that the identification of interdependence effects is a tough challenge since factors such as system management, legal issues, quality assurance as well as internal and management controls may also affect the effectivity of mechanisms. Exemplarily the interdependence of physical and logical access controls will be discussed. Logical access controls restrict access to information to authorized entities in non-physical environments such as web-based applications. This however can be easily circumvented if insufficient physical access controls to the respective IT systems are employed. For instance, an attacker could directly access the hardware and storage media which would leave logical access controls completely ineffective. Hence, logical access controls need to be flanked by appropriate physical access controls to be effective [GR95].

Similar to security mechanisms, security objectives are also silently assumed to be independent of each other by most authors. In this regard, WOLF and PFITZMANN were the first who discussed the interdependent nature of security goals [WP00]. They identified three different types of interdependencies for two security goals A and B:

1. A might *strengthen* B. Sometimes, A is a weak form of B.

2. A might *weaken* B.

3. A might *implicate* B, i. e. A is a guarantee for B or A is a sufficient condition for B. B may be a weak form of A.

This classification yields some similarities to the interdependence effects described in [GR95]. Strengthening relationships correspond to "synergy" effects and weakening relationships to "potential undermining" effects. Implication relationships have no correspondent in [GR95].

The presented types of interdependence were discussed based upon the security model shown in Fig. 2.15. Here, this security model will be used in order to explain the different types of interdependence with simple examples. For a more thorough discussion on each security goal and the model itself, the interested reader is referred to [WP00].

Strengthening relationships will be explained on the basis of the security goals *confidentiality* and *anonymity*. Confidentiality generally means that in case of communication nobody except the communication partners knows about the content

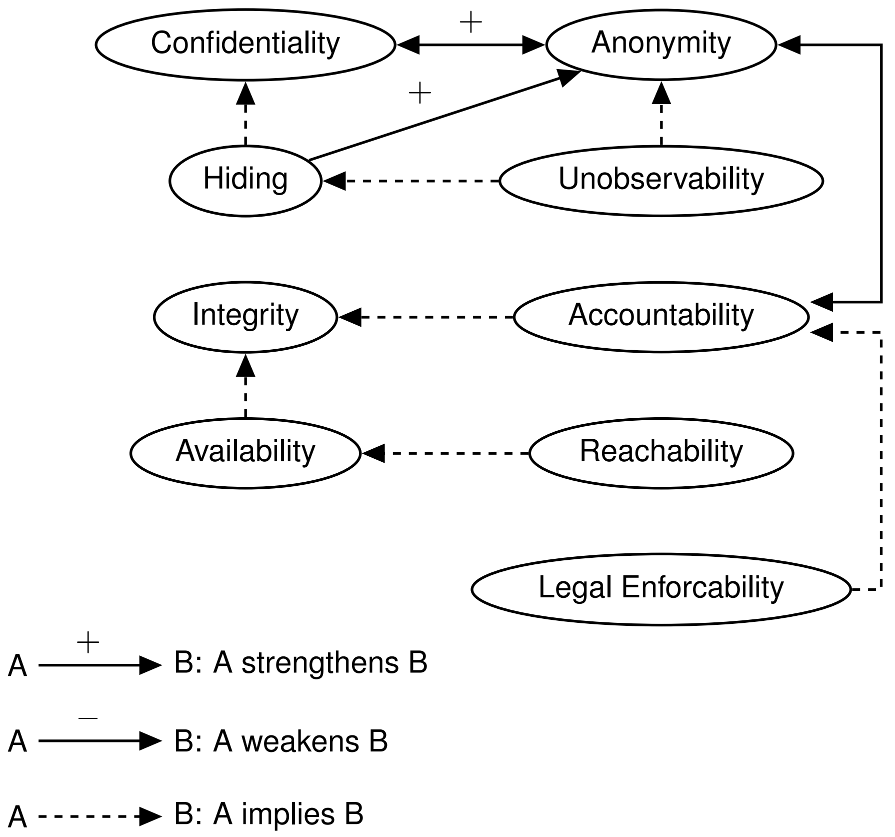


Figure 2.15: *The security model of* WOLF *and* PFITZMANN *(cf. [WP00])*

of the communication while their identities may be known. Anonymity means that a user is able to use a service or resource without disclosing his/her identity [WP00]. This means that the security of systems which utilize techniques for confidentiality (e. g. encryption) can be further enhanced by employing anonymization techniques such as MIX-networks (cf., e. g., [KP06]) and vice versa[36]. Therefore confidentiality strengthens anonymity and vice versa.

For explaining *weakening* relationships, again *anonymity* and *accountability* will be employed. Accountability provides proof of communication, i. e. no communication partner can successfully deny having sent or received a certain piece of information [WP00]. Systems which require accountability need to be able to link actions to identities which in general is not possible if users employ anonymization techniques. On the other hand, systems which require anonymization cannot link actions to identities. Thus, accountability and anonymity mutually weaken each other.

Finally, *implication* relationships will be explained on the basis of the security

[36] Of course, MIX-networks already employ encryption for building routes from the sender to the recipient. The previous statement refers to encrypting the plain messages exchanged between communication partners, independently from the MIX-network.

goals *unobservability* and *anonymity*. Unobservability is understood as the ability of a user to use a service or resource without anybody being able to observe that usage [WP00]. As such, unobservability can be viewed as a stronger form of anonymity. Therefore, a system which ensures unobservability of users automatically guarantees their anonymity as well. But in case that unobservability is not given, users may be anonymous or not. However, it can never be the case that a user is unobservable but not anonymous. In this case the implication relationship between unobservability and anonymity would not be true.

Research about interdependent security can be classified into the following groups: Technical approaches [GR95; WP00; Bis09; NL12] and descriptive models based upon game theory (e. g. [KH03]). Besides of the works discussed in the previous section [GR95; WP00], only few technical works have been published. BISKUP provides a discussion regarding the relation between security objectives as well as their support for autonomy and cooperation of communication partners [Bis09]. Interdependence is not explicitly addressed. NIETO and LOPEZ present a domain-specific QoS ontology aiming at integrating different types of networks [NL12]. This model takes interdependence between security objectives into account by means of single quality attributes.

In the group of descriptive models, the first seminal work was published by KUNREUTHER and HEAL [KH03]. This paper laid the foundation for a plethora of complementary research considering interdependent security in varying settings and under different assumptions. The drawback of these models is that they are not well suited to deduct real-world QoS requirements.

Conceptually related is the work of YAU et al. [YYA09] which presents a tradeoff model for balancing performance and security in service-based systems. Interdependence between security objectives is not addressed.

2.4 Business Process Compliance

In real-world scenarios, business processes or even single tasks are subject to semantic constraints which arise from guidelines, regulations, corporate policies and laws. These semantic constraints which for instance may restrict the execution of tasks at certain locations are called *compliance rules*. Generally, a compliance rule can refer to one or more *compliance aspects* (cf. below). A business process is called *compliant* if it does not violate any compliance rules it is subject to. Techniques for ensuring the compliance of business processes are subsumed under the term *business process compliance* [RW12]. Compliance management has become particularly important after the confidence of stakeholders has been erupted by several collapses of major enterprises in the early 2000s. Measures for increasing the transparency of business processes are intended to regain trust [Kar08].

Two groups of compliance approaches can be distinguished (cf. Figure 2.16). The first group are regulatory approaches which either stem from mandatory legal regulations such as SOX, EUROSOX and MIFID or from strategic goals such as corporate responsibility. Legal regulations are intended to increase transparency of companies' business by introducing mandatory standards for activities such as reporting (cf., e. g., [US 02]). Corporate governance on the other hand is a facility to supervise the achievement of self-declared goals of organizations. Such self-declared goals may be either monetary or non-monetary and are typically formulated by the board of directors [Kar08].

The second group consists of standardized approaches which were either developed by independent organizations such as International Organization for Standardization (ISO) or by groups of interest such as Information Systems Audit and Control Association (ISACA). In both cases the goal is to create uniform guidelines to increase efficiency and transparency of organizations on a voluntary basis [Kar08].

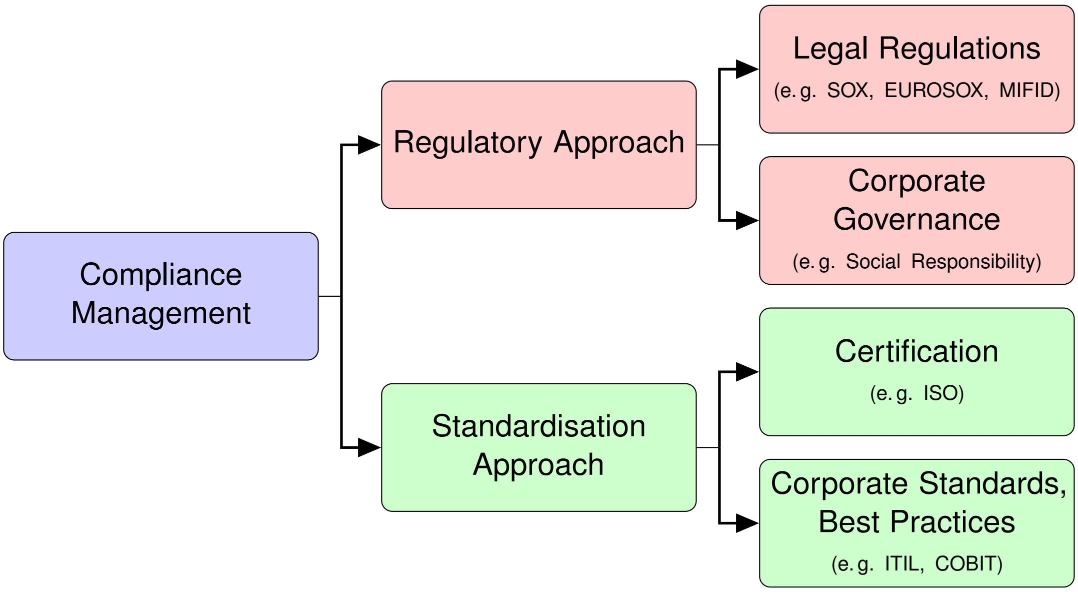


Figure 2.16: *Elements of Compliance Management (cf. [Kar08])*

As previously mentioned, a compliance rule refers to one or more compliance aspects. In total, the literature (cf., e. g., [CKO92; SZ01; SCM+07; CRR10; ALS11; RW12]) differentiates between five compliance aspects for workflows, namely **Activities**, **Data**, **Location**, **Resources** and **Time limits**[37]. In the following, the

[37] Some authors view the aspects *Location* and *Resources* as belonging together by means of an organizational perspective (cf., e. g., [CKO92]). In this thesis however these aspects will be

primary question which is addressed by each compliance aspect will be named and an example from either Health Insurance Portability and Accountability Act (HIPAA) [US 96] or the Bundesdatenschutzgesetz (BDSG)[38] [Bun90] will be cited:

- **Activities:** Which tasks are performed in which sequence? (BDSG:4) requires that users need to agree to the collection of their personal data before corresponding actions may take place. In terms of BPM this aspect refers to the control flow of process models.
- **Data:** What data objects are being produced and consumed in the process and which management rules are applied? (BDSG:3a) requires that personal data has to be anonymized or pseudonymized if there is no justified need to access these kinds of data in plain text. In a strict sense, this aspect refers to two distinct properties of processes. On the one hand, it refers to the data flow in processes and the general lifecycle of data in a process from creation to deletion. On the other hand, this aspect defines data management rules which can be considered as local restrictions for single tasks in a process.
- **Location:** Where are tasks performed? (BDSG:4b) requires that personal data might be transferred to 3rd parties outside of Germany only if an appropriate level of protection can be assured by the 3rd party for the transferred data. This aspect defines a local constraint on single tasks in a process.
- **Resources:** By whom are tasks performed? (HIPAA:164.530.a.1.ii) requires that institutions implementing HIPAA must appoint a person or office responsible for receiving privacy complaints. In terms of a process model this aspect restricts the entities that are allowed to perform a task.
- **Time limits:** Within which time constraints are tasks being performed? (HIPAA:164.524.a.1; HIPAA:164.524.b.2.i) give patients the right to be informed about certain types of their stored protected health information (PHI) within 30 days. Applied to a process model this aspect defines a time scope on a subset of tasks.

Given these aspects, the question is how to model compliance rules in order to capture each case. This decision also depends on the question when a compliance check shall be performed. Regarding the BPM lifecycle (cf. Section 2.1.1) there are four possibilities, i. e. process design, system configuration, process enactment and diagnosis. This view is however different from the classical dichotomy between

considered separately since they lead to different requirements in terms of service composition.

[38] *German Federal Data Protection Act*

design-time, run-time and backward approaches for business process compliance as proposed by e. g. El Kharbili et al. [EMS+08] and Ly et al. [LRG+12][39].

Mapping the BPM lifecycle on this tripartition reveals that the phase system configuration has an exceptional position between design-time and runtime. On the one hand, system configuration is timely located after design-time since process modeling is finished in principle[40]. On the other hand, system configuration is still before runtime as there are obviously no running process instances yet (cf. Figure 2.17).

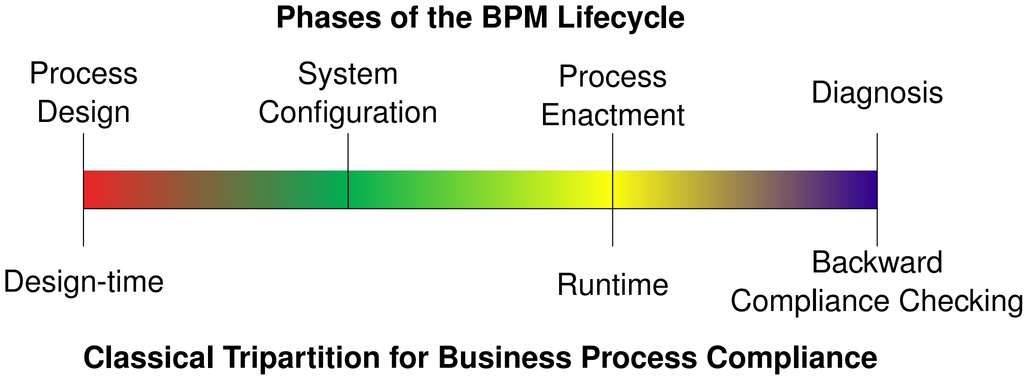

Phases of the BPM Lifecycle
Process Design
System Configuration
Process Enactment
Diagnosis
Design-time
Runtime
Backward Compliance Checking
Classical Tripartition for Business Process Compliance

Figure 2.17: *Mapping of BPM lifecycle phases to the classical tripartition of approaches for business process compliance*

In the following, proposed approaches for business process compliance will be reviewed. The approaches have been classified according to the phases of BPM lifecycle instead of the classical tripartition. This decision was taken in order to show which compliance aspects are being tackled by proposed approaches at each phase. Furthermore, this will serve as a preparatory step for the subsequent discussion where it will be investigated which compliance aspects can be verified at each phase of the BPM lifecycle. Due to the sheer mass of available literature for this topic the author has tried to select a representative number of works for each phase. The search was conducted over the databases ACM Digital Libaray, IEEE Xplore, AIS eLibrary and Springer Online. Furthermore, the citations of literature found as well as their references sections were analyzed to find further works.

[39] El Kharbili et al. additionally consider design-time and runtime approaches to be related and proposed a meta-level called forward compliance checking as counterpart to backward compliance checking [EMS+08]. This more coarse-grained view will however not be adapted here.

[40] "In principle" because in a strict sense this is only true in case of directly executable process models. In case of non-executable process models such as BPMN, remaining open questions after design-time/process design phase are which services to select and how to transform the process model into an executable form.

Process Design

Approaches for design time compliance checking range from methods to guide users in the design process to static checking methods for verifying compliance of modeled processes.

Maybe the most straight-forward way to model compliance requirements is to semantically enrich process models by annotating activities. Respective approaches have been proposed by SADIQ et al. [SGN07], THOMAS and FELLMANN [TF09] as well as D'APRILE et al. [DGM+12]. Compliance annotations are utilized in order to check control and data flow properties of processes against compliance rules after the design is finished. Proposed annotations include formal constraint languages [SGN07], ontologies [TF09] and temporal logic [DGM+12].

SCHLEICHER et al. proposed an approach which provides process designers with compliant process templates in a generic notation. Compliance rules are modeled utilizing so-called compliance activities which have a fixed position in the process model and thus cannot be moved. Between these compliance activities are so-called opaque activities which act as placeholders for designing individual process flows [SAL+09]. Similarly, KITTEL and SACKMANN propose the concept of reference controls for enriching process models. Each reference control implements a compliance requirement and more complex requirements can be captured with multiple reference controls. Furthermore, the authors propose to parametrize these controls in order to allow for dynamic adaptation of control flows during runtime [KS12].

GHOSE and KOLLIADIS proposed an approach for detecting compliance violations in terms of activities in BPMN models. In case that a process model is found to be not compliant, modifications are suggested to restore compliance again. The basis for these suggestions are so-called compliance patterns. A compliance pattern is an anti-pattern which models commonly occurring compliance violations [GK07]. Conceptually similar approaches were proposed by NAMIRI and STOJANOVIC [NS07] as well as GOVERNATORI et al. [GHS+09]. SCHUMM et al. present a complementary approach which additionally allows for identifying compliant parts of processes which can be reused [STK+10].

Besides these guiding approaches, a number of methods were proposed to check compliance properties of process models after the design is finished. A major group of these approaches are based on deontic logic and rule Petri nets. GOEDERTIER and VANTHIENEN utilize temporal deontic logic to verify sequences and timings of activities in process models [GV06]. A more formal approach has been proposed by AWAD et al. which operates on process models transformed into finite state machines and compliance rules expressed as temporal logic formulas.

Model checking is employed to verify if the temporal logic formulas are respected by the finite state machine. State-space explosion is circumvented by employing a set of reduction rules [ADW08]. BRÄUER et al. propose a generic model query language for identifying compliance violations in process models. The approach operates on a graph-based representation of process models as well as attribute values of object variables in order to identify the occurrence of certain patterns. Each pattern corresponds to a compliance violation [BDD+13].

ACCORSI et al. model compliance rules as rule Petri nets and statically check process models if they violate any rule. This approach is able to fully cover the aspects activities and resources. Data flows are also covered, however data management rules in terms of e. g. required encryption schemes are not as this depends on the concrete services employed to implement the process. Similarly location and time limits are only marginally covered as these aspects also depend on employed services and thus cannot be determined during design time [ALS11].

A work similar to [ALS11] but solely focusing on data flow properties in workflows, is presented by MEDA et al. It utilizes a graph traversal algorithm to detect compliance violations regarding data flows in nested as well as unstructured workflows. And like the work presented in [ALS11], the presented approach does not cover data management requirements [MSB10].

The works of BECKER et al. aim at bridging the gap between compliance and business rules. Compliance requirements are modeled as business rules which are mandatory or which must not occur in process models. Validating that a process model contains or does not contain certain business rules is performed employing model-checking. The authors make use a of a domain-specific business process notation called SBPML which has been developed for the financial sector [BBD+11; BAC+11].

In summary, approaches belonging to the process design phase of the BPM lifecycle are able to fully cover the compliance aspects activities and resources. Data aspects are only partly covered, i. e. data flow properties are covered but data management is not. Similarly restricted are location and time limit aspects. Single tasks in the process model can be annotated to mark such restrictions. But since the compliance of these aspects cannot be determined during design-time, they are considered only to be partly covered at this phase.

System Configuration

Compliance checking at this phase can be performed in basically two ways, namely by considering compliance 1) during transformation or 2) after transformation.

Surprisingly, proposed approaches for the first group are limited to detecting and solving structural issues such as deadlocks in service choreographies (cf. Section 3.1) using process calculi (cf., e. g., [BZ07; BBM+08]). The author is not aware of any approaches for service selection in orchestrations with respect to compliance aspects.

Approaches which aim at transformed process models usually perform checks against compliance rules and are thus very similar to static compliance checking methods proposed for the process design phase. An important difference is however that at this stage concrete services are already selected and thus compliance checks can be extended to these as well. Temporal logic is often utilized to model compliance rules while verification is performed utilizing either model checking or finite state automata.

The approaches proposed by YU et al. [YMH+06], LIU et al. [LMX07] and MONAKOVA et al. [MKL+09] all consider control and data flow properties of BPEL processes. In [YMH+06] and [LMX07] compliance rules are modeled in temporal logic. YU et al. however additionally provide a domain-specific language called PROPOLS to ease modeling of compliance rules by process designers [YMH+06]. Both works verify the compliance of process models utilizing finite state automata. In [LMX07] however the process model is first transformed into a equivalent formal representation utilizing the π-calculus [Mil93]. MONAKOVA et al. propose to model compliance rules as logical assertion expressions using predicate logic. Verification is performed by means of checking the satisfiability of these logical assertions using a Satisfiability Modulo Theories (SMT) solver [MKL+09].

The work of TRČKA is related to the works presented in [YMH+06; LMX07; MKL+09] in the sense that it also focuses on detecting compliance violations in terms of control and data flow errors. However, the approach is applied to workflow nets (cf. [Aal98]). Instead of compliance rules, the work focuses on anti patterns which model compliance violations utilizing temporal logic. Process models are verified against these anti patterns utilizing model checking techniques [TAS09].

BARESI et al. developed an assertion language for BPEL processes called ALBERT. The language supports the verification of control flow and time limit constraints of service compositions. Latter is possible as this approach considers QoS properties of selected services. The work also proposes a component for monitoring and evaluating compliance rules at runtime (cf. below) [BBG+07]. Similar approaches for BPEL processes include the works presented in [XCZ+08; LQS12]. XIAO et al. propose to verify SLA conformance of BPEL processes utilizing simulations [XCZ+08]. In the approach of LOMUSCIO et al., contracts are specified as BPEL behaviors. These contract descriptions are then transformed into a domain-specific language which forms the basis for verifying BPEL processes employing a model checker [LQS12]. An approach for visually modeling contracts was proposed by MARTINEZ et al. Compliance of processes against these constracts is verified using timed automatons [MDC11].

Regarding the verifiability of compliance aspects at this phase, it is necessary to consider approaches for service selection and approaches for checking compositions separately for the following reasons. During service selection it would be possible to verify aspects such as data management, location and time constraints by considering QoS properties of service alternatives. The sequence of activities on the other hand has to be taken as granted as it is not the purpose of the service selection problem to question the validity of the process model (cf. Section 3.2.4). The resources aspect finally requires configuration effort and is thus out of scope of service selection. As the author was not able to find any service selection approach providing this kind of compliance-awareness, this is considered a research gap. On the contrary, approaches for transformed process models additionally consider the order of tasks, data flow properties and resources. Generally, at this point of the BPM lifecycle and in all following phases, it is possible to verify all compliance aspects as necessary information is now available in the form of either process traces or service SLAs.

Process Enactment

At the third phase, process instances are deployed and running. As such, compliance verification at this phase of the BPM lifecycle is basically monitoring of running process instances.

PESIC and VAN DER AALST utilize linear temporal logic to construct state automatons [GPV+96] for monitoring and detecting violations in running process instances [PA06]. AGRAWAL et al. propose using activity logging and adapting running process instances according to these activity logs to enforce compliance [AJK+06]. A similar work is presented by ROZINAT and VAN DER AALST who verify conformance of process instances by analyzing event logs at runtime using a set of domain-specific metrics [RA08]. SACKMANN and KÄHMER propose the ExPDT language for codifying compliance policies which have to be checked by a reference monitor at runtime in order ot validate process compliance [SK08]. RODRIGUEZ et al. propose a method for defining so-called Key Compliance Indicators to monitor running process instances and detect possible future compliance violations. These indicators are applied to decision trees which are mined from process execution logs [RSD+10]. Similarly, LY et al. propose building compliance rule graphs from compliance requirements. The compliance of running process instances is verified by monitoring process traces and checking them against modeled compliance rule graphs [LRK+11].

Some works related to these monitoring approaches stretch several phases of the BPM lifecycle. NAMIRI and STOJANOVIC (cf. Process Design) propose validating process instances during runtime utilizing a knowledge-based approach. This approach utilizes process traces such as log files, message queues and database entries [NS07]. BARESI et al. (cf. System Configuration) developed a framework for checking compliance during runtime employing a set of software components for monitoring ALBERT statements [BBG+07].

Several works have been proposed which consider compliance in terms of detecting business contract violations. These contracts are specified employing (formal) languages and verified at runtime using a variety of techniques. WEIGAND and VAN DEN HEUVEL employ deontic control messages for verifying business contracts [WH02]. MILOSEVIC et al. propose a framework for monitoring process execution to detect compliance violations [MJD+02]. Similar approaches were also proposed by GIBLIN et al. [GMP06] and ALBERTI et al. [ACG+08].

Diagnosis

The diagnosis phase finally deals with finished process instances. Verifying compliance at this stage is a forensic task which is performed analyzing process traces. VAN DER AALST et al. proposed to analyze log files for this task. In their work the authors model compliance rules utilizing linear temporal logic. Compliance is verified employing model checking [ABD05].

The author is not aware of any approaches other than the one presented by VAN DER AALST et al for verifying compliance at this phase of the BPM lifecycle. Considering the circumstance that compliance of process instances should be ensured no later than at runtime, the reason seems obvious. Detecting compliance violations of finished process instances is usually an act of evidence gathering for legal disputes. Thus at this stage of the BPM lifecycle, there is a smooth transition from compliance to IT forensics.

Summary

It has been shown that the classical dichotomy between design-time, runtime and backward compliance checking is insufficient to classify approaches for business process compliance. Furthermore it became clear that at each phase of the BPM lifecycle different compliance aspects can be considered. The methodical consequences for service composition is that not all compliance aspects can be considered during service selection and that instead techniques need to be aligned with compliance checking measures taking place during process design. Table 2.2 summarizes these findings.

Table 2.2: *Verifiability of compliance aspects at different stages of the BPM lifecycle*

Aspect	Compliance Checking Time				
	Process Design	System Configuration		System Enactment	Diagnosis
		Service Selection	Post Transformation		
Activities	●	○	●	●	●
Data	◐	◐	●	●	●
Location	◐	●	●	●	●
Resources	●	○	●	●	●
Time Limits	◐	●	●	●	●

Note: ●= fully verifiable, ◐= partly verifiable, ○= not verifiable

2.5 Optimization Models and Solving Approaches

This section introduces optimization concepts which are utilized in this thesis. After a brief introduction to optimization in general, the concepts of linear and particularly integer linear programming are introduced. Next an overview is given about optimization with multiple objectives, followed by a brief presentation of heuristics and finally genetic algorithms.

2.5.1 Introduction

The goal of real-world *decision* and *planning problems* is usually to either generate a certain output with minimal input or to maximize output for a given input. Hence decision and planning problems are typically characterized by a set of multiple *alternative solutions* as well as by a set of certain *restrictions* which might stem from e. g. technical capacities or natural limits. The task of finding the "best", i. e. *optimal* solution with respect to these restrictions is what constitutes an *optimization problem*. The meaning of optimum highly depends on the problem at hand and might refer to quantities or qualities such as price. An *optimization model* is the formal (usually mathematical) representation of an optimization problem. The application of an optimization model for a concrete data set is called a *problem instance*. The general form of an optimization problem is defined as follows [HL01; DD07]:

$$\text{Maximize (max)/Minimize (min) } z = F(\mathbf{x}) \tag{2.3}$$

subject to (s.t.)

$$g_i(\mathbf{x}) \begin{Bmatrix} \geqslant \\ = \\ \leqslant \end{Bmatrix} 0 \qquad (i = 1, \ldots, m) \tag{2.4}$$

$$\mathbf{x} \in D_1 \times \ldots \times D_n, D_j \in \{\mathbb{R}_+, \mathbb{Z}_+, \mathbb{B}\} \qquad (j = 1, \ldots, n) \tag{2.5}$$

Here, $\mathbf{x}$ is a variable vector with n components $(x_1, \ldots, x_n)$ which are the *decision variables* of the problem. The vector itself is called *solution vector*. A solution vector is being evaluated by an objective function $F(\mathbf{x})$ (2.3) which has to be either maximized or minimized. A set of equalities and inequalities (2.4) is used to describe the restrictions of the problem at hand. Finally, for each component of the solution vector the corresponding domain must be specified (2.5). Each component of the solution vector must not be negative and can be either continuous ($x_j \in \mathbb{R}_+$), discrete ($x_j \in \mathbb{Z}_+$) or binary ($x_j \in \mathbb{B}$).

2.5.2 Linear Programming Problems

Linear Programming (LP) problems are characterized by a linear objective function which has to be either maximized or minimized with respect to linear restrictions. This is also called a Single-Objective Optimization (SOO) problem. Decision variables have to be continuous and usually must not be negative. A minimization problem can be turned into a maximization problem and vice versa by negating the objective function as well as the restrictions (cf. [DD07, p. 17] for a detailed description). Hence, any LP problem can be written in the following general form [HL01; DD07]:

$$\max F(x_1, \ldots, x_n) = \sum_{j=1}^{n} u_j x_j \tag{2.6}$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i \qquad (i = 1, \ldots, m) \tag{2.7}$$

$$x_j \geqslant 0, x_j \in \mathbb{R}_+ \qquad (j = 1, \ldots, n) \tag{2.8}$$

In this formulation, u_j denotes the utility associated with the j-th decision variable from an output perspective, w_j its weighting from an input perspective and c_i a capacity constraint.

A vector $\mathbf{x}$ which fulfills all restrictions (2.7) is called a *solution* of the LP problem. If $\mathbf{x}$ furthermore fulfills (2.8), it is called a *feasible solution*. A feasible solution $\mathbf{x}^*$ is called an *optimal solution* of a given LP problem if no other feasible solution $\mathbf{x}$

yields a greater function value than $F(\mathbf{x}^*)$. While LP problems usually have one optimal solution, some problem instances might have multiple optimal solutions. In this case the problem is said to have a *parametric solution*. Given two optimal solutions $\mathbf{x}^1$ and $\mathbf{x}^2$, all convex combinations obtained by

$$\mathbf{x} = \lambda \mathbf{x}^1 + (1 - \lambda)\mathbf{x}^2, 0 < \lambda < 1$$

are also optimal solutions of the problem instance [DD07].

One of the most powerful and universal methods for solving LP problems is the Simplex method. To put it briefly, the Simplex method explores the boundary of an optimization problem's feasible area for an optimal solution. Since the method has been thoroughly studied in science and is widely used in practice, it is covered in-depth in the standard literature [HL01; DD07].

2.5.3 Integer Linear Programming

An Integer Linear Programming (ILP) problem is a LP problem with discrete or binary decision variables. More commonly, these problems are called Integer Programming (IP) problems in the literature. From the authors point of view however, this does not sufficiently reflect the relationship between LP and ILP problems. Hence, in this thesis the term ILP will be used. In case that the problem uses discrete as well as continuous decision variables, it is called a Mixed Integer Programming (MIP) problem [HL01; DD07].

Many combinatorial problems can be formulated with an ILP model. One of the most frequently encountered problems which is modeled as an ILP problem and which has many real-world applications is the Knapsack Problem (KP). A classical 0-1 KP considers on the one hand a set of items, each having a weight and an utility value and on the other hand a knapsack with a certain capacity. The challenge is then to find the subset of items with maximum utility while respecting the capacity of the knapsack. In its basic form the classical 0-1 KP is defined as follows [MT87]:

$$\max z = \sum_{j=1}^{n} u_j x_j \tag{2.9}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \leqslant c \tag{2.10}$$

$$x_j \in \mathbb{B} \qquad (j = 1, \ldots, n)$$

Here, u_j and w_j denote the utility and the weight of item j respectively, c the capacity of the Knapsack, n the number of items and x_j represents a binary decision variable which indicates if item j is selected ($x_j = 1$) or not ($x_j = 0$).

The 0-1 KP and also all other ILP problems cannot be solved with the Simplex method since this method requires all decision variables to be continuous. Instead, for solving ILP problems, several exact approaches have been proposed which can be classified into the following groups [DD07]:

1. *Decision tree approaches:* This group includes enumeration approaches such as full and partial enumeration with the Branch-and-Bound algorithm [LD60]. Furthermore, approaches for dynamic programming belong to this group (cf., e. g., [CLR+09, Ch. 15]).
2. *Cutting-plane methods:* Approaches of this group iteratively improve the solution set of a given optimization problem by adding additional linear inequalities to the model (so-called *cuts*). Especially for MIP problems, such algorithms are popular for finding integer solutions. An introductory to this topic is given in e. g. [Cor08].
3. *Hybrid approaches:* These approaches combine principles from the former two groups. One prominent example is the Branch-and-Cut algorithm [PR91].

All exact approaches have in common that they guarantee to find an optimal solution after a finite number of steps. While this is undoubtedly a convenient property, employing exact approaches is not feasible in the following cases:

- **Large problem size:** A given problem instance might be too large to be handled on given computing capacities. The notion of "too large" however highly depends on the problem complexity. Especially for NP-hard problems exact approaches are usually infeasible.
- **Time constraints:** A solution might be required in a certain period of time. Exact algorithms however cannot give any guarantees regarding running time.

In case that exact approaches cannot be applied to an optimization problem, either because the problem size is too large or because of strict time constraints, heuristics are preferred (cf. Section 2.5.5). Another important area of application for heuristics are Multi-Objective Optimization (MOO) problems (cf. Section 2.5.4) which differ fundamentally from SOO problems in certain aspects [Deb01; HL01; DD07].

2.5.4 Optimization with Multiple Objectives

The most essential feature of MOO problems is that at least two conflicting objective functions need to be optimized simultaneously. Having several conflicting objectives is a trait which many real-world optimization problems share. A simple example is selecting a car with maximum performance in terms of horsepower and minimal fuel consumption. However, the major drawback is the complexity involved with considering multiple objective functions simultaneously. Particularly, the notion of optimality is different for MOO problems than for SOO problems. Whereas SOO problems typically have one optimal solution (if any at all), MOO problems usually have multiple Pareto optimal solutions in case of discrete decision variables. If any decision variable is continuous, MOO problems even have infinite Pareto optimal solutions. This requires to reconsider the notion of optimality in the context of MOO problems [Deb01; CLV07; BDM+08].

With respect to the general form of optimization problems as defined in Section 2.5.1, the general form of a MOO problem only differs regarding the objective functions and is defined as follows [Deb01; CLV07]:

$$\max/\min F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \tag{2.11}$$

s.t.

$$g_i(\mathbf{x}) \left\{ \begin{matrix} \geqslant \\ = \\ \leqslant \end{matrix} \right\} 0 \qquad (i = 1, \ldots, m) \tag{2.12}$$

$$\mathbf{x} \in D_1 \times \ldots \times D_n, D_j \in \{\mathbb{R}_+, \mathbb{Z}_+, \mathbb{B}\} \qquad (j = 1, \ldots, n) \tag{2.13}$$

For MOO problems, the same rules as for SOO problems apply for converting maximization problems into minimization problems and vice versa. Hence, in the following the focus will be on maximization problems without any loss of generality.

Let Ω denote the set of feasible solutions of a given MOO problem. Then each solution $\mathbf{x} \in \Omega$ represents a compromise with respect to the objective functions. That is, a solution may yield better than average results for one objective, but perform worse than average for the remaining objectives. Thus, the solutions in Ω are partially ordered which is expressed by the *dominance relation* [Deb01; CLV07]:

Definition 2.17 [Dominance Relation]. *A solution $\mathbf{x} \in \Omega$ is said to dominate another solution $\mathbf{x}' \in \Omega$ (denoted $\mathbf{x} \preceq \mathbf{x}'$) iff:*

1. *$\mathbf{x}$ is no worse than $\mathbf{x}'$ in all objectives, i. e.: $\forall i \in \{1, \ldots, n\}, f_i(\mathbf{x}) \geqslant f_i(\mathbf{x}')$*
2. *$\mathbf{x}$ is strictly better than $\mathbf{x}'$ in at least one objective, i. e.: $\exists i \in \{1, \ldots, n\}, f_i(\mathbf{x}) >$*

$f_i(\mathbf{x}')$

Given this property of Ω, a solution $\mathbf{x} \in \Omega$ is regarded as optimal, if no component of the solution vector can be further improved without deteriorating at least one other component. Such a solution is called *Pareto optimal*. Utilizing the dominance relation, Pareto optimality can be defined as follows [CLV07]:

Definition 2.18 [Pareto Optimality]. *A solution $\mathbf{x} \in \Omega$ is called Pareto optimal if it is not dominated by any other solution $\mathbf{x}' \in \Omega$, that is: $\neg\exists\mathbf{x}' \in \Omega . F(\mathbf{x}') \preceq F(\mathbf{x})$. The set of all Pareto optimal solutions is denoted by $\mathcal{P}^*$. The representation of Pareto optimal solutions in objective space is called the Pareto Front $\mathcal{PF}^*$, i. e.: $\mathcal{PF}^* := \{z = F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\}$.*

Figure 2.18 illustrates the relationship of above terms based on a fictitious MOO problem consisting of two objective functions which have to be maximized. Here, the red line indicates $\mathcal{PF}^*$ of the given problem instance. The solutions $\mathbf{x}^1$ and $\mathbf{x}^2$ are Pareto optimal since they are not dominated by any other solution. That is, neither violates any restriction nor can be further improved with regards to one objective function without deteriorating its function value for the other objective function. Solution $\mathbf{x}^3$ is dominated by $\mathbf{x}^1$ and $\mathbf{x}^2$ since both solutions yield better objective function values. Finally, $\mathbf{x}^4$ represents an infeasible solution and is thus not considered in terms of Pareto optimality although it yields better results than $\mathbf{x}^1$ and $\mathbf{x}^2$ for each objective function.

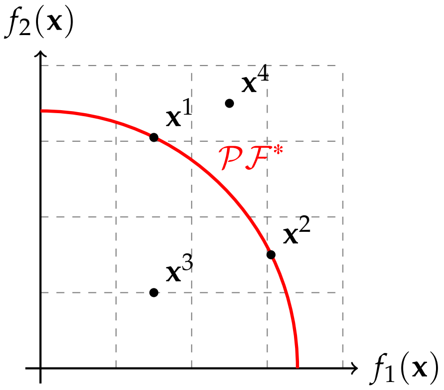


Figure 2.18: *Sample MOO problem instance*

While Pareto optimality of a solution can be unambiguously determined utilizing the dominance relation, it is not possible to decide per se which Pareto optimal solution is "the best". As already stated above, each solution represents a compromise with respect to the objective functions. Hence it cannot be decided automatically which compromise to choose. Instead a decision-maker needs to manually pick a solution, either based on individual preferences or on some set of (organizational) rules [Deb01; CLV07]. DEB proposed what he called an "ideal multi-objective optimization procedure"[Deb01, p. 4] consisting of two steps [Deb01]:

1. Find multiple Pareto optimal solutions with a wide range of values for objectives.
2. Choose one of the obtained solutions using higher-level information.

The first step is especially a challenge for rather complex MOO problems where finding even a single Pareto optimal solution is time consuming. Hence GA techniques are often employed in the context of MOO problems for finding a set of diverse and near-optimal solutions in a comparably short period of time (cf. Section 2.5.6). This does not necessarily mean finding all Pareto optimal solutions but usually refers to determining only a certain number of solutions up to a predefined maximum.

An alternative solving approach is to reduce MOO problems to SOO problems. While this reduces the mathematical complexity, a general disadvantage of this approach is that it does not allow for thoroughly exploring the search space. In particular, some Pareto optimal solutions are usually missed depending on the employed technique. In these cases a thorough search requires multiple applications of the method with no guarantee that all Pareto optimal solutions will be obtained. Several reduction methods have been proposed. Frequently employed are the weighted sum method and lexicographic goal programming. In the following these two methods will be briefly explained. For a detailed overview, discussion and classification of available techniques, the interested reader is referred to the corresponding standard literature such as [Deb01; CLV07; DD07].

Weighted Sum Method

Given a MOO problem with k objectives, the weighted sum method aggregates all objective functions into a single objective function $\Phi(\mathbf{x})$ by multiplying each objective $f_i(\mathbf{x})$ with a real number $0 \leqslant \lambda_i \leqslant 1$ such that $\sum_{i=1}^{k} \lambda_i = 1$.

As an example, let us consider a MOO problem with three objective functions which have to be maximized (cf. [DD07]):

$$\max f_1(x_1, x_2) = 10x_1 + 20x_2 \tag{2.14}$$

$$\max f_2(x_1, x_2) = x_1 + x_2 \tag{2.15}$$

$$\max f_3(x_1, x_2) = 60x_1 + 40x_2 \tag{2.16}$$

$$\text{s.t. } x_1 + x_2 \leqslant 100 \tag{2.17}$$

$$6x_1 + 9x_2 \leqslant 720 \tag{2.18}$$

$$x_2 \leqslant 60 \tag{2.19}$$

$$x_1, x_2 \in \mathbb{R}_+ \tag{2.20}$$

Given three objective weightings $\lambda_1 = \frac{1}{10}$, $\lambda_2 = \frac{8}{10}$ and $\lambda_3 = \frac{1}{10}$, the objective functions (2.14) – (2.16) are aggregated into a single objective function as follows:

$$\begin{aligned} \max \Phi(x_1, x_2) &= \frac{1}{10} f_1(x_1, x_2) + \frac{8}{10} f_2(x_1, x_2) + \frac{1}{10} f_3(x_1, x_2) \\ &= 7.8x_1 + 6.8x_2 \end{aligned}$$

Tackling $\Phi(x_1, x_2)$ with methods such as Simplex, one obtains the optimal solution $\mathbf{x}^* = (60, 40)$ which yields the objective function values $f_1^*(\mathbf{x}^*) = 1400$, $f_2^*(\mathbf{x}^*) = 100$ and $f_3^*(\mathbf{x}^*) = 5200$.

The advantage of the weighted sum method is that it is intuitive and easy to use. However, the method also yields a number of disadvantages for ILP problems with multiple objectives. The major disadvantage is that this method is usually not able to find more than one optimal solution for given weightings. Only in case that chosen weightings lead to a parametric solution, multiple optimal solution can be obtained. In case of MIP and non-linear optimization problems, this method yields several more disadvantages. But since these topics are out of the scope of this thesis, they will not be further discussed here [Deb01; DD07].

Lexicographic Goal Programming

This approach first requires the decision-maker to order objectives according to their priorities. For two given objectives A and B, the decision maker needs to decide whether A has higher priority than B (denoted $A \gg B$) or if B has higher priority than A (denoted $B \gg A$). A goal with a higher priority is regarded as infinitely more important than goals with a lower priority. Hence, this approach first exclusively considers the objective with the highest priority and the corresponding constraints for optimization. If the result is a parametric solution, the objective with the second highest priority and corresponding constraints are considered. Goals of the first objective are used as hard constraints and solutions are required not to violate these goals. Ultimately, this approach will result in a single Pareto optimal solution for a given MOO problem. The obtained solution however depends on the chosen order for the objectives. Especially in case of MOO problems consisting of many objective functions, testing all combinations might become a time consuming task [Deb01; DD07].

As a demonstration, this method will be applied to the objective functions (2.14) – (2.16). Assigning either f_1 or f_3 the highest priority will lead to an optimal solution in a single step without any further consideration of the remaining objectives. If however the highest priority is assigned to f_2, the result is a parametric solution. Depending on whether f_1 or f_3 is assigned the second highest priority, a different optimal solution will be obtained. Applying lexicographic goal programming to this MOO problem thus yields a total of four Pareto optimal solutions depending on the priority order (the "..." indicate that no further objective functions are considered):

$$
\begin{aligned}
f_1 \gg \ldots : \mathbf{x}^* &= (30,60); f_1^*(\mathbf{x}^*) = 1500, f_2^*(\mathbf{x}^*) = 100, f_3^*(\mathbf{x}^*) = 4200 \\
f_2 \gg f_1 \gg \ldots : \mathbf{x}^* &= (60,40); f_1^*(\mathbf{x}^*) = 1400, f_2^*(\mathbf{x}^*) = 100, f_3^*(\mathbf{x}^*) = 5200 \\
f_2 \gg f_3 \gg \ldots : \mathbf{x}^* &= (100,0); f_1^*(\mathbf{x}^*) = 1000, f_2^*(\mathbf{x}^*) = 100, f_3^*(\mathbf{x}^*) = 6000 \\
f_3 \gg \ldots : \mathbf{x}^* &= (100,0); f_1^*(\mathbf{x}^*) = 1000, f_2^*(\mathbf{x}^*) = 100, f_3^*(\mathbf{x}^*) = 6000
\end{aligned}
$$

2.5.5 Heuristics

In contrast to exact approaches, heuristics are usually capable of finding a "good" feasible solution in a comparably short period of time. The major drawback is however, that in general heuristics cannot guarantee to find an optimal solution. In order to successfully find a "good" feasible solution, heuristics incorporate procedure rules which are known or assumed to be successful in the scope of a certain problem structure. Heuristics can be broadly categorized into *constructive* and *local search* heuristics (for a more thorough discussion cf. [DD07]) [BK05].

Constructive heuristics aim at generating a first feasible solution for a given optimization problem. The quality of the resulting feasible solution greatly depends on the construction rules applied. Thus, results can vary from "very bad" to close to optimal [BK05; DD07].

Local search heuristics usually start with a feasible solution which has been determined either randomly or by employing a constructive heuristic. Starting from this initial solution, other solutions nearby (so-called *neighbors*) are considered for potential replacement. If a neighbor is accepted as replacement, the algorithm "moves" to this new solution. This procedure is repeated until a pre-defined termination criterion is reached. Termination criteria might be a certain period of time which has passed since starting the heuristic or a certain number of consecutive iterations without any improvement of the solution. An example for a common local search heuristic is hill climbing [BK05; DD07].

Constructive and local search heuristics can be further classified into *deterministic* and *stochastic* or *randomized* approaches. Deterministic approaches are characterized by the fact that when applied to a certain optimization problem with identical starting conditions, multiple applications will always yield the same solution. On the contrary, stochastic heuristics usually yield different results due to a random component [DD07].

Some heuristics such as hill climbing tend to get stuck in a local optimum. In case of hill climbing, the reason for this behavior is the replacement rule which dictates that a neighbor solution is only considered if it yields better results in terms of the objective function. Hence in order to avoid getting stuck in a local optimum, heuristics are required to allow for a temporary worsening of the solution in order to explore the search space more thoroughly. Such a guiding principle or master-strategy which can be applied to improve the overall results of other heuristics is called *metaheurisic*. An example for such metaheuristics are *genetic algorithms* [BK05; DD07].

2.5.6 Genetic Algorithms

A GA is a stochastic metaheuristic which mimics evolutionary principles of nature for search and optimization procedures[41]. Instead of a single solution, a set of solutions is generated. In case of optimization problems having a single optimal solution, this set can be expected to converge towards this optimum. In case of multiple optimal solutions as is the case for MOO problems, GAs can be expected to obtain multiple optimal solutions in their final set. However, due to the stochastic nature of GAs, it cannot be guaranteed that any optimal solution will be found at all. The set $\mathcal{P}$ of solutions that a GA obtains is thus hoped to hold $\mathcal{P}{=}\mathcal{P}^*$ or $\mathcal{P}{\subset}\mathcal{P}^*$, but this is usually not the case. In this regard the only guarantee is that all solutions in $\mathcal{P}$ are not dominated by any other solution found during the optimization process. Hence the resulting front of $\mathcal{P}$ is called the non-dominated front $\mathcal{NF}$[42] [FF93; Bäc96; Deb01; CLV07].

The basic component of a GA is an *individual* which is an encoded solution to

[41] For a detailed discussion of the biological foundations and their application to the field of computing cf. [Bäc96].

[42] In [CLV07] $\mathcal{P}$ and $\mathcal{NF}$ are denoted $\mathcal{P}_{known}$ and $\mathcal{PF}_{known}$ respectively. From the author's point of view however this notation is misleading since it does not sufficiently reflect that $\mathcal{P}$ and $\mathcal{NF}$ do not necessarily contain any Pareto optimal solution.

some problem. The encoding of a solution corresponds to a biological *genotype*. It typically consists of a single *chromosome* corresponding to the solution vector for a given optimization problem represented as string. This chromosome consists of multiple *genes* which can take on certain values or *alleles* from a certain genetic alphabet. Usually this genetic alphabet is either binary or continuous in the context of GAs. In the simplest case each gene corresponds to a single component of the solution vector for an optimization problem. However, different encodings are also possible[43]. As in biology, the genotype of an individual defines its *phenotype* which in the context of GAs is the concrete objective function value(s) that the solution yields. The set of all individuals is called a *population*. Starting with an initial population, a GA evolves the population iteratively towards increasingly better regions of the search space by applying *genetic operators* on the population, namely *selection*, *crossover*[44] and *mutation*. Each iteration of a GA produces a new *generation* of individuals. In conjunction with genetic operators, probabilistics are used for a better *exploration* of the search space. Probabilistics are step-wise reduced in order to achieve a better *exploitation* of discovered good areas of the search space. Furthermore, GAs utilize a diverse set of techniques such as distance measures to preserve *diversity* among the population. This aims at obtaining an equal distribution of the population along or close to $\mathcal{PF}^*$. The process ends as soon as some *termination condition* is met [Bäc96; Deb01; CLV07].

Figure 2.19 shows the working principle of GAs in a general form as flowchart. In the following, essential concepts in the scope of GAs will be briefly presented and discussed based on this depiction. It should be noted however, that real-world GAs usually differ from this form, either by using a different sequence of genetic operators or by performing additional steps. Furthermore, for each genetic operator a wide array of different strategies and modifications have been proposed. Hence this general form along with the following description do by no means represent the only true form of GAs but aim at providing a foundation for understanding the GA which has been developed in the scope of this thesis (cf. Section 5.3) and the reference GAs against which it was tested (cf. Chapter 7).

The first step of a GA is to generate the initial population of individuals. Two essential questions in this regard are how to generate individuals and how many. Individuals of the starting population are usually generated randomly since GAs are intended to be general purpose. However, customized GAs for specific prob-

[43] For instance, in [Deb01] a problem instance is presented with a solution vector consisting of two discrete components. Each decision variable is encoded using five binary genes.

[44] Frequently the term *recombination* is used synonymously (cf., e. g., [Bäc96; CLV07]). In this thesis however the term crossover will be used exclusively since this is also the name of the biological process which this operator mimics.

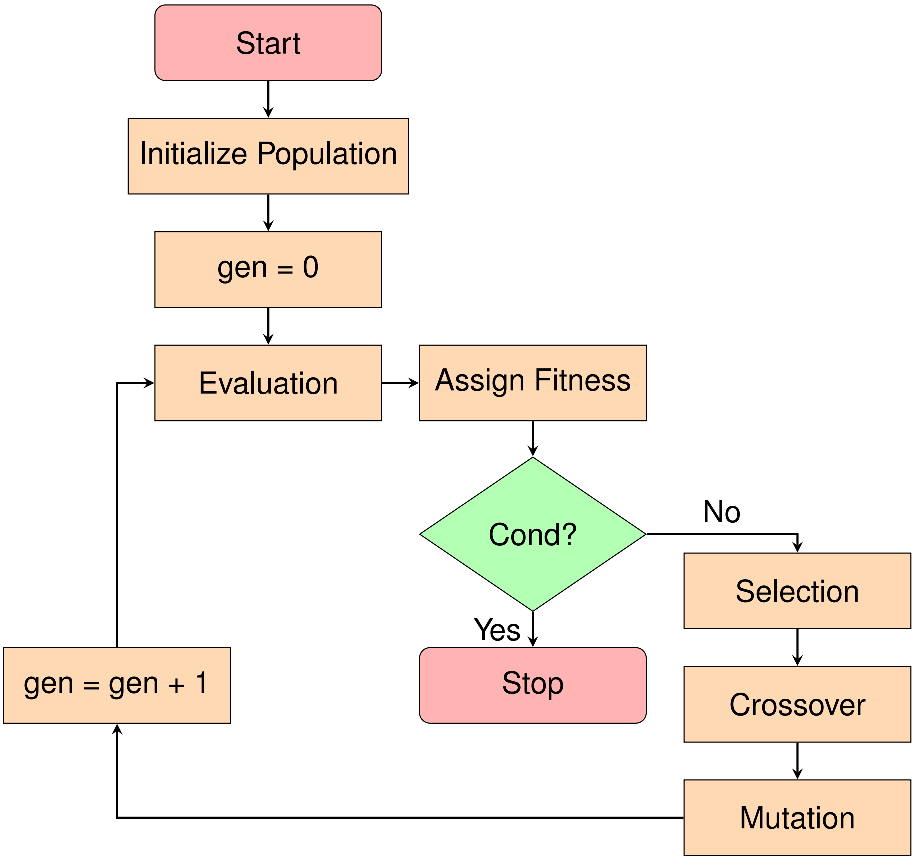


Figure 2.19: *The working principle of a GA as flowchart (based on [Deb01])*

lems sometimes utilize heuristic approaches to (partly) generate a good initial population (cf., e. g., [KIH12]). This may improve the speed of convergence and the quality of the final solutions [ZDT00; BGK04]. With regards to the population size, the question is not easy to answer. A “small” population size is usually expected to deliver poor solutions while a “large” population size may cause the GA to spend too many resources [Ala92]. PISZCZ and SOULE proposed to choose the size of the population depending on the problem complexity. However, determining the optimal population size quickly becomes very difficult for increasingly complex optimization problems [PS06]. Hence, as a rule of thumb, populations with a size of 50 to 200 individuals are commonly used today in conjunction with most GAs [CLV07].

At each generation, the population is being evaluated. This evaluation consists of computing the objective function value(s) that each individual yields as well as counting the constraints that are violated by each individual. Based on this information, a *fitness function* computes a *fitness value* for each individual. In other words, the fitness value is a metric expressing the desirability of an individual. The definition of the fitness function depends on the concrete GA [Bäc96; Deb01; CLV07].

After the population has been evaluated, the GA checks if a pre-defined termination condition is satisfied. If this is the case, the GA terminates and otherwise the population is modified by applying genetic operators which results in a new generation. This new generation is again evaluated before rechecking the termination condition and so on. In the simplest case the termination condition is a strict generation limit, i. e. the GA automatically terminates after a certain number of generations. More sophisticated termination conditions are *convergence criteria* which can also be combined with a generation limit. For a known $\mathcal{PF}^*$ of a given optimization problem, convergence criteria measure the distance of the population to $\mathcal{PF}^*$. If $\mathcal{PF}^*$ is unknown, quality indicators are utilized to measure the improvement of a population from one generation to the next. In this case, convergence relates to the continuous improvement of the population towards a certain quality value. The termination criterion then is usually a minimum improvement that every new generation must yield. Otherwise the GA terminates. Several quality indicators have been proposed (cf., e. g., [CLV07] for a thorough discussion). A frequently utilized quality indicator is the *Hypervolume* (denoted $\mathcal{HV}$) which measures the dominated portion of objective space [ZT99; Deb01; CLV07]. Figure 2.20 shows an example for a fictitious MOO problem with two objective functions which have to be maximized (the corresponding $\mathcal{PF}^*$ is depicted by the red line). The points $\mathbf{x}^1, \ldots, \mathbf{x}^4$ represent a fictitious solution, i. e., a $\mathcal{NF}$ (depicted by the blue line), as it might be obtained from a GA. The grey shaded area finally represents the corresponding $\mathcal{HV}$.

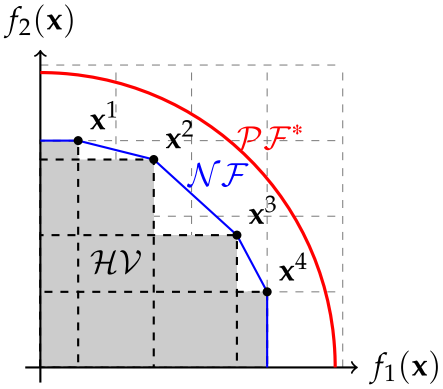
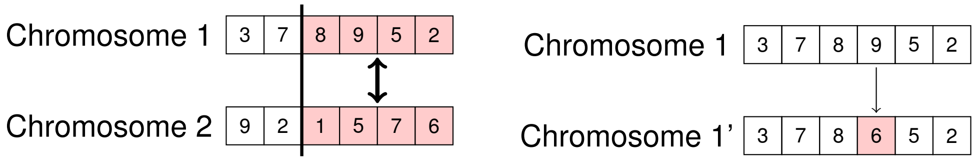


Figure 2.20: *Hypervolume of a population of solutions for a MOO problem*

Selection operators are utilized to decide, which individuals of the current population will survive to be part of the next generation. This decision is usually based on the fitness value and follows a certain strategy. A common strategy is to rank solutions according to their fitness value by performing pairwise comparisons. From this ranking, a certain number of individuals is selected as members of the next generation. The pairwise comparison approach is called *binary tournament* and it belongs to the class of *elitist* selection operators. In order to preserve diversity among the population, a common strategy is to randomly select a certain number of bad solutions, especially at earlier generations [Bäc96; Deb01; CLV07].

The crossover operator aims at producing new individuals by combining the chromosomes of two parent individuals. In combination with the selection operator, which sorts out individuals from the population, the purpose of crossover is thus to fill up the resulting gap in the population with new individuals. In its most basic form, crossover works by first selecting a cutting point in two chromosomes. After that, the subsequences of genes, from the cutting point to the end of the chromosome, are exchanged (cf. Figure 2.21(a)). This is also called *single-* or *one-point crossover* [Bäc96; Deb01; CLV07].

Mutation randomly changes one or more genes in a chromosome at a certain probability. This may prevent potential stagnation of the population in a local optimum and lead to a better exploration of the objective space. Chosen genes are replaced with random alleles (cf. Figure 2.21(b)). One frequently employed mutation operator is the *non-uniform mutation*. This operator is characterized by a mutation probability which decreases with each generation. The strategy behind this behavior is to better exploit discovered feasible areas in objective space at later generations [Bäc96; Deb01; CLV07].



(a) Crossover

(b) Mutation

Figure 2.21: *Working of basic genetic operators*

2.6 Summary

This chapter introduced a broad spectrum of topics which form the conceptual and methodical foundation this thesis is built upon. Due to the mass of available literature on certain topics, a representative selection of works have been discussed. The selection has been performed such that each direction of research with relevance for this thesis has been covered. A particular result of this chapter is that interdependent security and business process compliance have not been covered so far from a service selection perspective.

Chapter 3

Service Composition

"The whole is greater than the sum of its parts."

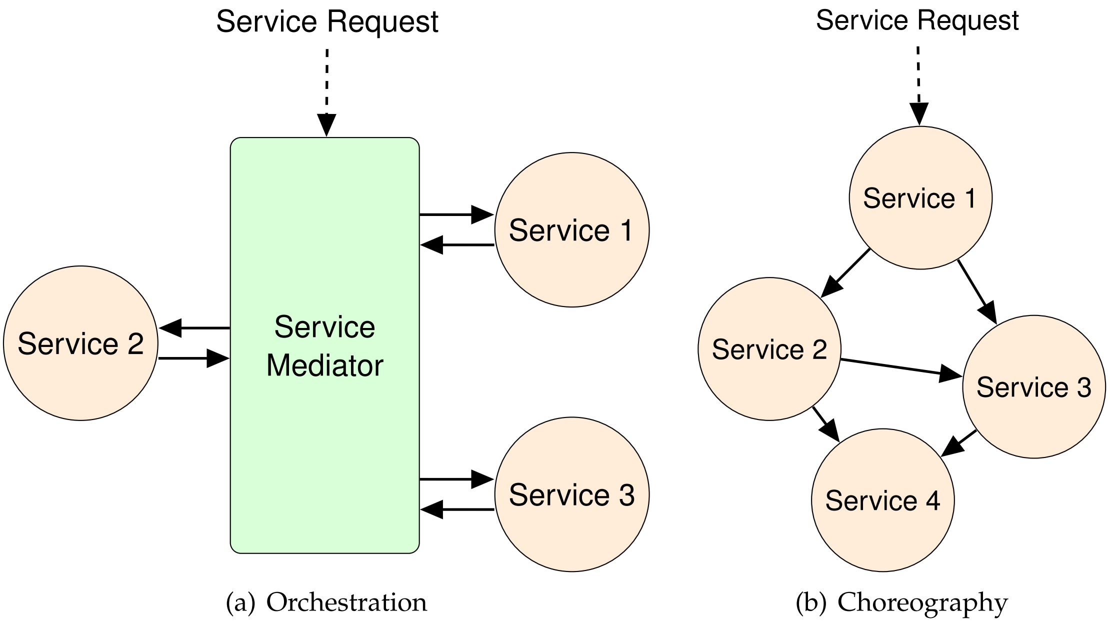
– Aristotle

In this chapter service composition will be discussed which is the main focus of the work at hand. Conceptually, service composition belongs to the 2nd phase of the BPM lifecyle, i. e. system configuration (cf. Section 2.1.1). Hence, service composition includes all activities which are necessary to transform a process model into an executable composition of services. At the network level, such a composition can be implemented with different communication patterns for participating services. This results in different composition topologies which will be discussed first (Section 3.1). The transformation itself can be performed employing different approaches. A majority of these approaches are collectively referred to as *QoS-aware service selection*[45]. Since this thesis contributes to service selection, this topic is discussed next (Section 3.2). Subsequently covered are the the consideration of process requirements regarding security (Section 3.3) and compliance (Section 3.4) during the transformation. Besides of service selection, alternative composition approaches exist which will be shortly presented afterwards (Section 3.5). The chapter closes with a brief summary (Section 3.6).

3.1 Composition Topologies

Services can be composed in two fundamentally different ways, namely as *orchestration* and as *choreography*. Each type of composition leads to a different kind of service topology [Erl05].

[45] The term *service selection* is used interchangeably in this thesis.



(a) Orchestration (b) Choreography

Figure 3.1: *Service composition topologies*

In an orchestration one service acts as a mediator and centrally coordinates the invocation of other services (cf. Figure 3.1(a)). This coordination is based on user inputs as well as service outputs and follows some formalism such as BPEL (cf. Section 2.1.2). The mediator is also responsible for implementing service states [Erl05]. The foremost advantages of this approach are that the mediator as the central instance is in control of all service invocations and data delegation while the invoked services do not need to know each other. On the other hand, the mediator is at the same time a single point of failure which in case of an outage causes the whole composition to crash. Another drawback is that in the face of multiple process instances, the mediator may become a bottleneck causing delays in process execution. The languages for implementing business processes discussed in Section 2.1.2 are all service orchestration languages. It should be noted that here the term 'central' refers to the logical system layer. On the physical layer, the mediator may run on multiple distributed physical machines.

A choreography is characterized by the circumstance that no central mediator is present, but every participating service may act as mediator and recipient at the same time (cf. Figure 3.1(b)). Participating services fulfill their activity as they are invoked by some other service and may themselves invoke other services. Effectively this leads to a peer-to-peer topology [Erl05]. Such a distributed execution of processes eliminates the problem of a single point of failure and furthermore allows for collaborative process modeling. In particular, this enables participants in a process to model their individual subprocesses independently from each other. On the flipside however, choreographies require services to

Table 3.1: *Advantages and disadvantages of composition topologies*

	Orchestration	Choreography
Advantages	• Mediator as central instance controls all service invocations and delegates all process data among the services • Single service don't need to know each other	• Distributed execution eliminates the problem of single point of failure • Enables collaborative process modeling allowing partners to model their local processes independently from each other
Disadvantages	• Outage of the mediator causes outage of the whole composition (single point of failure) • In case of many process instances, the mediator may become a bottleneck causing delays in process execution	• Services need to know all services they want to interact with • Local changes to process models may lead to inconsistencies or errors regarding the interaction with other collaboration partners • Logging and determination of error sources is usually more difficult in a distributed environment

know the partner services they are interacting with. Another issue is that local changes to process models may lead to inconsistencies or errors regarding the interaction with other collaboration partners. Logging and determining the source of errors in such distributed environments is also usually more difficult than in a centralized environment [RW12]. Service choreographies are not as common as service orchestrations. Accordingly there are only few languages supporting this kind of service composition. Alternatives include academic proposals such as Let's Dance [ZBD+06] and BPEL4Chor [DKL+07]. Furthermore, the W3C maintains a standardized language called Web Services Choreography Description Language (WS-CDL) [W3C05].

Table 3.1 briefly summarizes the advantages and disadvantages of both composition topologies. Especially due to the central mediator in orchestrations which allows to control the execution of processes and thus yields verifiability, orchestrations are clearly preferred for business applications. In contrast, the difficulties associated with collaboratively executing processes, hindered a broad adoption of choreographies [RW12]. Currently however, choreographies are being investigated in the context of military settings, particularly at the tactical level. These settings are characterized by requirements on robustness in the face of temporary unavailable network connections, data-driven process execution and real-time constraints on reaction to events [CSN+13]. Since the focus of this thesis is on business applications, it will concentrate on orchestrations in the following.

3.2 QoS-Aware Service Selection

This section firstly motivates service selection with a simple example before discussing how to determine global QoS properties of compositions by means of QoS aggregation. Subsequently it is presented how to normalize QoS of compositions which is a mandatory prerequisite for determining the quality of different alternatives and choosing the "best" solution. Latter is covered in the last subsection where the service selection problem is being introduced and discussed.

3.2.1 Motivation

The motivation and need for service selection will be illustrated with a simple example. Figure 3.2 shows a process model in BPMN for validating customer data as it may occur in e. g. online shops. As soon as the process is triggered, it first validates the provided e-mail address. Next the process validates either the bank account or the credit card number of the customer, depending on the data provided (i. e. the payment method selected by the customer). As soon as these checks are performed, the process is finished.

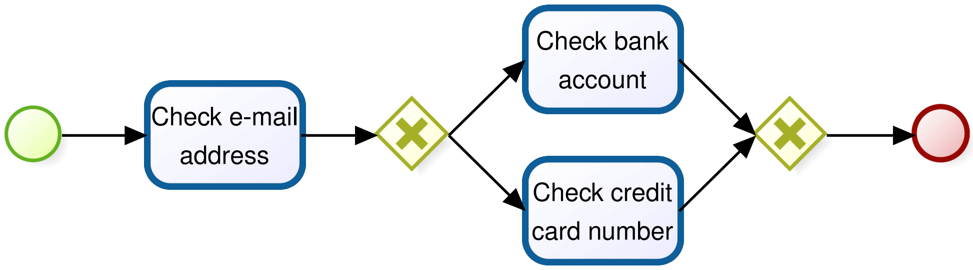


Figure 3.2: *Sample process*

In order to transform this process model into an executable service composition in e. g. BPEL, generally three steps are necessary:

1. *Service Tracking:* At first it is necessary to identify potential service candidates delivering required functionality. Identification of a service's provided functionality can be performed either manually or automatically. For automatic identification, technologies such as Resource Description Framework (RDF) [W3C04a] and Web Ontology Language (OWL) [W3C12] can be employed. This enables uniform description of service capabilities and reduces language barriers[46]. The potential service requester then requires appropriate matchmaking mechanisms in order to interpret this meta-information. The final

[46] The inherent danger of using natural language alone for describing the provided functionality of a web service is that the description might be in a language which the potential service

result of this step is a service repository including relevant service candidates along with their QoS values.

2. *Selection of Services:* As soon as relevant service alternatives are identified, the next step is to determine the optimal assignment of services to tasks. Optimal in this case means the solution which delivers the best QoS values while respecting the QoS constraints of the process.

3. *Transformation:* After the optimal solution is determined, the process model is transformed into an executable form utilizing the selected services.

As stated, each task requires a different kind of functionality which in the following will be referred to as *service class* S_i. For the remainder of this thesis, three simplifying assumptions are made regarding service selection:

1. **Service Atomicity:** Some services might deliver functionality required by more than one task[47]. For the sake of simplicity however, atomicity of services will be assumed, i. e. each service delivers functionality required by exactly one task and can thus be unambiguously assigned to a certain service class. From a granularity point of view, this means that services are assumed to have an equal functional coverage. In fact, considering services with different granularity in a composition leads to economic trade-offs (cf., e. g., [KHH+11; HZ12]). This is however beyond the scope of this thesis.

2. **Interface Uniformity:** Real world services of the same service class usually differ in terms of types, naming and ordering of input as well as output parameters. It is assumed that within a service class each service has a uniform interface in this regard.

3. **Data Homogeneity:** Some real world services (e. g. for encryption and storage) have to deal with different types of data such as text, video and music files. The type of data may have a crucial impact on the performance of a service. For instance, performing asymmetric encryption on video files requires more resources in terms of memory and CPU power than for small text files. Hence the performance of an encryption service will vary for these two different types of data. Thus it is assumed that all data passed to a

requester does not understand. E. g. a web service for checking credit card numbers might be described as “Überprüfung von Kreditkartennummern” in German and “Kredi kartı numarası doğrulama” in Turkish.

[47] In fact, especially the payment checking tasks in the sample process are good candidates for bundling. Web services such as Optimal Payments (`http://www.optimalpayments.com/`) provide functionality for checking bank account and credit card numbers.

Table 3.2: *Service alternatives for sample process*

Service Class	Service Candidate	Price	Response Time	Availability
S_1	s_{11}	2.25	7	0.92
	s_{12}	2.45	4	0.96
	s_{13}	2.30	5	0.93
S_2	s_{21}	2.80	3	0.91
	s_{22}	2.70	2	0.92
	s_{23}	2.65	4	0.95
	s_{24}	2.45	5	0.94
S_3	s_{31}	2.35	4	0.92
	s_{32}	2.55	4	0.94

service is homogeneous and has no significant impact on the performance of a service.

Given the tasks in Figure 3.2, let the corresponding service classes be defined as follows:

- S_1: Check e-mail address
- S_2: Check bank account
- S_3: Check credit card number

Furthermore, a process might be subject to different QoS requirements. For the sake of simplicity, the focus will be on three QoS attributes in this example, namely price, response time and availability. Let the sample process be subject to the following QoS requirements:

- Price $\leqslant 5$
- Response Time $\leqslant 10$ sec
- Availability $\geqslant 90\%$

What is now needed is a repository of service candidates providing required functionality. Table 3.2 provides a list of fictitious service alternatives which fulfill this criterion. For each service class S_i, different service alternatives s_{ij} are available which deliver the same functionality. However, each service alternative s_{ij} delivers its functionality at a different QoS level which will be referred to as QoS vector q_{ij}. Given these service alternatives, the challenge is to find the optimal solution (if any exists) which on the one hand delivers the required functionality and on the other hand respects the global QoS requirements of the process.

For this motivating example, the problem will be considered from a combinatorial perspective (cf. Section 3.2.4). To solve the selection problem, a full enumeration of all possible assignments of services to tasks is performed. The overall QoS of each solution is obtained by inserting the QoS vectors $\{q_{1i}, q_{2j}, q_{3k}\}$ for each triple of services $\{s_{1i}, s_{2j}, s_{3k}\}$ into the following aggregation functions[48]:

$$\Phi_{Price}(q_{1i}, q_{2j}, q_{3k}) = q^1_{1i} + max(q^1_{2j}, q^1_{3k}) \tag{3.1}$$

$$\Phi_{Availability}(q_{1i}, q_{2j}, q_{3k}) = q^2_{1i} * min(q^2_{2j}, q^2_{3k}) \tag{3.2}$$

$$\Phi_{ResponseTime}(q_{1i}, q_{2j}, q_{3k}) = q^3_{1i} + max(q^3_{2j}, q^3_{3k}) \tag{3.3}$$

A total of 24 possibilities exist in which to assign the fictitious service alternatives to the tasks of the sample process (cf. Table 3.3). In general, for a given process model with n tasks (and a corresponding number of service classes), the total number c of possible service compositions is given by the following term:

$$c = \prod_{i=1}^{n} |S_i| \tag{3.4}$$

As can be seen in Table 3.3, the only solution fulfilling all QoS requirements is $\{s_{12}, s_{24}, s_{32}\}$ which offers price 5.00, response time 9 sec and availability 90,24%. Hence, this solution is at the same time the optimum. Having determined the optimal solution, the final step is to transform this selection into an executable form. Figure 3.3 sketches a BPEL process as it may be produced by e. g. a model-driven approach (for an overview about model-driven software engineering cf. [OMG03]).

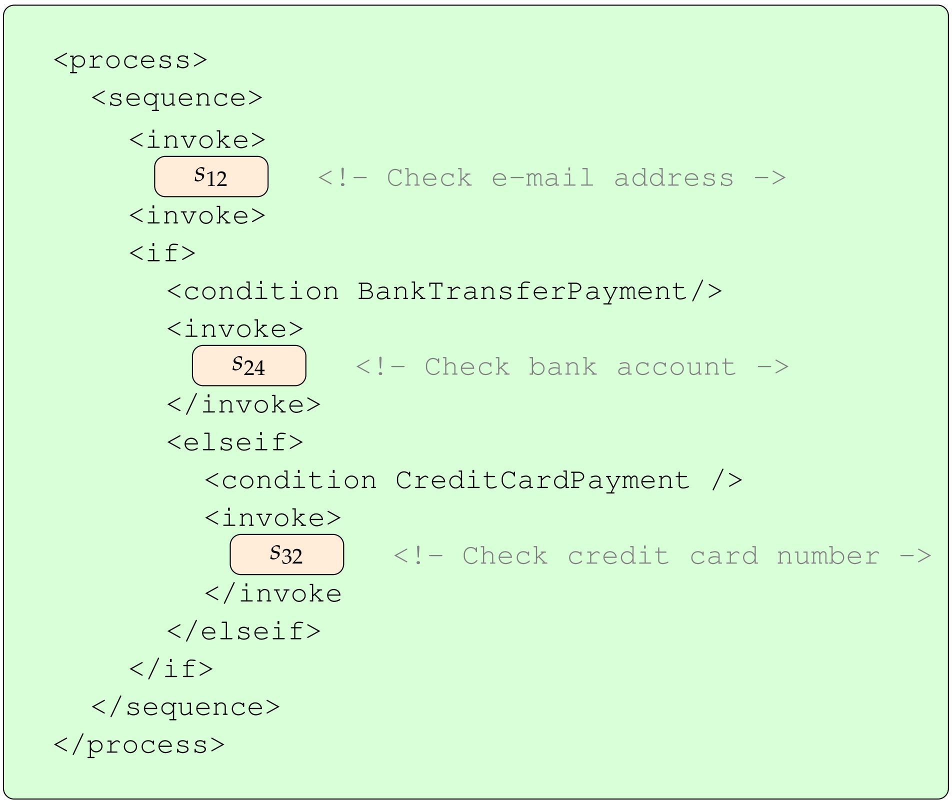
Obviously, enumerating all possible combinations is not an efficient approach for finding the optimal solution, especially for large process models or in the face of tight time constraints. Latter is the case for e. g. autonomic systems which are required to repair themselves automatically in case of failures such as unexpected performance lags of single services in a composition [KC03].

Tackling these situations efficiently requires facilities for dynamic service selection with respect to end-to-end QoS constraints [YZL07]. Determining the global QoS vector of a selection of services however requires aggregating the QoS vectors of each selected service.

[48] For the moment the reader is asked to take these functions for granted. How they are deducted from the process model will be explained in detail in Section 3.2.2.

Table 3.3: *Full enumeration of all possible solutions for the example*

Solution	Price	Response Time	Availability
$\{s_{11}, s_{21}, s_{31}\}$	5.05	11	0.8372
$\{s_{11}, s_{21}, s_{32}\}$	5.05	11	0.8372
$\{s_{11}, s_{22}, s_{31}\}$	4.95	11	0.8464
$\{s_{11}, s_{22}, s_{32}\}$	4.95	11	0.8464
$\{s_{11}, s_{23}, s_{31}\}$	4.90	11	0.8464
$\{s_{11}, s_{23}, s_{32}\}$	4.90	11	0.8648
$\{s_{11}, s_{24}, s_{31}\}$	4.70	12	0.8464
$\{s_{11}, s_{24}, s_{32}\}$	4.80	12	0.8648
$\{s_{12}, s_{21}, s_{31}\}$	5.25	8	0.8736
$\{s_{12}, s_{21}, s_{32}\}$	5.25	8	0.8736
$\{s_{12}, s_{22}, s_{31}\}$	5.15	8	0.8832
$\{s_{12}, s_{22}, s_{32}\}$	5.15	8	0.8832
$\{s_{12}, s_{23}, s_{31}\}$	5.10	8	0.8832
$\{s_{12}, s_{23}, s_{32}\}$	5.10	8	0.9024
$\{s_{12}, s_{24}, s_{31}\}$	4.90	9	0.8832
$\{s_{12}, s_{24}, s_{32}\}$	5.00	9	0.9024
$\{s_{13}, s_{21}, s_{31}\}$	5.10	9	0.8463
$\{s_{13}, s_{21}, s_{32}\}$	5.10	9	0.8463
$\{s_{13}, s_{22}, s_{31}\}$	5.00	9	0.8556
$\{s_{13}, s_{22}, s_{32}\}$	5.00	9	0.8556
$\{s_{13}, s_{23}, s_{31}\}$	4.95	9	0.8556
$\{s_{13}, s_{23}, s_{32}\}$	4.95	9	0.8742
$\{s_{13}, s_{24}, s_{31}\}$	4.75	10	0.8556
$\{s_{13}, s_{24}, s_{32}\}$	4.85	10	0.8742



```
<process>
  <sequence>
    <invoke>
      s12      <!- Check e-mail address ->
    <invoke>
    <if>
      <condition BankTransferPayment/>
      <invoke>
        s24      <!- Check bank account ->
      </invoke>
      <elseif>
        <condition CreditCardPayment />
        <invoke>
          s32      <!- Check credit card number ->
        </invoke
      </elseif>
    </if>
  </sequence>
</process>
```

Figure 3.3: *Sketch of BPEL process incorporating the optimal service selection*

3.2.2 QoS Aggregation

As previously stated, service selection is subject to global QoS constraints as defined by a process designer. In order to obtain the global QoS vector $\mathcal{Q}$ for a selection of services, QoS vectors q_{ij} of each selected service alternative s_{ij} need to be aggregated along the process model. Three major types of approaches have been proposed for this task in the literature about service selection. While sequences, loops and parallelisms are handled mostly identical, they differ in the way how conditional branches such as XOR-splits are treated. Hence in the following these three approaches will be shortly presented with a special emphasis on this difference. A thorough discussion of all details including minor variations and hybrid approaches is however beyond the scope of this thesis.

The first type of approach begins with determining all *execution paths* of a process model prior to performing QoS aggregation on each path separately (cf., e. g., [ZBD+03; YL04; ZBN+04; AZA+05; XB05; AP06; BSR+06; YZB11; HL13]). In case of conditional branches, this means that each branch represents a single path of execution. For instance, the sample process in Figure 3.2 can be decomposed into two different execution paths (cf. Figure 3.4). This approach generally delivers more accurate results in terms of QoS. However, it generally also consumes more resources in terms of computing time and memory since aggregation (and

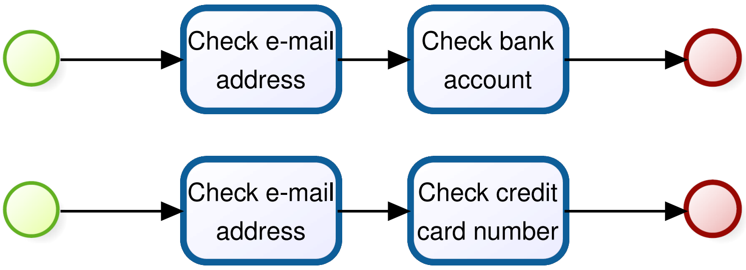


Figure 3.4: *Execution paths of the sample process*

subsequently also service selection) needs to be performed multiple times, i. e. for each execution path. This can become especially critical in cases of big process models containing lots of conditional branches.

In the second type of approach QoS values for all execution paths are aggregated simultaneously. In case of conditional branches however, each branch i is assigned a probability p_i such that $\sum_{i=1}^{n} p_i = 1$ holds for all branches of a conditional gateway (cf., e. g., [Men04; CDE+05; YZL07; CDE+08; WCS+08; MCD10; KIH11; LM11; SPG+11]). Applied to the sample process in Figure 3.2 and assuming equal probabilities for each execution path (i. e. $p_1 = p_2 = 0.5$), the aggregated price for the composition can be determined as follows:

$$\Phi_{Price}(q_{1i}, q_{2j}, q_{3k}) = q_{1i}^{1} + p_1 q_{2j}^{1} + p_2 q_{3k}^{1} \tag{3.5}$$

This type of approach has a major advantage over the first in that all execution paths of a process model are captured implicitly in the aggregation functions. On the other hand however, each conditional branch needs to be associated with a probability value. Depending on the service candidates, this might require continuous monitoring of running process instances in order to obtain accurate probabilities.

The third type of approach finally is similar to the second, i. e. QoS aggregation is also performed for all execution paths simultaneously and all execution paths of a process model are implicitly captured in terms of aggregation functions (cf., e. g., [Lee03; GJ05; CCG+07; JM07; MBK+09; ARN12; YCY12]). In contrast however, it considers only the branch with the worst QoS value at conditional gateways for aggregation. This is performed for each QoS attribute separately. In case of the sample process in Figure 3.2 and the optimal solution, the worst conditional branch in terms of response time was s_{24}. Hence this value went into the aggregation. In terms of price however, s_{32} had the worst (i. e. highest) price. The respective QoS values of the other service were not aggregated. This approach ensures that a solution for the service selection problem (cf. Section 3.2.4) respects the worst (i. e. critical) path for each QoS attribute along the whole composition. Since all other conditional branches yield at least the same or a better QoS value, it is at the same time ensured that these execution paths respect the global requirement for each QoS attribute as well. In the following the optimal solution for the sample process shall be utilized to clarify this. Computing the price for each execution path (denoted as $price_{upper}$ and $price_{lower}$ in correspondence with Figure 3.4) yields the following prices:

$$\begin{aligned} price_{upper} &= 2.45 + 2.45 = 4.90 \\ price_{lower} &= 2.45 + 2.55 = 5.00 \end{aligned}$$

As can be seen, the price for each execution path respects the global QoS constraint of the process. The major advantage of this approach is that probabilities are not required for computing the aggregated QoS of conditional branches. On the other hand however, it lacks the possibility for a differentiated examination of individual execution paths in terms of aggregated QoS. Nevertheless this approach has been applied in the motivation (cf. Section 3.2.1) as well as in the remainder of this thesis for two reasons: 1) The focus of this thesis is not on examination of execution paths and 2) like the second type of approach it aligns more closely with a combinatorial problem formulation for service selection (cf. Section 3.2.4) which is the focus of this thesis. Hence, this decision was rather a simplification.

Considering proposed approaches for QoS aggregation such as [PS97; Car02; YPH02; JRM04; Men04; JRM05], it can be concluded that aggregation of a QoS dimension is subject to two factors:

- *Semantics of QoS Dimension:* QoS dimensions are usually described by means of properties such as direction and value type. However, when aggregating a QoS dimension, different semantics may apply which cannot be logically derived from a QoS dimension's properties. For instance, price and availability are both numerical dimensions. However, aggregating these properties for a sequence is different: Price needs to be added and availability to be multiplied. Another aspect of numerical QoS dimensions is that they might be based on a statistical definition such as variance or mean. In case of composition patterns such as parallelisms, such QoS dimensions would require specific aggregation schemes. Finally, some QoS dimensions such as encryption scheme are not numerical and thus require aggregation schemes based on e. g. a rule-based approach.

- *Composition Pattern:* Composition patterns determine the order in which services are invoked. As mentioned previously, this order might have an influence on the way a QoS dimension needs to be aggregated. It is thus important to determine relevant composition patterns which have an influence on QoS dimensions.

Based upon these observations, QoS aggregation is defined as follows:

Definition 3.1 [QoS Aggregation]. *QoS aggregation is the process of determining the global QoS vector $\mathcal{Q}$ of a service composition by aggregating the QoS vectors of the selected services based upon the semantic of each QoS attribute and the composition patterns the process model consists of.*

QoS aggregation has long been a field of active research. Puschner and Schedl proposed a graph-based approach to determine maximum execution times for distributed real-time systems. Although the work considers several composition patterns and execution sequences, it is limited in the way that it only focuses on a single QoS dimension [PS97]. Menascé proposed a model to determine if selected web services in a composition offer sufficient throughput. The work considers simple sequential and parallel composition patterns [Men02]. An extended version of this work takes time and cost into consideration and also more composition patterns such as forks and joins. The work fails however to identify general structures for aggregating QoS attributes as well as the influence of composition patterns on aggregation. Instead, case-based schemes are proposed [Men04]. Yang et al. differentiate between data oriented, sequential and parallel composition structures [YPH02]. Similar to the works of Menascé this approach lacks in identifying general interrelations as well.

A more general approach was proposed by Cardoso who considers time, cost, reliability and fidelity. On a given process model, a step-wise reduction is performed along certain composition patterns until finally one task is left. At each reduction step, the QoS dimensions of the respective piece of process are being aggregated according to a general aggregation scheme [Car02]. Unfortunately, this approach does not consider all composition patterns found in real-world process models and is limited to certain QoS dimensions as well.

Currently the most general approach has been proposed by Jaeger et al. [JRM04]. The work is based on an analysis of 20 workflow patterns which was conducted by van der Aalst et al. on a total of 15 commercial workflow management systems [AHK+03][49]. Jaeger et al. studied the relevance of these patterns for service compositions and identified a total of seven relevant Composition Patterns (CPs)[50]. In their original work these patterns have been depicted using a generic

[49] In the meantime an expanded and revised version has been published [RHA+06]. Furthermore, a website has been launched providing rich information on this as well as related topics: `http://www.workflowpatterns.com/`

[50] In a later work Jaeger et al. extended this approach to consider service dependencies and QoS violations to obtain more accurate compositions at runtime [JRM05]. But since this thesis

notation explaining that this "can be applied to different composition descriptions" [JRM04, p. 150]. The author has done here so for BPMN with Figure 3.5 showing the result. An impression of how these CPs can be expressed in BPEL can be found in Appendix A.1. In the following these CPs will be briefly explained:

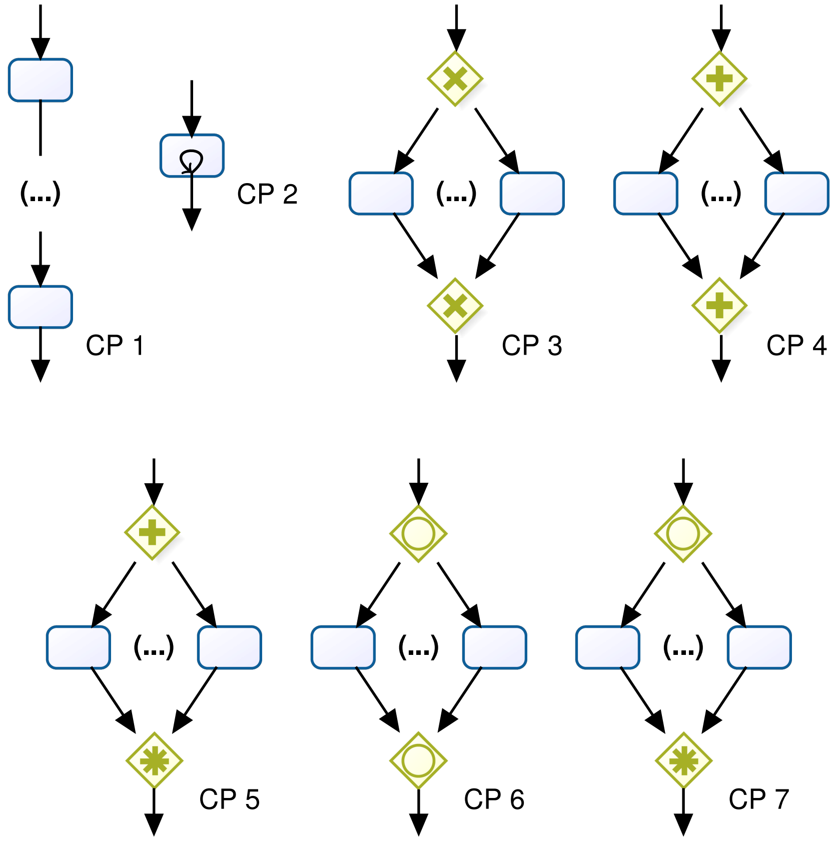


Figure 3.5: *Composition patterns for QoS aggregation (based on [JRM04])*

- **CP 1:** This pattern is a simple sequence of tasks which are executed in the defined order.
- **CP 2:** A loop, potentially with a predefined maximum number of iterations (cf. below). To indicate loops, BPMN offers a loop marker in the form of a curved arrow which is attached to a task [OMG11].
- **CP 3:** XOR-split followed by a XOR-join (cf. Section 2.1.1).
- **CP 4:** AND-split followed by an AND-join synchronizing all parallel flows (cf. Section 2.1.1).
- **CP 5:** AND-split followed by an m-out-of-n AND-join ($1 \leqslant m \leqslant n$). This kind of join synchronizes at least one flow and is also called a discriminator

is about design-time service composition, this approach will not be further investigated here.

[AHK+03]. To support this kind of synchronization behavior, BPMN provides a complex gateway element which is a diamond shape containing a marker in the shape of an asterisk [OMG11].

- **CP 6:** OR-split followed by an OR-join synchronizing all active parallel flows. This kinds of splits and joins are modeled as a diamond shape gateway including a circle [OMG11].
- **CP 7:** OR-split followed by an m-out-of-n OR-join ($1 \leqslant m \leqslant n$). Again the join is a discriminator. As for CP 5, this is also modeled utilizing the complex gateway provided by BPMN [OMG11].

While most CPs are pretty straight-forward, loops and OR-splits require some further explanation due to some specific properties regarding BPMN and QoS aggregation.

The loop marker in BPMN actually leads to repeatedly evaluating a boolean expression. As long as this boolean expression evaluates to `true`, the task is repeated, possibly ad infinitum. Thus, utilizing the loop marker effectively results in a `while`-loop. It is however possible to specify a numeric cap by attaching an intermediate conditional event to the loop task (cf. Figure 3.6). This ensures that as soon as the cap is reached, the event triggers the next task [OMG11]. In either case this means that an aggregation of QoS dimensions which are affected by loops is not exact, but based on an assumption regarding the number of iterations. Employing a cap as shown may however limit the degree of inaccuracy. This issue with loops is not specific to BPMN, but applies to all modeling languages where the number of loop iterations depends on runtime conditions [JRM04].

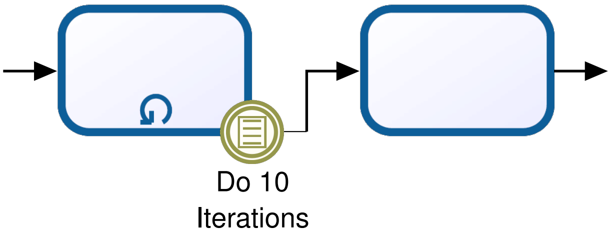


Figure 3.6: *Defining a cap for a loop with an intermediate event in BPEL*

OR-splits lead to an aggregation issue since it is not clear during design time which execution paths need to be taken into account. Thus JAEGER et al. proposed to utilize a simple probabilistic model assuming that the probability for each path to be taken is equal. Given the set $\mathbb{N}$ of all tasks in a process model, let $\mathbb{T}$ denote all sets of the power-set of $\mathbb{N}$ which may be triggered by an OR-split, i. e. each possible flow between an OR-split and the corresponding join. Determining upper and lower bounds for QoS dimensions in $\mathbb{T}$ requires taking all possible combinations of flows into account which is denoted by the power-set $\mathcal{P}(\mathbb{T})$. In order to simplify the notation of aggregation operations which are applied on $\mathcal{P}(\mathbb{T})$, JAEGER et al. defined that the following must hold [JRM04]:

$$f(\mathcal{P}(\mathbb{T})) := f(\{x : x = f(\mathbb{S}), \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}) \tag{3.6}$$

This allows to first aggregate the subsets $\mathbb{S}$ of $\mathcal{P}(\mathbb{T})$ with an aggregation function f which results in a new set, denoted $\mathbb{M}$ in the following. In a second step, the same aggregation is applied on $\mathbb{M}$ as well. This definition is useful for rather simple QoS dimensions requiring e. g. the maximum or the minimum of $\mathcal{P}(\mathbb{T})$. The usual aggregation in cases of OR-splits is however that the inner and the outer function are different, depending on the semantics of the considered QoS dimension. Following is a simple example for clarification. Let $\mathbb{T} = \{s_{1j}, s_{2j}\}$ be two potential flows between an OR-split and an OR-join. The corresponding power-set is then $\mathcal{P}(\mathbb{T}) = \{\{\emptyset\}, \{s_{1j}\}, \{s_{2j}\}, \{s_{1j}, s_{2j}\}\}$. It is assumed that the semantics of a hypothetical QoS dimension requires that QoS values need to be added. Then for the piece of process between the OR-split and the corresponding join, upper and lower bounds for the aggregated QoS vector $\mathcal{Q}$ can be determined as follows:

1. $\mathbb{M} = \{x : x = \sum\limits_{y \in \mathbb{S}} q^{\alpha}_{\iota(y)}, \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}$ where $\iota(y)$ denotes the index of y which in our example can be either $\emptyset$, $1j$ or $2j$.
2. Determine the upper bound for the hypothetical QoS dimension by $max(\mathbb{M})$ and the lower bound by $min(\mathbb{M})$.

In the remainder of this thesis all aggregations of OR-splits and joins will be assumed to apply this probabilistic model if not stated otherwise. As mentioned before, aggregating QoS dimensions requires assigning an aggregation scheme for each of these CPs depending on the semantics of each considered QoS dimension. In the following this will be done exemplarily for price, availability and response time since these were employed in the sample process in Section 3.2.1. Hence in the following the semantics of these QoS dimensions will be discussed and how they behave for different CPs:

- **Price:** Price is maybe the most simple QoS dimension and behaves additive. Each potential invocation of a service is bound to a cost, even repetitive invocations of the same service. In case of conditional branches such as XOR- and OR-splits, the third type of strategy as outlined at the beginning of this Section is applied. That is, when assigning services to tasks in conditional branches, only the branch with the highest price is considered for aggregation. This ensures that the price of all other branches is at least the same or lower.

- **Availability:** Availability represents the probability that a service executes successfully. Each service execution is assumed to be independent. Thus the corresponding probabilities are multiplied. As for price, in case of conditional branches only the branch with the lowest availability is considered for aggregation. Hence it is ensured that the services of all other conditional branches yield at least the same or a better (i. e. higher) availability.

- **Response Time:** As the name indicates, response time measures the time that a service requires to handle a request and send an answer. Response times of different service invocations are considered independent and are thus additive. In case of response time, parallelisms are treated like conditional branches. Only the branch with maximum response time is considered for aggregation. Again this ensures that the response times of the other, non-critical execution paths yield the same or a lower response time.

Table 3.4 shows the aggregation schemes for these QoS dimensions with q_{ij}^{α} denoting the QoS vector of service alternative s_{ij}. These were adapted from [JRM04] to fit with the notation employed in this thesis. In the shown aggregation schemes, $\mathcal{Q}^{\alpha}$ denotes component α of the global QoS vector $\mathcal{Q}$, n the number of tasks involved in the respective CP and k the number of (assumed) loop iterations [JRM04].

Given a set of aggregation schemes, $\mathcal{Q}$ can be determined by step-wise collapsing the process model into a single node. CARDOSO proposes to alternately aggregate sequences and parallelisms/conditional branches. The reduction rules are quite simple: Parallelisms or conditional branches are reduced first if the flows consist of exactly one task, otherwise sequences and loops are reduced first. Then iteratively the alternate group is reduced until the process model finally consists of only one task. The result of this reduction is an aggregation expression for each QoS dimension for a given process model [Car02; JRM04]. Applying this decomposition scheme to the sample process shown in Figure 3.2 leads to the following steps:

1. Aggregate the XOR-split
2. Aggregate the sequence

This yields the aggregation expressions as shown below with Φ_{α} denoting the aggregation function for QoS attribute α:

- **Price:** $\Phi_1(q_{ij}) = q_{1j}^1 + max(q_{2j}^1, q_{3j}^1)$
- **Availability:** $\Phi_2(q_{ij}) = q_{1j}^2 * min(q_{2j}^2, q_{3j}^2)$

Table 3.4: *Aggregation schemes for price, availability and response time (based on [JRM04])*

CP	Price	Availability	Response Time
Sequence	$\mathcal{Q}^{\alpha}= \sum_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= \prod_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= \sum_{i=1}^{n} q_{ij}^{\alpha}$
Loop	$\mathcal{Q}^{\alpha}= kq_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= q_{ij}^{\alpha k}$	$\mathcal{Q}^{\alpha}= kq_{ij}^{\alpha}$
XOR-XOR	$\mathcal{Q}^{\alpha}= max\{q_{1j}^{\alpha}, \ldots, q_{nj}^{\alpha}\}$	$\mathcal{Q}^{\alpha}= min\{q_{1j}^{\alpha}, \ldots, q_{nj}^{\alpha}\}$	$\mathcal{Q}^{\alpha}= max\{q_{1j}^{\alpha}, \ldots, q_{nj}^{\alpha}\}$
AND-AND	$\mathcal{Q}^{\alpha}= \sum_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= \prod_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= max\{q_{1j}^{\alpha}, \ldots, q_{nj}^{\alpha}\}$
AND-COMPLEX	$\mathcal{Q}^{\alpha}= \sum_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= \prod_{i=1}^{n} q_{ij}^{\alpha}$	$\mathcal{Q}^{\alpha}= max\{q_{1j}^{\alpha}, \ldots, q_{nj}^{\alpha}\}$
OR-OR	$\mathcal{Q}^{\alpha}= max\{z : z = \sum_{y \in \mathbb{S}} q_{\iota(y)}^{\alpha}, \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}$	$\mathcal{Q}^{\alpha}= max\{z : z = 1 - \prod_{y \in \mathbb{S}} (1 - q_{\iota(y)}^{\alpha}), \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}$	$\mathcal{Q}^{\alpha}= max(\mathcal{P}(\mathbb{T}))$
OR-COMPLEX	$\mathcal{Q}^{\alpha}= max\{z : z = \sum_{y \in \mathbb{S}} q_{\iota(y)}^{\alpha}, \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}$	$\mathcal{Q}^{\alpha}= max\{z : z = 1 - \prod_{y \in \mathbb{S}} (1 - q_{\iota(y)}^{\alpha}), \forall \mathbb{S} \in \mathcal{P}(\mathbb{T})\}$	$\mathcal{Q}^{\alpha}= max(\mathcal{P}(\mathbb{T}))$

- **Response Time:** $\Phi_3(q_{ij}) = q^3_{1j} + max(q^3_{2j}, q^3_{3j})$

Hence every dimension of $\mathcal{Q}$ correlates to an aggregation function or, in a general form:

$$\mathcal{Q}^\alpha = \Phi_\alpha(q_{ij}) \tag{3.7}$$

3.2.3 Normalizing Quality Attributes

In the previous section it has been explained how to aggregate QoS properties of a service selection. Obviously this is a necessary requirement to decide if a solution fulfills given QoS requirements or not. However, if multiple (Pareto optimal) solutions fulfill QoS requirements, the question arises which one is better? The sample in the previous chapter employed three QoS attributes, namely price, availability and response time. For availability the rule is "the higher the better" while for price and response time it is "the lower the better". Furthermore, each QoS attribute is quantified using a different measure. Hence Multiple Attribute Decision Making (MADM) tools such as Simple Additive Weighting (SAW) cannot be applied [YH95]. Generally, to make different QoS attributes comparable, some kind of normalization is required depending on the scale of measure [JD88]. Normalizing QoS attributes is particularly crucial for modeling service selection as an optimization problem (cf. Section 3.2.4).

Measurement theory differentiates between a total of four measurement scales. The scale defines, among other things, which mathematical operations are valid on measured values [Ste46; JD88]. In the following, each measurement scale will be briefly introduced with a short example from web services:

1. **Nominal:** A nominal scale is employed to classify elements into different categories with no order between these categories whatsoever. Examples for nominal scales include categorization of parliament members into political parties or of answers into categories "yes" and "no". In the context of web services, nominal scales are employed to measure QoS properties such as provider country or utilized encryption algorithm. Allowed mathematical operations are tests for equality and inequality [Ste46; JD88].

2. **Ordinal:** An ordinal scale is characterized by two properties. First, it allows for defining categories of elements like a nominal scale. Second, it allows to define an ordering relation between these categories. However, it does not make any assertions regarding the distance between two categories. For instance, the distance between the natural numbers 1 and 2 is the same as between 2 and 3, i. e. one. However, in terms of e. g. military ranks (as an

example for an ordinal scale), a general is greater than a major and a major is greater than a private. But it is not possible to make any statements regarding the distance between these ranks. More precisely, a mathematical operation such as $General - Private = Major$ does not make any sense. In the context of web services, typical candidates for QoS properties measured using an ordinal scale are key lengths for encryption algorithms and resolutions of video streams. Mathematical operations allowed are equality, inequality as well as comparison operations [Ste46; JD88].

3. **Interval:** Interval ratios are similar to ordinal scales. However, they additionally incorporate a unit of measurement which implicitly defines an equal distance between measured values. For instance, the difference between 2° and 3° Celsius is the same as between 3° and 4° Celsius, i. e. 1° Celsius. QoS properties such as response time (measured in ms) and backup intervals (measured in any unit of time) are typical examples from the web services domain. Valid mathematical operations are the operations for ordinal scales as well as addition and subtraction [Ste46; JD88].

4. **Ratio:** In addition to the properties of an Interval scale, Ratio scales have an absolute zero value. Temperature measured in Kelvin is an example from real-world. For web services, QoS properties measured utilizing Ratio scales are price (measured in any currency) as well as availability (measured in % of a year's time). In addition to the mathematical operations allowed for Interval scales, Ratio scales also allow for applying multiplication and division [Ste46; JD88].

Nominal and Ordinal scales are also referred to as *qualitative scales* while the Interval and Ratio scales are subsumed under the term *quantitative scales* [Ste46; JD88]. A remaining challenge is however the distinction of different ordering relations. In particular, for some QoS properties such as availability a higher value is better while for other such as price a lower value is better. To differentiate these two kinds of ordering relations, a simplification will be utilized in the remainder of this thesis: If the ordering relation of an Ordinal, Interval or Ratio scale is increasing (such as availability), it will be referred to as *ascending* and if the ordering relation is decreasing (such as price) as *descending*.

In the following, proposed approaches for normalizing quantitaive QoS attributes in the context of web services will be discussed. Respective methods have been proposed by [ZBD+03; Ber07; YZL07; AR09; LDS+10; YCY12]. It should be noted, that the approach described in [ZBD+03] has also been used in e. g. [ZBN+04; GJ05; AP06; AP07; MBK+09; BYL+11; SSS11; SLI12]. To the best of the author's knowledge, the method proposed in [AR09] has been reused only once in [ARN12]. All other approaches have not been employed in other publications so far. Thus, the approach presented in [ZBD+03] is the most widely used normalization method in service composition. The approaches presented in [LDS+10] and [YCY12] are slightly modified versions of [ZBD+03]. Table 3.5 provides an overview of these approaches.

The discussion will be started with the normalization method of Zeng et al. Following are the equations employed for normalization [ZBD+03]:

$$q'^{\alpha}_{ij} = \begin{cases} \frac{q^{\alpha}_{ij} - Q^{\alpha}_{min}}{Q^{\alpha}_{max} - Q^{\alpha}_{min}} & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} \neq 0 \\ 1 & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} = 0 \end{cases} \tag{3.8}$$

$$q'^{\alpha}_{ij} = \begin{cases} \frac{Q^{\alpha}_{max} - q^{\alpha}_{ij}}{Q^{\alpha}_{max} - Q^{\alpha}_{min}} & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} \neq 0 \\ 1 & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} = 0 \end{cases} \tag{3.9}$$

Here, q^{α}_{ij} denotes the actually measured quality value while Q^{α}_{max} and Q^{α}_{min} denote overall maximum and minimum values of the currently observed QoS attribute over all S_i. Equations (3.8) and (3.9) are used to normalize ascending and descending QoS attributes respectively. This method ensures that quality values are normalized into the interval $[0, 1]$. Independently of the reading direction, a higher normalized value means "better". Li et al. extended this model in order to cope with QoS values that might differ from design to execution time. The basic idea is to predict QoS values at execution time and use this predicted value in the process of service selection. In case that no prediction is possible for any reason, equations (3.8) and (3.9) are extended for a third possible outcome for q'^{α}_{ij} which is -1 [LDS+10]. Yan et al. use equations (3.8) and (3.9) in reverse order, i. e. equation (3.8) is utilized to normalize descending and equation (3.9) to normalize ascending QoS attributes. The reason is that service selection is modeled as a graph problem aiming at finding the shortest path through the graph employing Dijkstra's algorithm (cf. [Dij59]) and is thus a minimization problem [YCY12].

Berbner aims at providing a scoring for QoS values in the range $[0, 10]$ where 10 is interpreted as best. This scoring is utilized to order web services for each service class S_i according to user preferences. Consequently (and contrary to [ZBD+03]), $Q^{\alpha}_{i,max}$ and $Q^{\alpha}_{i,min}$ are determined for each service class S_i rather than globally for all possible service compositions [Ber07].

The approach of Yu et al. normalizes QoS values utilizing statistical measures rather than maximum and minimum values. In their formulation, μ^{α}_{i} and σ^{α}_{i} denote the average and the standard deviation respectively for q^{α} in service class S_i [YZL07].

Table 3.5: *Overview of normalization methods for quantitative QoS attributes*

Approach	Ascending	Descending
ZENG et al. (2003)	$q'^{\alpha}_{ij} = \begin{cases} \frac{q^{\alpha}_{ij} - Q^{\alpha}_{min}}{Q^{\alpha}_{max} - Q^{\alpha}_{min}} & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} \neq 0 \\ 1 & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} = 0 \end{cases}$	$q'^{\alpha}_{ij} = \begin{cases} \frac{Q^{\alpha}_{max} - q^{\alpha}_{ij}}{Q^{\alpha}_{max} - Q^{\alpha}_{min}} & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} \neq 0 \\ 1 & \text{if } Q^{\alpha}_{max} - Q^{\alpha}_{min} = 0 \end{cases}$
BERBNER (2007)	$q'^{\alpha}_{ij} = \frac{q^{\alpha}_{ij}}{Q^{\alpha}_{i,max}} * 10$	$q'^{\alpha}_{ij} = \frac{Q^{\alpha}_{i,min}}{q^{\alpha}_{ij}} * 10$
YU et al. (2007)	$q'^{\alpha}_{ij} = \frac{q^{\alpha}_{ij} - \mu^{\alpha}_{i}}{\sigma^{\alpha}_{i}}$	$q'^{\alpha}_{ij} = 1 - \frac{q^{\alpha}_{ij} - \mu^{\alpha}_{i}}{\sigma^{\alpha}_{i}}$
ALRIFAI and RISSE (2009)	$q'^{\alpha}_{ij} = \frac{Q^{\alpha}_{i,max} - q^{\alpha}_{ij}}{Q^{\alpha}_{i,max} - Q^{\alpha}_{i,min}}$	$q'^{\alpha}_{ij} = \frac{Q^{\alpha}_{i,max} - q^{\alpha}_{ij}}{Q^{\alpha}_{i,max} - Q^{\alpha}_{i,min}}$

ALRIFAI and RISSE employ a simplified version of the normalization method presented in [ZBD+03]. In particular, the case for normalizing QoS attributes where $Q^{\alpha}_{max} - Q^{\alpha}_{min} = 0$ has been skipped. What is however remarkable is that the normalization expression which is used in [ZBD+03] to normalize descending QoS attributes, is employed by the authors to normalize descending as well as ascending QoS attributes. These differences lead to two problems:

1. QoS attributes where only a single value is measured (i. e. $Q^{\alpha}_{min} = q^{\alpha}_{ij} = Q^{\alpha}_{max}$) cannot be represented as this would require a division by 0 due to the skipped case differentiation.

2. As the only normalization expression employed was originally proposed for descending values, normalized values for ascending values have an opposite reading direction. This especially hinders comparison of QoS attributes with different reading directions and thus contradicts the motivation for employing normalization.

Especially due to the second issue, this approach has to be considered erroneous.

Regardless of the chosen aggregation scheme, aggregation of a normalized QoS attribute q'^{α}_{ij} is in the following denoted with the aggregation function Φ'_{α}. Aggregation schemes are applied to normalized QoS values as presented in Section 3.2.2.

3.2.4 The Service Selection Problem

The service selection problem can be verbally expressed as follows: For a given process model, find the best service for each task from the corresponding service class such that the QoS constraints of the process model are respected. Figure 3.7 shows a fictitious result of a selection process for the sample process.

The task of finding the best selection of services for a given process model is obviously an optimization problem. In Section 3.2.1 a brute force approach was presented, i. e. all possible solutions were enumerated and the best alternative was picked. The example was in particular simple as it yielded only one valid solution. Real-world service selection problems are however facing several challenges:

- **Multiple feasible solutions:** It is unlikely that for a process model only one feasible solution exists. Even if there are only two, service selection must support the user in the process of deciding which solution is “better”.

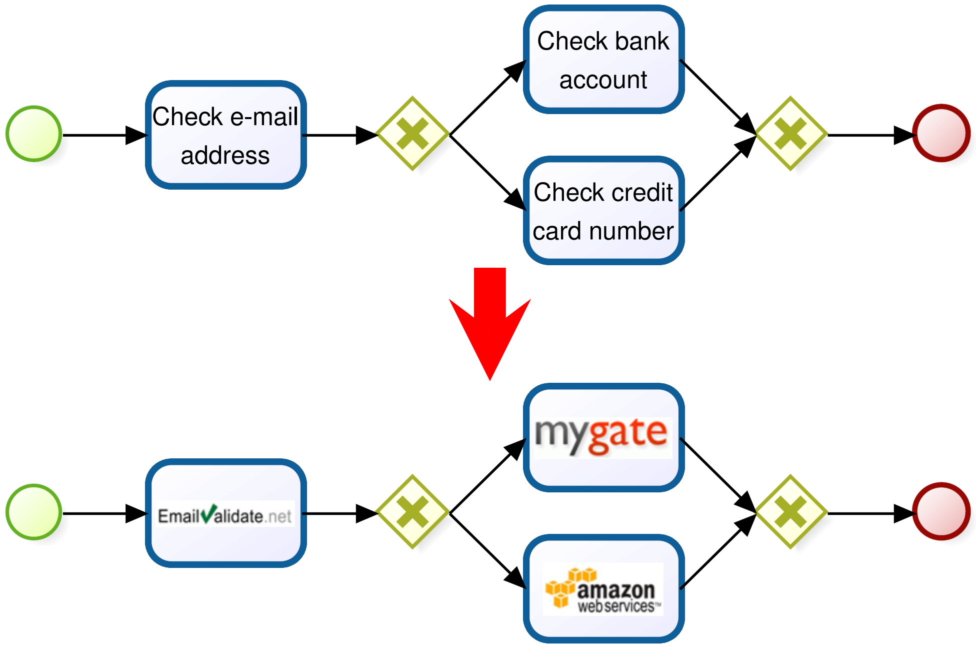


Figure 3.7: *Sample process with an exemplary service selection*

- **Big search space:** Real-world processes contain a lot more tasks than the presented sample process and there are numerous available service alternatives. As such exhaustive search space exploration in terms of full enumeration is infeasible for real-world process models.
- **Strict time constraints:** In some cases, service selection is subject to strict time constraints. Particularly, this includes cases where a maximum service outage time has been codified in form of an SLA as well as autonomous systems which are required to recover from any service outage in a certain time span.

Facing these challenges, more effective approaches than simple brute force have been proposed to tackle the service selection problem. Two major groups of approaches are *combinatorial* and *graph-based* approaches. Combinatorial approaches (e. g. [YL04; CDE+05; AP06; YZL07; CDE+08; KIH11]) utilize mathematical problem formulations and solving approaches known from standard literature on operations research (cf. Section 2.5). Depending on the chosen optimization model, different solving approaches can be employed. A pivotal advantage of combinatorial approaches is that the structure of a process model in terms of composition patterns generally does not require special consideration in either the optimization model or the solving approach. The control flow in terms of composition patterns is addressed separately by utilizing aggregation functions. Aggregated QoS vectors then form the basis for the actual selection and optimization process. On the contrary, suiting optimization models for service selection usually cannot be solved exactly at all or in acceptable time since the problems considered are usually NP-hard. Hence, combinatorial service selection often generates only a sufficiently well solution rather than an optimal and usually does not give any guarantees on algorithm runtime.

Graph-based approaches (e. g. [Men02; ZBD+03; YL04; CCG+06; YCY12]) utilize a Directed Acyclic Graph (DAG) as system model. This leads to a shortest path problem which can be solved exactly in P and is well-suited for process models which are mainly sequential. Hence, the control flow in terms of composition patterns becomes part of the problem formulation. The major drawback of graph-based approaches is however that any non-sequential process structure needs special consideration in the problem formulation or the solving approach. Loops for instance need to be unrolled to multiple nodes in the DAG which might lead to a tremendous growth of the model. Forks and joins require special consideration in the problem formulation, the solving approach or both.

Table 3.6 summarizes the advantages and disadvantages of these groups of approaches. The assumption that the underlying models of business processes as considered in the scope of this thesis are mainly sequential, is in general untenable. This is already illustrated by the sample process utilized in this chapter. Hence, this thesis focuses on combinatorial approaches for service selection.

Table 3.6: *Advantages and disadvantages of composition approaches*

	Combinatorial	**Graph-based**
Advantages	• All CPs can be considered • CPs do not require special consideration in modeling or solving approach	• Sequential flow structures can be formalized as DAG and solved exactly in P
Disadvantages	• Problem formulations usually NP-hard, hence exact solutions usually unfeasible • Usually no runtime and optimality guarantees	• Non-sequential flow structures need special consideration in the problem formulation, the solving approach or both

3.2.4.1 System Model

The combinatorial system model is an integer array with a length equal to the number of tasks in a process model. This system model forms the basis for the solution vector of an optimization problem. Depending on the problem formulation, the system model can be used either as it is as a solution vector or might require a certain transformation. Each array entry corresponds to a service class S_i and indicates which service alternative s_{ij} to select. Thus, the value of each entry is limited by $|S_i|$. Figure 3.8 shows the combinatorial model for the sample process and the service alternatives presented in Table 3.2. As stated above, the control flow is considered separately by means of aggregation functions and is thus not reflected at all by the system model.

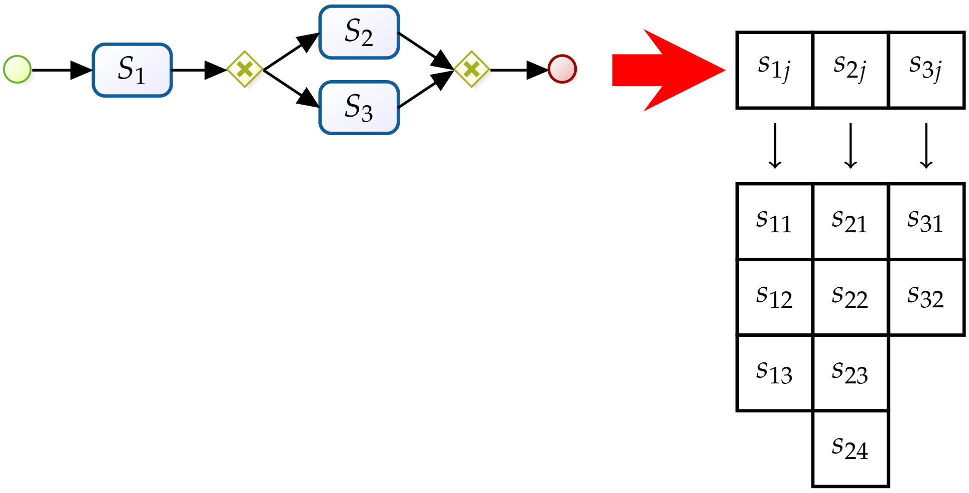


Figure 3.8: *Combinatorial model for the sample process*

3.2.4.2 Problem Formalization

The service selection problem is often formalized as KP. The classic 0-1 KP introduced in Section 2.5.3 assumes that all objects belong to one class and that possibly multiple items are selected based on a single abstract utility value with respect to a capacity restriction. However, the service selection problem has two properties which are not captured by this basic model:

1. Different classes of objects (i. e. service classes) and from each class exactly one item has to be selected.

2. Different dimensions of utility (i. e. QoS attributes), some of which yield better utility with increasing value, some with decreasing and again others by means of a partial ordering.

Due to these properties the MMKP[51] has become popular to model the service selection problem since first employed by YU et al. [YZL07]. Similar to the classical 0-1 KP, the MMKP aims at finding a subset of items with maximum utility while

[51] Alternate names found in the literature are e. g. "Multiple-Choice Multi-Dimension Knapsack Problem" [Kha98] and "Multidimension Multichoice Knapsack Problem" [YZL07].

respecting m constraints on capacity (3.11). In contrast to the classical 0-1 KP, items are divided into k distinct groups or classes G_i with each item belonging to exactly one group. From each group G_i, exactly one item is to be chosen (3.12). The basic form of the MMKP is as follows [KPP04]:

$$\max z = \sum_{i=1}^{k} \sum_{j \in G_i} u_{ij} x_{ij} \tag{3.10}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j \in G_i} w_{ij}^{\alpha} x_{ij} \leqslant c^{\alpha} \qquad (\alpha = 1, \ldots, m) \tag{3.11}$$

$$\sum_{j \in G_i} x_{ij} = 1 \qquad (i = 1, \ldots, k) \tag{3.12}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, k; j \in G_i)$$

Generally, weights w_{ij} can be properties of any kind. In the context of service selection, the weights represent QoS properties of a single service alternative. The basic formulation of the MMKP assumes that weights are summarized. As has been seen in Section 3.2.2, the aggregation scheme is dictated by the control flow of a process model as well as the semantics of a QoS property. Additionally, the reading direction of each QoS property might be different. Therefore, in the context of service selection, the capacity constraint (3.11) is reformulated to refer to the normalized aggregation function Φ' and normalized quality vector q'_{ij}. Normalized global capacity constraints on QoS dimension α are denoted by C'^{α}. Furthermore, the utility value u_{ij} is replaced with a utility function F_{ij}. This function calculates a utility value based on all (normalized) QoS properties of selected service alternatives s_{ij} with respect to user preferences (cf. Section 3.2.4.3 for a thorough discussion). Thus, the MMKP as adapted for service selection has the following form:

$$\max z = \sum_{i=1}^{k} \sum_{j \in S_i} F_{ij}(s_{ij} x_{ij}) \tag{3.13}$$

$$\text{s.t. } \Phi'_{\alpha}(q_{ij}^{\prime\alpha} x_{ij}) \leqslant C'^{\alpha} \qquad (\alpha = 1, \ldots, m) \tag{3.14}$$

$$\sum_{j \in S_i} x_{ij} = 1 \qquad (i = 1, \ldots, k) \tag{3.15}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, k; j \in S_i)$$

In a strict sense, MMKP is a hybrid model. By settings $\alpha = 1$ one obtains the Multiple-Choice Knapsack Problem (MCKP). Setting $k = 1$ and omitting the choice constraint (3.12) or (3.15) respectively yields the Multidimensional or d-dimensional Knapsack Problem (d-KP). Both problems are known to be NP-hard [KPP04]. This makes MMKP one of the more challenging KP variants.

MMKP has been employed in several real-world applications such has resource management in multimedia systems (e. g. [Kha98; AMS+01]). Due to its complexity however only few algorithms exist for solving problem instances exactly [Kha98; Sbi07; RG09; GR11]. All of these approaches make use of the branch and bound method [LD60] and are designed only for a single objective function. To the best of the author's knowledge, there is currently no exact approach for tackling MMKP by means of a MOO problem.

Besides these exact algorithms, numerous heuristics have been proposed utilizing either general concepts such as convex hulls or problem-specific knowledge (e. g. [Kha98; AMS+01; ARK+06; IBR10; CHM+12]). Reported results indicate that these heuristics deliver good near-optimal solutions in a comparatively short time. For instance, KHAN reports that his heuristic is able to achieve objective function values within 4% of the optimum while being 17 to 28152 times faster than his proposed optimal algorithm for a single objective function [Kha98].

In the context of service selection, MMKP has been utilized numerous times for problem formulation (e. g. [LJL+03; CDE+05; CAH05; AP06; YZL07; CDE+08; WCS+08]). This underlines that MMKP is well suited for representing the service selection problem.

3.2.4.3 Utility Function

The utility function is used to obtain an utility value for a service alternative based on its actual (normalized) QoS properties. The obvious advantage of this approach is that service alternatives can be ranked according to this utility value. Utility functions furthermore enable users to express preferences regarding QoS dimensions. Implicitly this also allows for defining the MMKP for service selection as a SOO problem. More precisely, the objective in this case is to optimize utility rather than multiple QoS properties. However, employing a utility function leads to a restriction of the search space similar to the weighted sum method presented in Section 2.5.4. The reason is that in cases where no user preferences are given, utility methods assume uniform importance of QoS attributes which leads to a single (rarely few) optimal solution(s). But as has been discussed in Section 2.5.4, this usually excludes compromise (i. e. Pareto optimal) solutions which would have been obtained if the problem had been formulated as a MOO problem.

The most often employed utility function in the field of service selection is

SAW which is a classic technique from the field of MADM. Furthermore, several proprietary utility functions have been proposed for either normalized or non-normalized QoS attributes. Following is a brief presentation of these utility functions.

The general SAW method for service selection is defined as follows [JD88]:

$$F_{ij} = \sum_{\alpha=1}^{m} w_\alpha q_{ij}^{'\alpha} \tag{3.16}$$

Here, m denotes the number of QoS attributes and w_α the weight of normalized QoS attribute α. The SAW method assumes that preferences are mutually independent. Although not explicitly part of the SAW method, applications often require that $\sum_{\alpha=1}^{m} w_\alpha = 1$ holds[52]. The reason is that the SAW method presumes that weights are proportional to changes of QoS values. For instance, if a utility function is defined over two QoS attributes with weightings $w_1 = \frac{1}{3}$ and $w_2 = \frac{2}{3}$, changes of two units of $q_{ij}^{'1}$ and one unit of $q_{ij}^{'2}$ must yield the same utility value [JD88]. To the author's best knowledge, the unrestricted SAW method has only been used in [AP07; WLH07; AM10] while the restricted version has been employed in [ZBD+03; ZBN+04; GJ05; CCG+06; AP06; CCG+07; AR09; KIH11; ARN12; Sim12; YCY12]. Vu et al. use a slightly modified unrestricted SAW method where the utility value is divided by the sum of weights in order to receive a proportional utility score as in the restricted SAW method [VHA05]. Another variant of the unrestricted SAW method is utilized by Berbner et al. QoS attributes are differentiated in three distinct groups or types, i. e. additive, multiplicative and min-aggregated. The classification is based on the aggregation function which is applied on the respective QoS attribute in sequential workflows. The attributes of each type are handled by the unrestricted SAW method. The final utility value is the sum of these three values [BSR+06; BSR+07].

The utility function proposed by Yu et al. is applied on non-normalized QoS attributes. It performs normalization utilizing statistical measures (cf. Section 3.2.3) and is defined as follows [YL04; YZL07]:

$$F_{ij} = \sum_{\alpha=1}^{x} w_\alpha \left(\frac{q_{ij}^{\alpha} - \mu^{\alpha}}{\delta^{\alpha}} \right) + \sum_{\beta=1}^{y} w_\beta \left(1 - \frac{q_{ij}^{\beta} - \mu^{\beta}}{\delta^{\beta}} \right) \tag{3.17}$$

where $\delta^\alpha, \delta^\beta \neq 0$. This function assumes that x QoS attributes are to be maximized

[52] In the remainder of this thesis, this version will be referred to as the *restricted SAW method* whereas the classic formulation will be called the *unrestricted SAW method*.

and y QoS attributes to be minimized. w_α and w_β are weights for which $0 < w_\alpha, w_\beta < 1$ and $\sum_{\alpha=1}^{x} w_\alpha + \sum_{\beta=1}^{y} w_\beta = 1$ must hold. As already mentioned in Section 3.2.3, μ and δ are the average and the standard deviation respectively of the QoS values for all service candidates in a service class [YL04; YZL07]. Recently, HEINRICH and LEWERENZ showed that limiting μ and δ on a single service class may easily lead to confusing results when modeling service selection as a graph problem. In particular, compositions incorporating different paths and service classes may be systematically under- or overrated. In other words, a "bad" composition may be considered as optimal [HL13].

Finally, MABROUK et al. utilize a rather simple utility function which is defined globally as the average of normalized and aggregated QoS values [MBK+09]:

$$F = \frac{\sum_{\alpha=1}^{m} Q'^{\alpha}}{m} \tag{3.18}$$

where $m \neq 0$. As already stated, employing an utility value instead of multiple QoS values reduces mathematical complexity because it allows for defining the MMKP as SOO problem. An important question in this context is how the weights are obtained? Generally, these are considered to be determined by the user himself. For rather simple QoS models consisting of three attributes (e. g. [CCG+06]) this might be an almost trivial task. But how about rather complex QoS models such as the proposal of the W3C which consists of 13 attributes, some of which are even subdivided into more fine-grained measures [LJL+03]? It turns out that while employing a utility function reduces mathematical complexity, its usage might lead to non-trivial semantic problems and restricts the search space. Furthermore, some combinations of problem formulation and certain utility functions may lead to confusing results [HL13]. Hence the formulation of the service selection problem as MOO problem is raising in popularity.

3.2.4.4 Multi-Objective Service Selection

As has been stated in Section 2.5.4, a MOO problem is characterized by at least two conflicting objective functions which have to be optimized simultaneously. Applied to the service selection problem, each QoS attribute forms an objective on its own. Taking up the aggregation functions introduced in Section 3.2.2, the service selection problem for normalized QoS attributes can be formulated as a MOO problem as follows:

$$\max \{\Phi'_1(q'^{1}_{ij} x_{ij}), \Phi'_2(q'^{2}_{ij} x_{ij}), \Phi'_3(q'^{3}_{ij} x_{ij})\} \qquad (i = 1, \ldots, k; j \in S_i) \tag{3.19}$$

$$\text{s.t. } \Phi'_\alpha(q'^{\alpha}_{ij} x_{ij}) \leqslant C'^{\alpha} \qquad (\alpha = 1, \ldots, m) \tag{3.20}$$

$$\sum_{j \in S_i} x_{ij} = 1 \qquad (i = 1, \ldots, k) \tag{3.21}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, k; j \in S_i)$$

As can be seen, the objective in this formulation is to maximize each QoS dimension separately. Achievement of this goal is measured employing multiple aggregation functions. The major advantage of this approach is that it does not require to define a utility function in order to optimize. Instead the aggregation functions can be used unchanged as objective functions. However, as outlined in Section 2.5.4, the major drawback is the complexity of considering multiple objective functions simultaneously. As has been recently discussed in Section 3.2.4.2, there are only few exact approaches for solving the MMKP but these require a single objective function.

For tackling service selection as a MOO problem, GAs (cf. Section 2.5.6) are being frequently employed [CDE+05; CAH05; JM07; CDE+08; WCS+08; KIH12; WKI+12]. The respective approaches will be reviewed in Chapter 4. Here, the principles of aligning service selection with GAs will be discussed.

Encoding the solution vector as chromosome is performed based on the system model presented in Section 3.2.4.1. This means, that the chromosome is an integer-array with a length equal to the number of tasks in a process model. The i-th array element hence corresponds to the i-th task and the respective array entry reflects the chosen service alternative s_{ij}. An important property of this chromosome representation is that it implicitly respects the one-item-per-class restriction of the MMKP (3.12). Since the MMKP utilizes binary decision variables, the chromosome needs to be transformed accordingly when evaluating a solution obtained by a GA. Let *chr* denote an integer array representing the chromosome. A binary solution vector can then be constructed with the following mapping:

$$x_{ij} = \begin{cases} 1 & \text{if } j = chr[i] \\ 0 & \text{else} \end{cases} \tag{3.22}$$

Figure 3.9 shows an example of transforming a chromosome into a binary solution for the sample process and the service alternatives presented in Table 3.2.

Regarding the fitness function, there is no single definition used in the scope of this thesis. The reason is that for solving the optimization problems considered in this work, different GAs have been employed which were proposed in the literature. Each of these GAs however defines its own fitness function. The employed GAs and the corresponding fitness functions are being presented in Chapter 7.

$$
\begin{aligned}
chr[1] &: \boxed{2} \rightarrow x_{11} = 0, x_{12} = 1, x_{13} = 0 \\
chr[2] &: \boxed{3} \rightarrow x_{21} = 0, x_{22} = 0, x_{23} = 1, x_{24} = 0 \\
chr[3] &: \boxed{1} \rightarrow x_{31} = 1, x_{32} = 0
\end{aligned}
$$

Figure 3.9: *Transforming a chromosome into a binary solution vector*

3.3 Security in Service Composition

Security in the context of service composition is a vast field of research. It has been considered from different perspectives such as QoS models and formal process calculi. Respective works can be roughly classified into QoS-based, policy-driven and model-driven approaches (cf. Figure 3.10). Some approaches however conceptually belong to multiple classes. In particular, policy- and model-driven methodologies are combined in the literature. The following subsections discuss the different classes with respect to the consideration of interdependence effects.

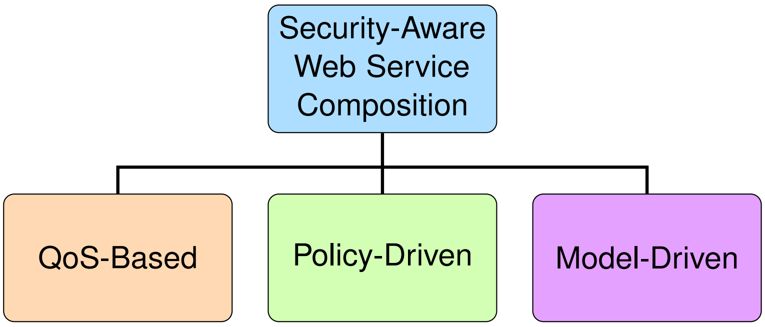


Figure 3.10: *Classification of approaches for security-aware service composition*

QoS-Based Approaches

QoS-based approaches consider security properties to be non-functional requirements which are codified in QoS models and verified and/or negotiated based on an SLA. QoS models for web services have been addressed by numerous publications (e. g. [SCD+97; OEH02; LJL+03; JRM04; CDE+05; CP09]). Regarding security, two groups can be identified. The first group treats security as a single QoS attribute [OEH02; LJL+03] whereas the second group partly decomposes security into multiple security attributes [SCD+97; JRM04; CDE+05; CP09]. While the approaches of the second group usually consider security objectives to be independent, SABATA et al. take into account interdependence between availability on the one hand and fault tolerance as well as resource management on the other hand. The consequence for their proposed QoS taxonomy is that availability is not treated as security objective but as a criterion for service level [SCD+97]. In general however, interdependence between security objectives is not covered by QoS-based approaches.

Policy-Driven Approaches

The common denominator of these approaches is that they state what is allowed and what is not based on (formally) defined policies. These policies are subsequently used to verify if service compositions comply with the specified policies. Some approaches also go a step further and propose policy enforcing mechanisms.

NARAYANAN and MCILRAITH employ Petri nets to formalize service compositions. Policies are expressed by means of flow conditions which are verified for reachability, liveness and the existence of deadlocks [NM02]. BARTOLETTI et al. propose utilizing the λ-calculus to formalize security policies. Security-critic code is wrapped into safety framings which enforce the given security policy at the time of executing the wrapped code. Service orchestration is performed with respect to the security policies of the orchestrator and the clients (i. e. services) [BDF06]. Similar formal approaches have been proposed in [RM09; MDC11]. ROSSI and MACEDONIO define a typed process algebra for specifying security requirements in service compositions. The work explores under which conditions single services may be replaced without deteriorating the security of the composition [RM09]. MARTÍNEZ et al. use a formal language to specify contracts between partners in a composition in terms of obligations, permissions and prohibitions. Contract compliance is verified using timed automata [MDC11].

PACI et al. focused on access control policies in service choreographies, especially on the question if a choreography can be implemented based on the access control policies of each service and the credentials each service is willing to disclose. Choreographies are modeled as transition system and credential disclosure policies as directed graphs [POM08]. The work of SUN et al. focuses on verifying security requirements of service compositions using finite state automata. From a user perspective, security requirements are expressed using terminology such as "low encryption" which is internally mapped to concrete security measures such as encryption algorithms and key sizes [SBH+10].

Conceptually related is the work of LAVARACK and COETZEE which addresses trade-offs in policy intersections. Originally introduced in the scope of the standard WS-Policy [W3C06], policy intersection aimed at identifying compatible policy alternatives (i. e. collections of policy assertions) in case of two or more policies[53]. The result was thus again a policy called intersection which consisted of assertions that are compatible with all policies. However, the original model did not allow for any other reasoning except of policy compatibility. In this context the authors propose to determine the pros and cons of intersections with methods from fuzzy logic. For this purpose the authors identified a total of six aspects such as trust and authentication which have an influence on the security level of a system. With these aspects, for each intersection a Fuzzy Cognitive Map is generated which describes the mutual relations and influences of these aspects expressed by weights. These maps are then evaluated with fuzzy interference which enables to order intersections according to the security level they offer [LC11].

Model-Driven Approaches

Approaches of this class employ meta-information such as semantic enrichment of process models in the course of selecting and/or configuring web services with respect to security requirements.

A lot of research has been conducted in order to align security features such as access control with the web service paradigm, either standards-based or on a conceptual basis [BBG05; SIM+07; WMS+09]. BHATTI et al. address trust and context issues in service selection and propose an extended XML-based Role-Based Access Control (RBAC) framework incorporating these aspects. The authors also describe an implementation architecture for the proposal [BBG05]. SRIVATSA et al. propose an approach for specifying and enforcing access control policies on service compositions. Policies are specified via a declarative language using pure-past linear temporal logic. Access control policies are being enforced using a dedicated service that verifies actions with deployed policies [SIM+07]. WOLTER et al. propose a model-driven approach which closely follows the MDA methodology (cf. [OMG03]). The foundation of the framework is a collaboratively designed business process model using a graphical notation such as BPMN which forms the Computational Independent Model (CIM). The subsequent step is the definition of security features by a security expert. The result is a security annotated business process model which forms the Platform Independent Model (PIM). Finally, the Platform Specific Model (PSM) is obtained by transforming the PIM with XSLT into target models. In particular, these are a BPEL process as well as WS-Security and XACML files. The latter two are intended to be deployed in a policy decision

[53] In the scope of service composition, the respective policies are defined by the service consumer and the service providers.

and enforcement component in order to enforce specified security constraints at runtime [WMS+09].

Another line of research aims at providing frameworks from modeling security constraints over security-aware service selection to composition [NTI+05; CFH06; JF09]. NAKAMURA et al. consider static service compositions which are modeled using UML. Security requirements are being attached to UML models by means of annotations. The annotated UML diagrams are subsequently being transformed to service-oriented applications according to the MDA approach. Both the application and the security requirements are intended to be performed by the same person, i. e. the application designer [NTI+05]. A similar but conceptually closer work to service composition was proposed by JENSEN and FEJA who propose custom security modeling elements for e. g. end-to-end encryption and access control. The modeling notation of choice are EPCs. Enriched process models are being transformed to BPEL afterwards [JF09]. CARMINATI et al. propose a framework for secure service composition which is build around what the authors call a secure WS-broker. Common security vocabulary is established between all parties using the OWL ontology. Security capabilities of web services are expressed using SAML assertions. Assigning web services to tasks is performed using a component called security matchmaker. This component builds a graph-structure representing all possible compositions with regards to compatibility of security capabilities of web services on the one hand as well as security requirements of the process model on the other hand. From among the composition alternatives, the best with regards to QoS criteria is selected and transformed into BPEL [CFH06].

Summary

Security-aware service composition has been studied from very different perspectives. Despite the great deal of conducted research however, interdependent security has gained only little attention. Except of the work of SABATA et al. [SCD+97], the author is not aware of any other approach considering interdependence effects in this context.

3.4 Compliance in Service Composition

In Section 2.4 it was identified, that out of the five compliance aspects differentiated in the literature, the aspects **Data**, **Location** and **Time Limits** need to be considered in the context of service selection. Hence in this section it will be at first analyzed to what extent these compliance aspects have an influence on service composition in general and service selection in particular.

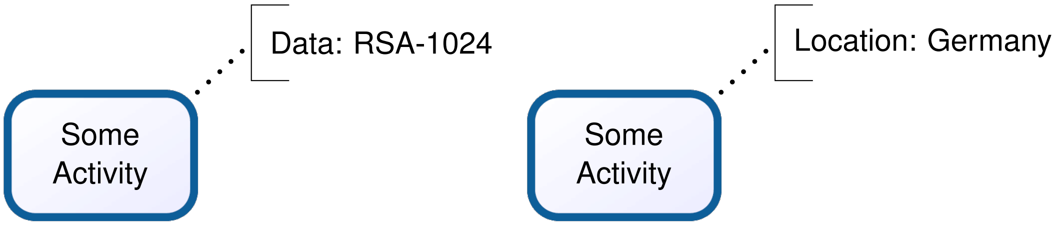
- **Data:** As stated in Section 2.4, this aspect covers the life cycle of data objects as well as data management. Since the life cycle of data objects (i. e. all activities from creation to deletion in a process model) is a modeling issue, it does not need to be considered in service selection. Depending on the type of processed data however, services might be required to fulfill a minimum of security measures such as a certain encryption strength. This leads to the necessity to consider these requirements as local constraints during service selection. In the following, respective constraints will be considered as required minimum encryption strength in terms of key length for a certain algorithm. Comparisons of encryption strengths between different algorithms will not be covered for the sake of simplicity.

- **Location:** Also depending on the type of data processed, tasks might be restricted to certain countries and/or regions. This usually applies to single tasks and thus needs to be considered as local constraint in service selection. Although some regulations require actions to take place in a region such as the EU, only exact locations in terms of countries will be considered in this thesis for the sake of simplicity. Since the decision if a location constraint is fulfilled or not is binary for either regions or single countries, this simplification does not lead to any loss of generality.

- **Time Limits:** It has already been discussed that single activities or subprocesses might be subject to legal time constraints. However, processes might be composed of non-human as well as human tasks. For human tasks, process designers usually allot a certain amount of time for completion. Thus, service selection for sub-processes which are subject to a time limit not only needs to consider this time limit in terms of an upper bound but also allotted times for human tasks.

From a modeling perspective, the compliance aspects which are relevant for service selection can be represented as follows. **Data** and **Location** aspects can be captured in BPMN by using text annotations on single activities. A text annotation (represented as an open rectangle) is linked to an activity using an association (depicted as a dotted line) [OMG11]. Figure 3.11 illustrates this.

For modeling **Time Limits**, BPMN offers a Timer Intermediate Event (represented as a clock marker) which may be attached to the boundary of an activity. In case that this activity is not finished within a certain period of time, process execution leaves the normal flow and is redirected to an exception flow. This subsequently leads to triggering an alternate activity to handle the exception. To



(a) Data (b) Location

Figure 3.11: *Modeling Data and Location constraints in BPMN*

indicate that an activity is performed by a human, BPMN offers a user task which is an activity containing a human figure in the upper left corner [OMG11]. Figure 3.12 shows a corresponding sample process. The activities "Non-human Activity", "Human Task" and "Exception Handling" are enclosed in a sub process which is also an activity itself in BPMN. Hence this allows to define time limits at different conceptual levels as stated above.

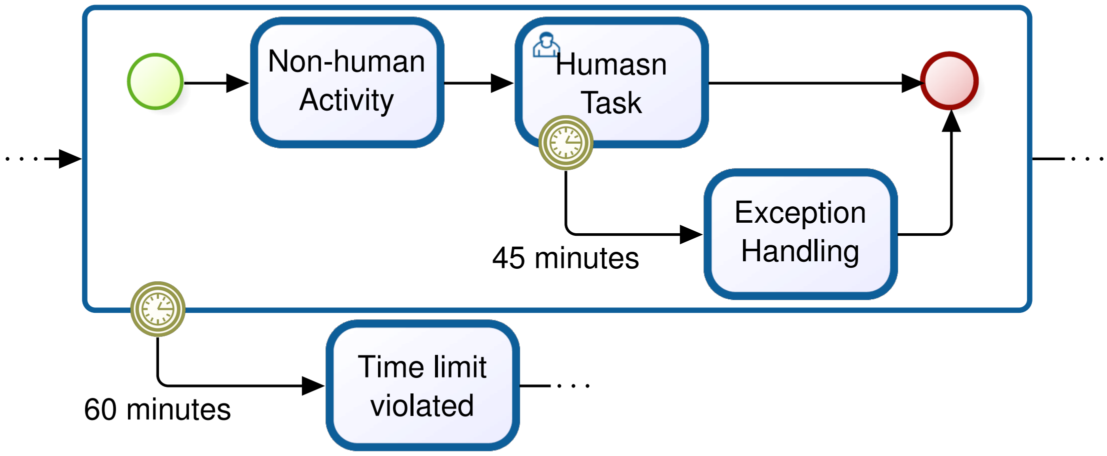


Figure 3.12: *Modeling Time Limit constraints in BPMN*

Transforming compliance requirements into optimization constraints is pretty straight forward. Let q^{enc}, q^{loc} and q^{rt} denote the QoS dimensions for encryption, location and response time respectively. Furthermore, let c_{enc} and c_{loc} denote local constraints regarding encryption and location. Then a local constraint on task t (which corresponds with service class S_t) can be expressed as follows:

$$\begin{aligned} q_{tj}^{enc} &\geqslant c_{enc} && (j = 1, \dots, |S_t|) \\ q_{tj}^{loc} &= c_{loc} && (j = 1, \dots, |S_t|) \end{aligned}$$

Modeling scope-based constraints requires at first hand to capture the tasks it consists of. Let T denote the set of task identifiers of the considered scope. A scope based constraint on time limit (denoted c_{rt}) can then be expressed as follows:

$$\sum_{t \in T} q_{tj}^{rt} \leqslant c_{rt} \qquad\qquad (j = 1, \ldots, |S_t|)$$

3.5 Alternative Composition Approaches

Besides of service selection, two other groups of approaches for service composition exist, namely *service planning* and *service mashups*. In the following these composition approaches will be briefly presented.

Service Planning

Approaches for service planning (e. g. [SPW+04; KG06; Zho07; LKS08; ZVB13]) do not require a process model but start with a description of the goals of a service composition in terms of tasks, similar to the introductory example in Section 2.1.1. Often this description is specified formally, e. g. in some calculus. This description forms the input for an Artificial Intelligence (AI) planner such as SHOP2 [SPW+04] or XPlan [KG06] which determines from available web services a composition fulfilling these required tasks. Although current mechanisms differ in terms of employed technologies and complexity, the basic mechanism is as follows[54]. For a given list of tasks, web services are step-wise assigned such that the capabilities of the service cover the current task. Planning is finished as soon as all tasks are iterated. Web service capabilities are mapped to tasks employing semantic facilities such as OWL-S ontologies and the DAML-S language. Depending on the capabilities of single web services, resulting plans can be fundamentally different. Again the sample process is utilized to derive the following goals (corresponding tasks are denoted t_i):

1. Check e-mail address (t_1)
2. Check either bank account number (t_2) or credit card number (t_3), depending on payment method

In correspondence to the introductory sample process, it is specified that t_1 has to be performed before t_2 and t_3. Furthermore, let the service repository contain a total of four services (cf. Table 3.7). Except of s_4, each service provides exactly one capability matching a task. That is, s_1 provides e-mail address checking, s_2 bank

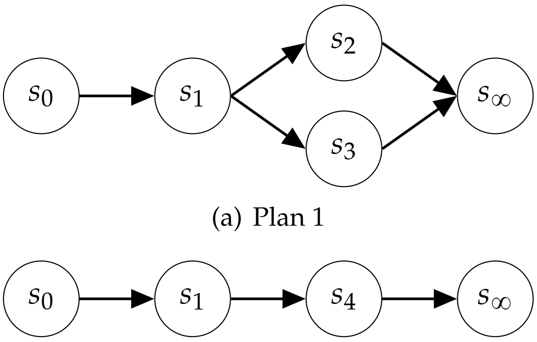
[54] A thorough presentation can be found in e. g. [Zho07].

account number checking and s_3 credit card number checking. Service s_4 provides checking capabilities for both, bank accounts and credit card numbers.

Table 3.7: *Service repository for service planning example*

Service	Capabilities
s_1	t_1
s_2	t_2
s_3	t_3
s_4	t_2, t_3

Given these goals and services, two different plans are possible. The first possibility is to invoke s_1 and afterwards either s_2 or s_3 depending on the concrete payment method. The second possible composition is to invoke s_1 and afterwards s_4. Figure 3.13 illustrates these possibilities as DAG where s_0 denotes the start and s_∞ the end node respectively. Either plan can be subsequently transformed into a BPEL process.



(a) Plan 1

(b) Plan 2

Figure 3.13: *Possible plans for composite service*

Service planning mechanisms have the advantages that no process model is required and that multiple service capabilities can be taken into consideration during service composition. BPEL processes can be generated (semi-)automatically from a high-level specification which makes service planning very flexible.

The major drawback of current approaches is however that service planning mechanisms are not able to handle complex process structures such as choices. Furthermore, planning is computationally very expensive and usually does not lead to optimal solutions from a QoS perspective [LAP06]. Finally, compliance aspects such as the sequence of tasks need to be considered explicitly when defining the goals. The definition of complex processes can thus easily become unmanageable. Some approaches such as XPlan [KG06] and QBroker [YZL07] therefore use service planning to generate an initial plan which is further optimized utilizing service selection.

Service Mashups

Similar to service planning and selection, service mashups are a development methodology for web applications. In contrast however, mashups have a much stronger focus on content publishing and visualization. Generally speaking, service mashups aim at combining web services with Web 2.0 technologies such as tags and microformats. From a conceptual perspective, the focus is not limited to implementing business processes. Hence, the utilization of service mashups in business settings is just one of several application areas [BDS08].

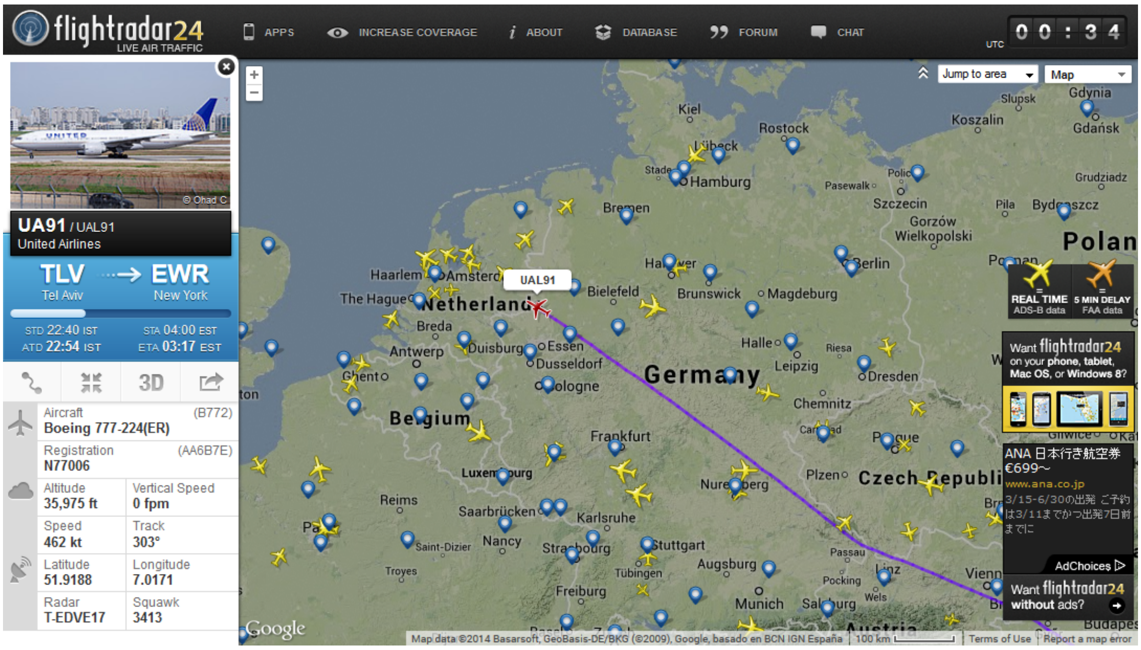
Originally, there were a lot of development tools for service mashups developed and maintained by companies such as IBM, MICROSOFT, GOOGLE and YAHOO. As of 2014 however, the only major tool remaining to create service mashups is YAHOO PIPES[55]. The other major vendors have discontinued their respective products since 2009. Service mashups are however still being actively developed. ProgrammableWeb lists a total of 7379 mashups as of March 8th 2014. An example for a service mashup is the project Flightradar24[56] which is a service for tracking flights. Based upon GoogleMaps, real-time information is provided regarding thousands of flights worldwide. The necessary data is gathered from different sources which are aggregated and used to populate the map with information regarding flights (including type information about planes in use) as well as airports. Figure 3.14 gives an impression.

3.6 Summary

This chapter introduced and discussed service composition from diverse perspectives such as composition topologies and modeling approaches. It lay the foundation for the approaches proposed in this thesis by discussing possibilities on how to consider security and compliance in service composition. Furthermore, essential assumptions have been introduced on which our contributions are built on. Finally, the mathematical model has been introduced and discussed which is used by the proposed approaches for service selection.

[55] `http://pipes.yahoo.com/pipes/`

[56] `www.flightradar24.com`



UA91 / UAL91
United Airlines

TLV → EWR
Tel Aviv New York

STD 22:40 IST STA 04:00 EST
ATD 22:54 IST ETA 03:17 EST

Aircraft (B772) Boeing 777-224(ER)	
Registration (AA6B7E) N77006	
Altitude 35,975 ft	Vertical Speed 0 fpm
Speed 462 kt	Track 303°
Latitude 51.9188	Longitude 7.0171
Radar T-EDVE17	Squawk 3413

Figure 3.14: *Screenshot of Flightradar24*

Part II

Design & Implementation

Chapter 4

Requirements Analysis and State of the Art

"A goal without a plan is just a dream."

– ANONYMOUS PROVERB

AFTER HAVING DISCUSSED fundamental topics in the previous chapters, it is now possible to define the requirements of an approach for service composition which considers interdependent security as well as compliance (Section 4.1). The identified requirements are then being compared against state of the art approaches in service composition in order to identify research gaps (Section 4.2). A brief summary of the findings finishes this chapter (Section 4.3).

4.1 Requirements Analysis

An approach for service composition with consideration of interdependent security objectives and compliance is subject to several requirements. In the following these will be explicated and discussed based on the previous chapters.

Requirement 1 [COMPLETENESS]. *The approach must support all relevant composition patterns.*

This requirement may sound obvious, but as has been discussed in Section 3.2.4, the problem formulation affects which composition patterns can be captured and thus has an impact on the completeness of the approach. In some cases the assumption may be justified that not all composition patterns are important. Particularly this is the case for semi-structured processes which are not based on a defined process model. For instance, in case of travel planning the assumption is justified that processes are sequential in general. A corresponding sample process may be comprised of tasks for booking a hotel, a flight and on-site transportation (in any sequential order). The utilization of BPMN however is usually the consequence of structured processes which also include non-sequential structures. In this case, assuming that certain composition patterns cannot occur is unjustified. Hence the selection of the right problem formulation is not trivial.

Requirement 2 [CONSIDERATION OF MULTIPLE QOS]. *The approach must be able to cope with multiple general as well as domain-specific QoS attributes when evaluating service compositions, along with their individual semantics for all supported composition patterns.*

Similar to the first requirement, assuming that certain QoS dimensions are more important than others or that some QoS attributes are not important at all is unjustified. Hence the approach must be able to cope with all kinds of QoS attributes when evaluating service compositions, either general or domain-specific. Furthermore, the approach is required to handle every QoS attribute for all supported composition patterns in terms of aggregation schemes. Particularly, this means that the approach needs to be extensible in terms of aggregation functions for individual QoS attributes.

Requirement 3 [THOROUGH SEARCH SPACE EXPLORATION]. *The approach must not restrict the search space unjustified in terms of QoS importance.*

By default, the search space must be explored as thorough as possible by means of a MOO problem. Restrictions to the search space may only be applied in justified cases, for instance if the user explicitly states that some QoS attributes are more important than others. Hence the approach must be flexible enough to support possible user preferences in order to reformulate the selection problem from a MOO problem to a SOO problem and vice versa. Other search space restrictions which might bias the results are not permitted.

Requirement 4 [EFFICIENCY]. *The approach must be able to deal with process models of real-world size efficiently.*

If the approach is to be used in real-world scenarios, it has to be able to generate results for large process models in acceptable time. What is considered as "acceptable time" is generally any time span for generating a composition such that the (monetary) benefits of the composition outweigh cost of idleness before enacting it.

This is particularly important since change requests for IT systems (and eventually for processes which they support) are required in ever shorter time frames (cf. Section 2.2.2). Hence brute force approaches such as full enumeration are not acceptable to deal with process models of real-world size. Efficient approaches which exactly solve the service selection problem are usually modeling service selection as SOO problem which however restricts the search space as discussed in Section 3.2.4.3. Therefore, (meta-)heuristics which deliver near-optimal solutions in polynomial time and which thoroughly explore the search space by modeling service selection as a MOO problem are preferred.

Requirement 5 [CONSIDERATION OF INTERDEPENDENT SECURITY OBJECTIVES]. *The approach must select services with respect to multiple interdependent security objectives.*

Since security objectives are subject to interdependence (cf. Section 2.3), the approach needs to consider these interdependence effects and has to be able to determine if a composition suffices defined security requirements. However, as already stated, there is no formal model for interdependent security to the best of the author′s knowledge. Hence this requirement does not demand for formally provable security for service compositions, but for a security assessment.

Requirement 6 [COMPLIANCE-AWARENESS]. *The approach must suffice compliance requirements when selecting services.*

As has been explained in Section 3.4, compliance requirements are mandatory and thus have to be respected when selecting services in order to generate a composition. This requirement is especially crucial since compliance violations may lead to monetary as well as non-monetary losses such as customer and employee trust if they ever become publicly known.

4.2 State of the Art

QoS-aware service selection is a very active field of research since the end of the 1990s. Hence over time a plethora of approaches have been proposed. Works on alternative composition approaches, i. e. service (re-)planning and service mashups are not considered here as this would otherwise exceed the scope of this work. The literature review will be performed based on the previously defined requirements. That is, for each requirement it will be analyzed if they are covered by related works and to which extent.

4.2.1 Completeness

Regarding the completeness of approaches, two major groups can be differentiated, namely graph-based and combinatorial approaches (cf. Section 3.2.4).

Among the graph-based approaches, the works presented in [ZBD+03; YL04; ZBN+04; YZL07; HL13] utilize a classic DAG and thus can only handle sequences. Loops can be handled by unrolling them to sequences. However, YU et al. further propose to utilize an annotated DAG which empowers their approach to additionally consider splits and joins for AND and XOR patterns. OR patterns as well as discriminators are not supported [YZL07]. This model has also been utilized in [WCS+08; KIH11]. A slightly more comprehensive set of annotations has been proposed by MENASCÉ which is additionally able to cover OR patterns, but no discriminators [Men04].

The approaches presented in [CCG+06; CCG+07; AM10; YCY12; LRM+12] utilize customized DAGs to model service selection. In particular, CARDELLINI et al. utilize nodes to model service classes containing a set of concrete services. Similar to classic DAG-based approaches, only sequences are supported and loops are unrolled to sequences [CCG+06]. The same approach has also been employed in [AM10]. In a later work however, CARDELLINI et al. propose a more expressive custom DAG which additionally supports XOR and AND patterns as well as `pick` activities from BPEL (cf. [OAS07]). However, it still does not cover all relevant composition patterns [CCG+07]. Similarly, the approaches presented in [LRM+12; YCY12] are restricted to sequences, loops and parallelisms.

Another alternative graph presentation are directed graphs [AZA+05; GJ05; KIH12]. ARPINAR et al. use a directed graph in a similar way as DAG-based approaches. From a modeling point of view their approach has the advantage that loops do not need to be unrolled. However, as in DAG-based approaches, sequences and loops are the only composition patterns supported [AZA+05]. GRØNMO and JAEGER internally use a customized directed graph for QoS-aware service selection before transforming a solution to BPEL. The approach supports all composition patterns [GJ05]. KLEIN et al. use a customized directed graph where nodes can be either tasks or control flow elements. Although not explicitly stated, it is certain that the approach cannot represent discriminators. This is because the authors propose a tree structure for QoS aggregation which is derived from the graph representation. The tree structure however only considers the split but not the join. Hence it is obvious that the approach assumes that each split is closed by a join of the same type [KIH12].

Undirected graphs are employed in [XB05; WKI+12] but are similarly restricted as DAGs. While the approach presented in [XB05] is only able to represent sequences and unrolled loops, the modified graph model in [WKI+12] is additionally able to represent AND and XOR patterns.

The majority of combinatorial approaches formalize service selection as ILP problem [Lee03; AVM+04; CDE+05; CAH05; AP06; BSR+06; CDE+08; SPG+11]. As has been discussed in Section 3.2.4, this enables to consider all composition patterns, since the problem formulation is orthogonal to QoS aggregation and thus allows to consider any aggregation scheme for any QoS attribute. Alternative combinatorial problem formulations with similar expressiveness have been proposed in [JM07; LWL09; YZB11]. JAEGER and MÜHL propose a problem formulation incorporating aggregation functions which can be freely defined [JM07]. LIANG et al. formalize service selection as an assignment problem [LWL09]. YE et al. finally propose a problem formulation considering a hierarchy of services, i. e. application and utility services where services of the latter group provide the infrastructure for services of the former group to run on [YZB11].

Other approaches have been proposed which however do not support all composition patterns. WANG et al. proposed a semantic approach for selecting single services which does not support composition patterns at all [WLH07]. ALRIFAI et al. utilize a MIP problem to decompose global QoS constraints into local constraints. Instead of composition patterns, the approach considers the aggregation functions summation, multiplication and min which do not cover all composition patterns presented in Section 3.2.2 [AR09; ARN12]. MABROUK et al. use a decision tree approach which does not support discriminators [MBK+09]. MENASCÉ et al. use a non-linear problem formulation which supports composition patterns by means of BPEL constructs. However, not all composition patterns introduced in Section 3.2.2 are supported [MCD10]. LÉCUÉ and MEHANDJIEV formulate the problem as Constraint Satisfaction Problem (CSP) which only considers sequences, AND and OR patterns [LM11].

4.2.2 Consideration of Multiple QoS

From the perspective of supporting multiple QoS attributes, current approaches can be divided into several classes.

The by far biggest group employs a utility function to compute a score for service compositions based on the normalized and aggregated QoS of a solution (cf. Section 3.2.4.3). The respective approaches [ZBD+03; YL04; ZBN+04; GJ05; AP06; CCG+06; CCG+07; JM07; YZL07; AR09; LWL09; MBK+09; AM10; KIH11; YZB11; ARN12; YCY12; HL13] consider by default a certain set of QoS attributes which can be easily extended for new attributes along with individual aggregation schemes. Hence these approaches are considered as fully supporting multiple

QoS in the meaning of Requirement 2. The utility functions employed by these approaches have already been reviewed in Section 3.2.4.3.

Similar to utility functions, some approaches which employ GAs for solving the service selection problem, utilize the fitness function in a similar manner [CDE+05; CDE+08; WCS+08; LM11]. Multiple QoS attributes are weighted which results in a fitness score which, like an utility score, is used to determine the desirability of an individual solution (cf. Section 2.5.6). Analogously to approaches employing a utility function, the approaches of this group can also be easily extended for new QoS attributes along with their aggregation schemes.

A group of approaches model QoS attributes mutually independent, i. e. without computing an abstract utility value from different QoS attribute values. In particular, these approaches consider each aggregated QoS attribute either as decision criterion [Men04; MCD10] or as objective function on its own [CAH05; WKI+12]. What these approaches have in common is that they can be easily extended for new QoS attributes as well.

Semantic approaches for service selection [Lee03; AZA+05; WLH07; LRM+12] form a group on their own which differ significantly in terms of extensibility. The approach of LEE considers multiple QoS dimensions as mutually independent objectives, but not in the sense of a MOO problem. Instead, the approach allows for optimizing problem instances for different QoS objectives by means of a MCKP problem (cf. Section 3.2.4.2). This approach is extendable for new QoS attributes which however requires defining an individual objective function and capacity constraint for each additional QoS attribute [Lee03]. ARPINAR et al. propose a trade-off formula for weighting multiple quality criteria against semantic similarity of service interfaces which yields conceptual similarities to a utility function and is thus considered as fully expandable for new QoS attributes [AZA+05]. A similar approach has been proposed by WANG et al. But since the authors only consider single service selection, no QoS aggregation is supported [WLH07]. LAMA et al. proposed a semantic approach for cost-oriented service selection. Consequently, the only aggregation scheme supported is for cost while other QoS attributes and their respective aggregation schemes are not supported [LRM+12].

Some approaches [AVM+04; XB05; BSR+06; SPG+11; KIH12] propose a custom methodology for considering multiple QoS attributes. Except of the approach presented in [BSR+06], all proposals fulfill Requirement 2. AGGARWAL et al. propose an ontology-based approach to describe QoS attributes along with their metrics and aggregation schemes. This information is used by the LINDO solver for service selection [AVM+04]. For the graph-based modeling approach presented by XIAO and BOUTABA, the authors propose a custom shortest path algorithm which incorporates a fixed set of QoS attributes. Providing support for further

QoS attributes would thus require modifying this search algorithm which seems to be possible judged on the provided pseudo-code [XB05]. Berbner et al. propose an approach which supports a set of arbitrary QoS attributes which are used to compute a score using the weighted sum method. However, only additive, multiplicative and min aggregation schemes are supported which do not cover all semantics of QoS attributes [BSR+06]. Schuller et al. optimize for cost only and take other QoS attributes into consideration in terms of constraints [SPG+11]. Klein et al. propose service selection with respect to network locations of services aiming at minimizing network latency. Orthogonal to this network QoS, multiple service QoS attributes and their individual aggregation schemes are supported [KIH12].

4.2.3 Thorough Search Space Exploration

As has been stated previously, thorough search space exploration means modeling service selection as MOO problem if no user preferences have been explicated regarding QoS attributes, and otherwise as SOO problem with consideration of user preferences by means of weightings, etc. Related works can be classified into approaches which model service selection as SOO problem and those which model it as MOO problem. To the best of the author's knowledge, no current approach supports both model worlds.

All approaches which employ either utility [ZBD+03; YL04; ZBN+04; GJ05; AP06; BSR+06; CCG+06; CCG+07; JM07; YZL07; AR09; LWL09; MBK+09; AM10; KIH11; YZB11; ARN12; YCY12; HL13] or fitness functions [CDE+05; CDE+08; WCS+08; LM11] for selecting and ranking solutions model service selection as SOO problem. As has been discussed in Section 3.2.4.3, this allows for considering user preferences and reduces mathematical complexity but restricts the search space since some Pareto optimal solutions are usually missed. Hence these approaches are considered as partly fulfilling Requirement 3.

A number of approaches supports multiple mutually independent QoS attributs in service selection. Some approaches consider QoS attributes in terms of objective functions [Lee03; CAH05; WKI+12]. In a similar fashion, Wang et al. consider multiple QoS attributes by means of a matching degree which is determined employing fuzzy interference [WLH07]. Another major group of approaches have been proposed considering multiple aggregated QoS attributes without any further weighting or prioritization during the selection process. Particularly, this holds for the approaches presented in [AVM+04; Men04; AZA+05; XB05; MCD10; KIH12; LRM+12]. However, Arpinar et al. also propose a version of their selection approach with human interaction. In contrast to automatic selection, this version provides a user with a list of the highest ranked services for a task. The user can then select the service which best fits his/her needs [AZA+05]. The work of LAMA et al. only considers price during service selection [LRM+12]. As these approaches do not support prioritizing certain QoS attributes, these are also considered as partly fulfilling Requirement 3.

SCHULLER et al. model service selection as ILP and optimize for price only. Other QoS constraints are taken into account by means of restrictions [SPG+11]. Since no other QoS attributes than price are considered in the objective function, this approach is not considered to thoroughly exploring the search space by means of Requirement 3.

4.2.4 Efficiency

Proposed service selection approaches utilize a vast array of techniques for solving the problem. In general however they can be categorized into brute force, exact and (meta-)heuristic approaches.

The most general brute force approach, namely full enumeration is employed in [Men04; GJ05]. Since this quickly leads to exorbitant runtimes with increasing problem size, these approaches cannot be considered efficient.

Exact solving approaches for service selection can be classified into two groups. The first group utilizes solvers such as IBM OSL [Lee03; ZBD+03; ZBN+04], LINDO [AVM+04], CPLEX [AP06; SPG+11] and MATLAB with the SNOPT package [CCG+06; CCG+07]. The second group utilizes either general or custom algorithms for tackling the selection problem. YU et al. model service selection as combinatorial problem and use PISINGER's algorithm [Pis95] to tackle it [YL04]. In a later work, the authors model service selection as a graph-based problem as well as ILP problem. For the graph-based problem, the authors propose a custom shortest path algorithm which considers multiple constraints named MCSP for sequential and MCSP_General for general flow structures. The MCSP algorithm is also used by HEINRICH and LEWERENZ [HL13]. For the combinatorial model, the BBLP algorithm [Kha98] as well as the WS_IP algorithm are employed [YZL07]. MENASCÉ et al. formulate the problem as non-linear optimization problem which is solved using a custom algorithm called JOSes [MCD10]. YAN et al. propose a custom approach based on DIJKSTRA's algorithm [Dij59] to solve their graph-based approach [YCY12]. These approaches have in common that they are well-suited to exactly solve small to medium-size problem instances. However, solving problem instances of real-world size is usually infeasible. Hence these approaches are only considered as partly efficient in accordance to Requirement 4.

Among the (meta-)heuristic approaches, two major groups can be identified. The first group employs GAs for determining near-optimal solution sets for the service selection problem. The approaches presented in [CDE+05; JM07; CDE+08; WCS+08; LM11; YZB11; KIH12; WKI+12] employ custom GAs while CLARO et al. employ NSGA-II [DPA+02] which is known to be a fast metaheuristic delivering good results (cf. Section 7.2) [CAH05]. The second major group of approaches utilizes custom heuristics which leverage certain properties of the problem formulation in order to determine near-optimal solutions. This particularly holds for the approaches presented in [XB05; BSR+06; YZL07; LWL09; MBK+09; AM10; MCD10; KIH11; LRM+12]. Some heuristic approaches however need some further explanation due to their specific solution approach. The semantic approach of ARPINAR et al. aims at finding service selections which on the one hand fulfill QoS requirements and on the other hand provide sufficient semantic similarity in terms of the interfaces of selected services. That is, outputs of preceding services in a composition must provide a certain semantic similarity with the inputs of the directly following services. Hence the proposed Interface Matching Algorithm (IMA) aims at finding solutions providing a good tradeoff between these two properties [AZA+05]. WANG et al. determine QoS fulfillment via fuzzy logic. Solutions are generated using a custom GA before being evaluated using fuzzy inference [WLH07]. ALRIFAI et al. propose to globally decompose QoS requirements into local upper/lower bounds constraints. This is modeled as MIP and solved using the LPSolve solver. Since the decomposition is not exact, the approach leads to near-optimal solutions [AR09; ARN12]. All these approaches have in common that they deliver near-optimal solutions in polynomial time. Therefore they are considered as efficient in the sense of Requirement 4.

4.2.5 Consideration of Interdependent Security Objectives

From a security perspective, approaches for service composition can be divided into two groups. The first group of approaches does not consider any security aspects at all. In particular this holds for the works presented in [Lee03; ZBD+03; AVM+04; Men04; YL04; ZBN+04; AZA+05; CDE+05; CAH05; GJ05; XB05; AP06; BSR+06; CCG+06; CCG+07; JM07; WLH07; YZL07; CDE+08; WCS+08; AR09; LWL09; MBK+09; AM10; MCD10; KIH11; LM11; SPG+11; YZB11; ARN12; KIH12; LRM+12; WKI+12; YCY12; HL13].

The second group are approaches which formally consider a certain set of security objectives in the context of service composition. Since they do not consider any QoS requirements (cf. Section 3.3), they will not be further discussed here.

As has been previously stated, interdependence effects have not been considered yet in the context of service composition to the best of the author's knowledge.

4.2.6 Compliance-awareness

As already mentioned in Section 2.4, compliance issues have not been considered yet in the scope of service selection. However, as discussed in Section 3.4, compliance requirements are constraints either on single services or on a set of services which can be mathematically modeled. Hence the comparison presented here will focus on the potential of approaches to support compliance issues. More precisely, it will be analyzed if respective approaches support constraints on single as well as sets of services.

In general, all proposed approaches might be upgraded for the possibility for supporting local as well as scope-based constraints. In fact however only the approaches presented in [ZBN+04; AP06; AR09; ARN12] support expressing local constraints. Consideration of scope-based constraints is not supported by any of the approaches discussed here. Hence in their current form only the works presented in [ZBN+04; AP06; AR09; ARN12] are able to partly model compliance requirements.

4.2.7 Conclusion

Table 4.1 summarizes the fulfillment of the gathered requirements by state of the art approaches for service composition. As can be seen, especially the requirements 3, 5 and 6 are not completely fulfilled by any available approach so far. Since the reviewed works cover a large spectrum of problem modeling and solving approaches, their properties are being compared in greater detail in Table A.1. Due to its size however, the Table has been relocated to Appendix A.2.

4.3 Summary

In this chapter, a number of requirements has been established for an approach for service composition with consideration of interdependent security and compliance. A comparison of these requirements against state of the art approaches in service composition revealed that no approach completely fulfills them. Hence research gaps have been identified especially with respect to thorough search space exploration as well as consideration of interdependent security and compliance in service composition.

Table 4.1: *Fulfillment of requirements by related approaches*

		R1: Completeness	R2: Multiple QoS	R3: Thor. Search	R4: Efficiency	R5: Interdep. Sec.	R6: Compliance
Approach	Lee (2003)	●	●	◐	◐	○	○
	Zeng et al. (2003)	◐	●	◐	◐	○	○
	Aggarwal et al. (2004)	●	●	◐	◐	○	○
	Menascé (2004)	◐	●	◐	○	○	○
	Yu and Lin (2004)	◐	●	◐	◐	○	○
	Zeng et al. (2004)	◐	●	◐	◐	○	◐
	Arpinar et al. (2005)	◐	●	◐	●	○	○
	Canfora et al. (2005)	●	●	◐	●	○	○
	Claro et al. (2005)	●	●	◐	●	○	○
	Grønmo and Jaeger (2005)	●	●	◐	○	○	○
	Xiao and Boutaba (2005)	◐	●	◐	●	○	○
	Ardagna and Pernici (2006)	●	●	◐	◐	○	◐
	Berbner et al. (2006)	●	◐	◐	●	○	○
	Cardellini et al. (2006)	◐	●	◐	◐	○	○
	Cardellini et al. (2007)	◐	●	◐	◐	○	○
	Jaeger and Mühl (2007)	●	●	◐	●	○	○
	Wang et al. (2007)	○	◐	◐	●	○	○
	Yu et al. (2007)	◐	●	◐	●	○	○
	Canfora et al. (2008)	●	●	◐	●	○	○
	Wada et al. (2008)	◐	●	◐	●	○	○
	Alrifai and Risse (2009)	◐	●	◐	●	○	◐
	Liang et al. (2009)	●	●	◐	●	○	○
	Mabrouk et al. (2009)	◐	●	◐	●	○	○
	Ardagna and Mirandola (2010)	◐	●	◐	●	○	○
	Menascé et al. (2010)	◐	●	◐	●	○	○
	Klein et al. (2011)	◐	●	◐	●	○	○
	Lécué and Mehandjiev (2011)	◐	●	◐	●	○	○
	Schuller et al. (2011)	●	◐	○	◐	○	○
	Ye et al. (2011)	●	●	◐	●	○	○
	Alrifai et al. (2012)	◐	●	◐	●	○	◐
	Klein et al. (2012)	◐	●	◐	●	○	○
	Lama et al. (2012)	◐	○	◐	●	○	○
	Wagner et al. (2012)	◐	●	◐	●	○	○
	Yan et al. (2012)	◐	●	◐	◐	○	○
	Heinrich and Lewerenz (2013)	◐	●	◐	◐	○	○

Note: ●= fulfilled, ◐= partly fulfilled, ○= not fulfilled

Chapter 5

Approach

"I'm proud of my invention, but I'm sad that it is used by terrorists."

– Mikhail Kalashnikov

This chapter presents an approach to address the requirements identified in Chapter 4. First, an overview is given of how each requirement is being addressed in order to draw the big picture (Section 5.1). Subsequently, separate approaches for addressing interdependent security (Section 5.2) as well as compliance (Section 5.3) in service selection are being presented. Finally, the key points of these approaches are being summarized (Section 5.4).

5.1 Overview

In order to fulfill the requirements identified in Section 4.1, the proposed solution approach employs the following measures:

R1 (Completeness):
To avoid limitation with regards to composition patterns, a combinatorial problem formulation has been selected. More precisely, the MMKP has been chosen since this formulation is the most accurate problem definition for service selection. It does not suffer from any limitations regarding composition patterns or specific properties of the problem at hand such as different QoS dimensions (cf. Section 3.2.4.2).

R2 (Consideration of Multiple QoS):
A generic approach is proposed to define models for an arbitrary set of QoS attributes. Hence there is no limitation in this regard. The respective approach

will be presented in Section 5.2.1.

R3 (Thorough Search Space Exploration):
Depending on user preferences, the presented approach supports reformulating the optimization goal. In case that no preferences are given, the problem is defined as MOO problem. Otherwise, the problem is reformulated into a SOO problem using either the weighted sum method or lexicographic ordering of objectives (cf. Section 2.5.4). The respective approach is an integral part of the implemented proof of concept tool and hence will be presented in Section 6.2.2

R4 (Efficiency):
As has been previously stated in Section 3.2.4.2, there is to the best of the author's knowledge no efficient approach for solving the MMKP as MOO problem exactly. Hence the optimization problem is tackled with GAs (cf. Section 2.5.6) in order to deliver near optimal solutions to problem instances of real-world size in a reasonable amount of time.

R5 (Consideration of Interdependent Security Objectives):
A custom approach is proposed to assess the security of service compositions considering interdependence of an arbitrary set of security objectives. The approach is based on the notion of structural decomposition. In the presented approach, the assessment has the form of multiple objective functions so it can be seamlessly aligned with the service selection problem. Furthermore, SANDHU's notion of "good enough security" [San03] is respected by means of constraints in the optimization model. More precisely, a user defines her security requirements in terms of minimum function values which a protection function has to yield. The approach is being presented in Section 5.2.

R6 (Compliance-awareness):
A custom drop-in solution for considering compliance requirements is proposed. This enables existing GAs to consider compliance aspects and thus become compliance-aware. The respective approach will be presented in Section 5.3.

Utilizing these proposed measures, it is possible to construct a method for service composition with respect to interdependent security and compliance. Figure 5.1 shows the outline of this method. The first step is to define a process model for a business process in a notation such as BPMN as has been laid out in Section 2.1.1. For this given process model, the user needs to define security as well as QoS requirements in the second step. The third step is the selection of appropriate services to implement the process and is divided into two sub-steps. In sub-step

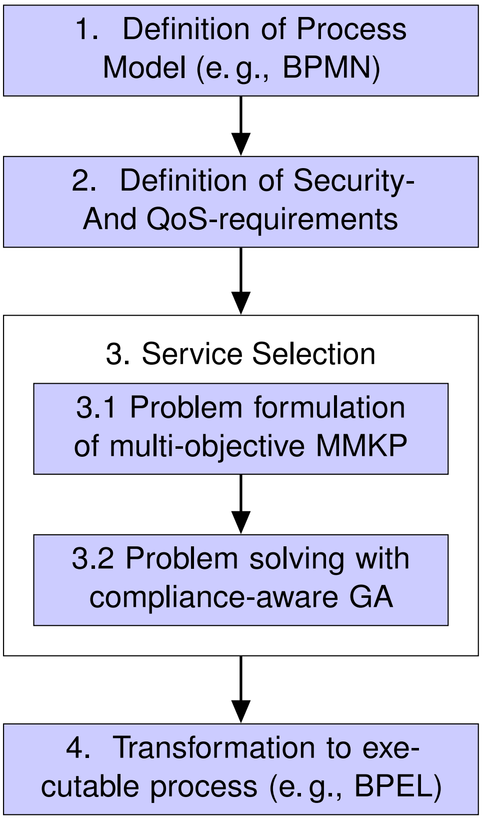

1. Definition of Process Model (e. g., BPMN)
2. Definition of Security- And QoS-requirements
3. Service Selection
3.1 Problem formulation of multi-objective MMKP
3.2 Problem solving with compliance-aware GA
4. Transformation to executable process (e. g., BPEL)

Figure 5.1: *Outline of proposed service composition method*

3.1 the selection problem is formulated as MMKP with multiple objective functions capturing the interdependencies of considered security objectives. Sub-step 3.2 is to solve the resulting multi-objective MMKP with a compliance-aware GA and to obtain a set of (Pareto optimal) solutions. Among these solutions the user needs to pick an alternative which best suits her preferences. The fourth and final step is to transform the selected solution to an executable form such as BPEL.

5.2 Assessing Interdependent Security of Service Compositions

This section presents a framework for assessing interdependent security. It introduces approaches for building QoS and domain models. QoS and domain model together will be called *security assessment model* in the remainder of this thesis.

5.2.1 Building QoS Models

As stated in Section 3.2.2, QoS attributes need to be aggregated in service composition according to the control flow of a process model and the attribute's semantics.

The first aspect has already been covered in Section 3.2.2 by means of aggregation schemes for three QoS attributes, namely Price, Availability and Response time. Here, the focus will be on the semantics of QoS attributes. The semantics of a QoS attribute depends on two essential features: *attribute type* and *measurement scale*. Measurement theory traditionally distinguished between the attribute types *binary*, *continuous* and *discrete* as well as the measurement scales[57] *nominal*, *ordinal*, *interval* and *ratio* [Ste46; JD88]. As both aspects are necessary in order to aggregate QoS attribute values correctly along process models, the author proposes to merge both worlds by describing QoS attributes in a generic way as 4-tuple $qa = (\mathcal{T}, \mathcal{M}, \mathcal{V}, \mathcal{A})$ consisting of the following elements:

- $\mathcal{T}$: Attribute type
- $\mathcal{M}$: Measurement scale
- $\mathcal{V}$: Set of valid values
- $\mathcal{A}$: Set of Aggregation schemes

The set of all QoS attributes qa, i. e. the QoS model, is denoted by QA. Following is a discussion of the components a QoS attribute qa is comprised of.

Valid values for $\mathcal{T}$ are *Binary*, *Continuous* and *Discrete* which mathematically correspond to the sets $\mathbb{B}$, $\mathbb{R}_+$ and $\mathbb{N}$ respectively.

$\mathcal{M}$ can have one of the following values: *Nominal*, *Ordinal ascending*, *Ordinal descending*, *Interval ascending*, *Interval descending*, *Ratio ascending* and *Ratio descending*. The basics of these measurement scales have already been introduced in Section 3.2.3. Here, some additional aspects regarding nominal scales for QoS attributes will be discussed. As has been stated, nominal scales are employed if no natural order relation exists for the elements in $\mathcal{V}$, i. e. if each element defines a category of items which is independent from the remaining categories. A typical example for a QoS attribute employing a nominal scale is *encryption algorithm*. There is no natural ordering of cryptographic algorithms by means of security. However, it is possible to construct an order, e. g. based on a set of hypotheses. Respective approaches have been proposed in e. g. [LV00; LV01; Len06; OH04; BBB+12; Sma12; Bun13b; Bun13a; NSA13][58]. However this is a subjective matter and as

[57] The author is aware that this typology, especially in case of nominal and ordinal scales, is subject to controversy (cf. e. g. [VW93]). However, as these critics mostly relate to issues about statistical interpretation of data and as this typology is still widely in use, it is adopted here.

[58] A web-based tool for cryptographic key length recommendation based on these approaches is available at: `www.keylength.com`

Table 5.1: *Allowed spaces for $\mathcal{V}$ with respect to variable type and measurement scale*

Type Scale	Binary	Continuous	Discrete
Any ascending	–	min - max	min - max
Any descending	–	|min| - |max|	|min| - |max|
Nominal	Single value/ Enumeration	–	–

such it seems appropriate to associate a utility value with each category in order to allow users to express their preferences regarding each category. Effectively, this means transforming a nominal scale into an ordinal scale. Applied to the example with the QoS attribute encryption algorithm, this means assigning a utility value for each combination of encryption algorithm and key sizes. These utility values might be for instance derived from the approaches mentioned above. The set of these assignments then forms the content of $\mathcal{V}$ (cf. below).

The space of $\mathcal{V}$ depends on the attribute type as well as the measurement scale. Table 5.1 shows how $\mathcal{V}$ is specified for each combination. Some combinations are forbidden (denoted "–") as they yield no additional expressiveness but introduce additional overhead for handling. In case of either an ordinal, interval or ratio scale, the content of $\mathcal{V}$ is defined by determining minimum and maximum values. If the order relation is descending, minimum and maximum values are expressed as absolute values. In case of a nominal scale, $\mathcal{V}$ consists of either a single value or an enumeration of utility values where the sum must not exceed 1. As the only valid attribute type for nominal scales is binary, it may sound contradictory to provide more than one utility value. One needs however to differentiate between different categories (which are each associated with an utility value) and the binary decision if a certain category is supported by a service alternative or not. The latter is what attribute type refers to. Equation (5.1) shows parts of a utility function with two nominal properties *Encryption* and *SecurityProtocols*, each containing more than one value.

$$\ldots + \underbrace{\overbrace{0.01b_{i+1}}^{AES-128} + \overbrace{0.03b_{i+2}}^{AES-192} + \overbrace{0.06b_{i+3}}^{RSA-512}}_{Encryption} + \underbrace{\overbrace{0.02b_{i+4}}^{HTTPS} + \overbrace{0.05b_{i+5}}^{TLS}}_{SecurityProtocols} + \ldots \tag{5.1}$$

Each category is associated with an utility value and a binary decision variable b. The utility values given here express the importance of each category with respect to the others in the same group as well as its overall importance with respect to the utility function. Mixing these two aspects in one value might lead to an increased complexity in determining sound utility values. This aspect will be discussed more thoroughly now and it will be especially shown that the proposed reformulation for providing an enumeration

a) can be sensibly rendered into a single value measurement,

b) is in no contradiction to the "traditional" view without associating a utility value with each category in $\mathcal{V}$ and

c) eases the definition of security assessment models (cf. Section 5.2.2).

Technically speaking, the proposed approach clusters similar items in $\mathcal{V}$ (e. g., different encryption algorithms) which enables to

a) determine the impact of a group as a whole on some utility function and

b) rank categories within a group by means of different utility values.

This goes beyond the pure binary decision if a service supports a particular feature but implicitly defines a (partial) ordering of similar categories at the same time. Given a nominal attribute with $\mathcal{V}$ containing p elements and a linear utility function, the proposed approach replaces (in mathematical terms)

$$\ldots + w_1 b_1 + w_2 b_2 + \ldots + w_p b_p + \ldots \tag{5.2}$$

with

$$\ldots + w(u_1 b_1 + u_2 b_2 + \ldots + u_p b_p) + \ldots, \tag{5.3}$$

where w_i is a weighting factor (cf. Section 5.2.2), b_i a binary decision variable and $u_i \in \mathbb{R}$ an utility value (which is contained in $\mathcal{V}$). The utility values in (5.3) can be obtained from the weightings in (5.2) by the following computation:

$$u_i = \frac{w_i}{\sum\limits_{j=1}^{p} w_j} \tag{5.4}$$

where $\sum\limits_{j=1}^{p} w_j \neq 0$. Thus, the proposed approach re-formulates a sequence of binary decision and utility variables into a single measurement value. Moreover, a formulation as in (5.3) enables domain experts to separate the question of what impact a certain group of categories has from the question of how to order these categories. This is especially advantageous in case that the weighting of a single group needs to be modified or if a single group changes in terms of number and importance of categories. While the traditional approach requires in either case modifications on all utility values in the utility function, the proposed approach requires that either the group weightings or the single utility values in a group need to be adjusted. The author believes that this vastly reduces the complexity of building security assessment models. Equation (5.5) shows the result of applying the proposed method on the sample in Equation (5.1).

$$\ldots + w_i(0.1b_{i,1} + 0.3b_{i,2} + 0.6b_{i,3}) + w_{i+1}(0.29b_{i+1,1} + 0.71b_{i+1,2}) + \ldots \tag{5.5}$$

This reformulation yields similarities with a metric interpretation of nominal properties. However, it should be noted that the proposed interpretation takes place on a meta-level (i. e. utility values associated with each category). Hence, the homomorphic properties of the nominal scale are being preserved.

Finally, $\mathcal{A}$ is a set of 2-tuples (*Composition Pattern*, *Aggregation Scheme*). As has been shown in Section 3.2.2, aggregation schemes of attributes vary depending on the composition pattern of the process model. Therefore it is necessary to list which aggregation scheme to use for each composition pattern. As this is a matter of semantics and as such usually different for each QoS attribute, this task cannot be fully automatized.

5.2.2 Building Domain Models

A domain model describes a set of security goals and their interrelationships. It consists of protection functions which capture these structures by means of interdependent utility functions.

5.2.2.1 Defining Protection Functions

The basic idea behind a protection function is to assess the utility of service compositions towards fulfilling a certain security objective. In this regard, the utilized definition is analogous to the idea of a utility function as typically employed in service composition (cf. Section 3.2.4.3). A protection function semantically serves several purposes, namely to

- describe the influence of single QoS attributes, which are realized by technical facilities, on achieving a certain protection level,
- weight the influence of different QoS attributes in case of non-uniformity and

- consider the varying effectiveness of different technical implementations for the same service property. Here, effectiveness is understood as "the degree to which something is successful in producing a desired result."[59]

At this, the proposed approach is based on the notion of structural decomposition which aims at identifying measurable components for high-level requirements [WW97; HHA04; SA09]. Since requirements engineering is out of the scope of this thesis, it is assumed in the remainder that the respective steps have already been taken beforehand. In particular, it is assumed that necessary security objectives have already been identified and clearly differentiated against each other. For a given set of security objectives, structural decomposition consists of three steps [WW97; SA09]:

1. For each security objective, identify components that contribute to the success of the goal. Arrange the components as subordinate nodes in a tree structure.
2. Examine the leafs if further decomposition is needed. If this is the case, apply the first step on the respective components as well.
3. Terminate decomposition if no leaf can be further decomposed. At termination, the leafs of the tree should be mutually independent measurable components.

Figure 5.2 shows an example from [SA09] for decomposing *authentication* into more fine-grained components. This decomposition has been performed in the scope of the GENOM project which aims at providing a secure message oriented middleware. As can be seen the decomposition identified two major components contributing to the security objective authentication. The first is the identity itself and its impact is determined by the question what properties it has in terms of uniqueness, structure and integrity. The second major component is the employed mechanism to authenticate entities (e. g. username-password and certificates) and the impact of this component is determined by its reliability and integrity.

Given such a decomposition with n identified components, the security of different configurations can be assessed as a function as follows [WW97]:

$$f_{objective} = \sum_{i=1}^{n} w_i u_i \tag{5.6}$$

where w_i and u_i denote the impact to a security objective and the utility of a component respectively. This formulation assumes that each component has an

[59] `http://www.oxforddictionaries.com/definition/english/effectiveness`

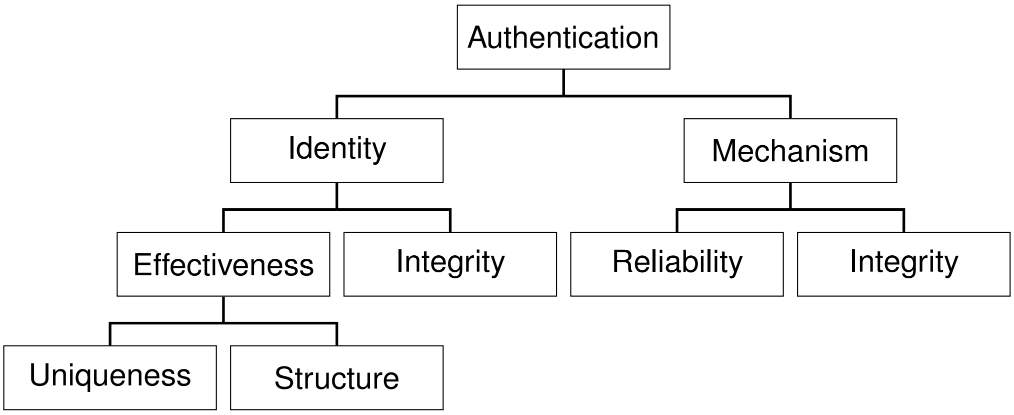


Figure 5.2: *Example for structural decomposition (cf. [SA09])*

impact on its own on a certain security objective and that the security level of a system is determined by the collective impacts of all components. Applying this scheme to the example depicted in Figure 5.2 thus would yield a function with five components. An important question in this regard is how to determine appropriate weightings and utility values. WANG and WULF suggested to employ decision support tools such as the Analytic Hierarchy Process (AHP) [Saa90]. Due to its extent, AHP cannot be discussed here. To validate obtained weightings and utility values, WANG and WULF suggested to utilize either empirical relations or formal experiments. A concrete method is however not defined [WW97].

A research gap that remains and which is addressed with the proposed approach is the consideration of different degrees of effectivity for different technical implementations of the same feature. In the context of service selection this needs to be addressed at the level of single services during selection. In order to align the concept of effectivity seamlessly with service selection, the approach proposed here considers it during QoS aggregation. Before presenting this approach in Section 5.2.2.2, it is first discussed how to align structural decomposition with service selection.

In the following it is assumed that measurable components for each security objective have been identified and included in the QoS model and that the importance of each component for each security objective has been determined utilizing e. g. AHP. Then each security objective can be expressed as a protection function which receives as input the aggregated and normalized QoS vector $Q^{'}$ of a service composition and returns a utility value which represents an assessment on the fulfillment of a security objective. The importance of each measurable component for a protection function can then be expressed as an attribute weighting in the interval $[-1, 1]$ where a negative value indicates a negative impact, 0 no impact and a positive value a positive impact respectively.

Let D denote a domain model, i. e. a set of n protection functions pf_i. A protection function pf_i, $i \in \{1, \ldots, n\}$, is defined as follows:

$$pf_i(\mathcal{Q}') = \sum_{j=1}^{m} w_{ij} \mathcal{Q}'^j \tag{5.7}$$

where

$$\mathcal{Q}' = (\mathcal{Q}'^1, \ldots, \mathcal{Q}'^m) : \text{Aggregated and normalized global QoS vector and}$$
$$\sum_{j=1}^{m} |w_{ij}| = 1, w_{ij} \in [-1, 1] : \text{Attribute weightings for protection function } i.$$

Analogously to the approach presented in [WW97], it is assumed that each QoS attribute of a component has an impact on its own on a security objective. The security level of a system regarding security objective i is then determined as the sum of all impacts. From a semantic point of view, protection functions are equivalent to the assessment functions employed in [WW97; HHA04; SA09]. Protection functions form the basic elements that will be used in the following sections for modeling interdependence and assessing service compositions.

5.2.2.2 Normalization of QoS Attributes

In previous sections the necessity for aggregating (cf. Section 3.2.2) as well as normalizing (cf. Section 3.2.3) QoS attributes has been explained in detail. The usual order in which this happens is aggregation before normalization (cf., e. g., [ZBD+03; YCY12]). However, as already stated, the proposed approach aims at considering different degrees of effectivity for the QoS attributes of each selected service in a normalized range $[0, 1]$. Hence it is necessary to perform these actions in reverse order, i. e. normalization before aggregation.

Normalization of ascending or descending QoS attributes employing either ordinal, interval or ratio measurement scales is performed with the method proposed by ZENG et al. which has been presented in Section 3.2.3. The reason for this decision is that this normalization method is the most widely used.

In case of nominal measurement scales, normalization is performed based on the utility values associated with each category. The normalized QoS value is simply the sum of all utility values associated with the features which are provided by a service alternative:

$$q'^{\alpha}_{ij} = \sum_{i=1}^{f} u_i b_i, \tag{5.8}$$

where

f : The number of features defined in $\mathcal{V}$,

u_i : the utility value associated with feature i and

$b_i \in \{0,1\}$: a binary decision variable indicating if feature i is supported by a service alternative or not.

Analogously to the definition of protection functions (cf. Section 5.2.2.1), it is assumed that each feature has an impact on its own on the security level of a component and that they mutually complement each other. In the remainder of this thesis, the following notation will be utilized to represent normalized sets of features: $\{$ name$_1 = u_1$, name$_2 = u_2, \ldots,$ name$_f = u_f\}$. In this notation, name$_i$ is a textual representation of the feature name (e. g. RSA and AES in case of encryption algorithms) while u_i represents the utility value which is associated with the feature and interpreted in the course of assessment.

Effectivity of a service alternative s_{ij} is expressed by means of an effectivity vector e_{ij}. This vector contains one effectivity value $e_{ij}^{\alpha} \in [0,1]$ per quality attribute q_{ij}^{α}. Given a normalized quality vector q'_{ij} and an effectivity vector e_{ij}, aggregation schemes are applied on the product of q_{ij}^{α} and e_{ij}. For instance, the multiplicative aggregation scheme

$$Q^{\alpha} = \prod_{i=1}^{k} q_{ij}^{\alpha}$$

thus becomes

$$Q'^{\alpha} = \prod_{i=1}^{k} q'^{\alpha}_{ij} e^{\alpha}_{ij}$$

An important question which should be addressed here is how to determine effectivity values. As stated above, an effectivity value indicates how well a domain expert assesses the technical implementation of a facility. As the assessment method can be expected to be generally different for different types of organizations, employed standards, legal requirements and so on, there is no simple answer to this question. However, in general it appears that constructing an assessment framework is inevitable. Such a framework needs to address the question of what factors have which impact on the assessment. This shall be illustrated employing the example of the QoS attribute encryption again.

In case of a certified state-of-the-art implementation of a cryptographic algorithm, a domain expert would assign a high effectivity value. If the implementation is known to be outdated and buggy, a medium value would be assigned and finally a low (or even minimum) value if the provider does not disclose any information regarding the utilized implementation. Thus the first essential task for domain experts in this context is the question of how to determine essential influence factors on effectivity for each quality attribute. Secondly, the impact of each identified factor on the effectivity of a QoS attribute needs to be determined. But since such frameworks quickly tend to become very complex, this topic will not be discussed in more detail here. In fact, the work required to be done by domain experts here is very similar to defining protection functions and utility values for binary attribute values. Hence it is to be expected that the construction of a consolidated evaluation framework would be more feasible than developing three separate frameworks.

5.2.2.3 Modeling Interdependence between Protection Functions

As discussed in Section 2.3.3, WOLF and PFITZMANN determined that interdependence between two security objectives might be strengthening, weakening or implicating. This section shows how to express these kinds of interdependence with protection functions.

Strengthening and weakening effects of a protection function pf_A on another protection function pf_B can be modeled by means of a interference factor $\lambda \in \mathbb{R}$. Let an update of protection function pf_B into pf_B' be defined as follows:

$$pf_B' = pf_B + \lambda_A pf_A \tag{5.9}$$

Depending on the sign of λ_A, the interference relation is interpreted as either strengthening or weakening. In case that either λ_A, pf_A or pf_B is 0, the particular strengthening/weakening relationship between pf_A and pf_B has no effect on assessment results.

Modeling implication relationships between two protection functions pf_A and pf_B (denoted $pf_A \rightarrow pf_B$) requires some more effort. It is insufficient to simply compare the order of the results of pf_A and pf_B, because the primary requirement is that a service has a positive protection value to be considered true. As such, at first it needs to be defined what “true” and “false” mean in the context of protection functions. In the following, let a function value $pf_k > 0$ be defined as “true” or in words: “Security goal achievable with the given quality attribute values.” Accordingly, a function value $pf_k \leqslant 0$ is interpreted as “false” or in words: ”Security goal not achievable with the given quality attribute values.” With these rules, the implication relation between pf_A and pf_B can be formulated in the usual boolean way as given in Table 5.2.

As can be seen, it must be ensured that no situation occurs where $pf_A > 0 \wedge pf_b \leqslant 0$. Otherwise the implication relation between pf_A and pf_B would not

Table 5.2: *Implication between security functions pf_A and pf_B*

pf_A	pf_B	$pf_A \rightarrow pf_B$
> 0	> 0	*true*
> 0	$\leqslant 0$	*false*
$\leqslant 0$	> 0	*true*
$\leqslant 0$	$\leqslant 0$	*true*

be true. To verify this property and thus that the implication relation between pf_A and pf_B is true, the following two step approach is proposed:

1. Attribute Chaining
2. Pessimistic Attribute Weighting

In the following these steps will be explained in more detail.

1) Attribute Chaining:

Let QA_A and QA_B denote the sets of quality attributes with a non-zero weighting in pf_A and pf_B respectively. Then the following conditions must hold:

1. $QA_A \subset QA_B$
2. The sign of each quality attribute $qa_A \in QA_A$ must be the same as in QA_B

If these conditions hold, pf_A becomes a sufficient condition for pf_B which means that any quality attribute which effects pf_A will effect pf_B as well and vice versa. Following are two short numerical examples to show the necessity of these two properties. In general, if w_{ij} and e_{ij} are skipped in the following, they will be assumed to be (-)1, i. e. without impact on a protection function. Consider the following two protection functions pf_A and pf_B ($QA_A \nsubseteq QA_B$) defined as follows:

$$pf_A = q'_{1j}$$
$$pf_B = q'_{2j}$$

In case that $q'_{1j} = 1 \wedge q'_{2j} = 0$ one receives $pf_A = 1 \wedge pf_B = 0$ which invalidates implication relationship between pf_A and pf_B. However, the first condition alone is insufficient. Consider the following modified protection functions pf_A and pf_B which violate the second condition:

$$pf_A = q'_{1j} - q'_{2j} + q'_{3j}$$

$$pf_B = -q'_{1j} + q'_{2j} + q'_{3j}$$

In case that $q'_{1j} > 0 \wedge q'_{2j} = 0 \wedge q'_{3j} = 0$ one receives as solution $pf_A > 0 \wedge pf_B < 0$ which also invalidates the implication relationship. Hence both conditions are necessary to guarantee the validity of the implication relationship.

2) Pessimistic attribute weighting:

Given the sufficient condition defined in the first step, the question arises how to ensure that $pf_A > 0$ holds whenever $pf_B > 0$ holds in terms of factor weightings. This is achieved by determining the maximum factor weightings which may be assigned to all quality attributes with a negative sign in pf_B. This is what the term pessimistic attribute weighting refers to. In particular, the proposed approach ensures that $pf_B > 0$ in the worst case which is defined as follows here:

- The factor with minimum positive weighting has a minimal QoS attribute value and minimal effectivity,
- all factors with negative weighting have maximum QoS attribute values and maximum effectivity.

Since the QoS attributes of pf_A and pf_B are chained, it may happen in this worst case that $pf_A > 0$ and $pf_B < 0$ as has been seen in the last example. This in turn would break the implication relationship. Hence it is necessary to make sure that in this worst case $pf_B > 0$ holds which will ensure that the implication relationship between pf_A and pf_B remains valid in all other cases as well. The proposed approach is based on the assumption that for any normalized QoS attribute it is possible to define practical lower boundaries $q'_{ij,lb}$ and $e_{ij,lb}$ which exhibit the following properties: $0 < q'_{ij,lb} \leqslant 1$ and $0 < e_{ij,lb} \leqslant 1$. The interpretation of $q'_{ij,lb}$ is that by experience a QoS value never falls below this value, even though the metric has been defined in a broader range. Similarly, $e_{ij,lb}$ represents a lower boundary that by experience is achieved by all providers, no matter how bad their technical realization is.

With these lower boundaries in mind, let W_+ and W_- be two sets containing all positive and negative factor weightings in pf_B as follows:

$$W_+ = \{w_{ij} q'_{ij,lb} e_{ij,lb} \mid w_{ij} > 0\}$$
$$W_- = \{w_{ij} \mid w_{ij} < 0\}$$

The reason that q'_{ij} and e_{ij} are not included in W_- is that these are assumed to be 1 in the worst case. Hence they can be omitted without any loss in generality.

Let further Σ_{W_-} be the absolute sum of all factor weightings $w_k \in W_-$:

$$\Sigma_{W_-} = \sum_{k=1}^{|W_-|} |w_k \in W_-| \tag{5.10}$$

Then, Σ_{W_-} can be utilized to determine the maximum weighting that negative factors are allowed to have such that $pf_A > 0$ whenever $pf_B > 0$ utilizing the following expression:

$$\Sigma_{W_-} < \min(W_+) \tag{5.11}$$

If this criterion does not hold, it cannot be guaranteed that the implication relationship will hold in all cases. Particularly, if $\Sigma_{W_-} \geqslant 1$, implication cannot be guaranteed for pf_A and pf_B in their current form since no single attribute with weighting $w_k \in W_+$ will be able to overcompensate all attributes with overall weightings Σ_{W_-} in the worst case. Hence a reformulation of the underlying domain model is required. Following is a numerical example to clarify this. Consider the following two protection functions:

$$pf_A = q'_{1j}$$
$$pf_B = 0.5q'_{1j} - 0.7q'_{2j} + 0.6q'_{3j} + 0.6q'_{4j}$$

Obviously both attribute chaining criteria hold for these functions. However in case that $q'_{1j} = 1 \wedge q'_{2j} = 1 \wedge q'_{3j} = 0 \wedge q'_{4j} = 0$ the results are $pf_A = 1 \wedge pf_B = -0.2$ which invalidates the implication relationship. Hence in the following the method of pessimistic attribute weighting will be applied in order to determine the factor weighting such that q'_{2j} will no longer invalidate the implication relationship between pf_A and pf_B. At first, let the lower boundaries for $q'_{1j}, q'_{2j}, q'_{3j}, e_{1j}, e_{2j}, e_{3j}$ be defined as follows:

$$q'_{1j,lb} = 0.8, e_{1j,lb} = 0.8$$
$$q'_{2j,lb} = 0.8, e_{2j,lb} = 0.8$$
$$q'_{3j,lb} = 0.8, e_{3j,lb} = 0.8$$

The sets W_+ and W_- then contain the following elements:

$$W_+ = \{0.32, 0.384, 0.384\}$$
$$W_- = \{-0.7\}$$

Based on W_- one can compute that $\Sigma_{W_-} = 0.7$. As can be seen, the established validity criterion $\Sigma_{W_-} < \min(W_+)$ is not fulfilled. One alternative formulation of pf_B which respects this criterion is the following:

$$pf_B = 0.5q'_{1j} - 0.3q'_{2j} + 0.4q'_{3j} + 0.4q'_{4j}$$

Recomputing the case given above yields a function value of $pf_B = 0.532$ which results in a valid implication relationship between pf_A and pf_B.

5.2.2.4 Known Limitations

The proposed approach for modeling interdependence relations currently only works under two conditions:

1. There must not be cyclic dependencies between protection functions[60]
2. In case of implication relationships $A \rightarrow B$, A must not have a predecessor.

In order to check these properties, the security assessment model is transformed into a directed graph at first. For validating condition one, TARJAN's algorithm [Tar71] is employed in order to check if the graph contains any strongly connected components. Checking the second condition is trivial. The developed support tool (cf. Chapter 6) checks at design time if these conditions hold and shows an error message if this is not the case.

5.2.3 Discussion

The proposed method for security assessment does not guarantee security of service compositions, but gives domain experts a tool to evaluate different composition alternatives with respect to certain security goals. Following is a discussion about the merits of the proposed approach compared to classical formal approaches as well as its general limitations.

As has been stated in Section 2.3.2, deciding if a program yields a certain nontrivial property is undecideable. The consequence of this observation from a methodical perspective is that formally verifying security of a composition would at first require to manually verify the security of each service in a composition. Several frameworks exist for this task with the *Common Criteria*[61] being a prominent

[60] Such as between *confidentiality* and *anonymity* in the security model of WOLF and PFITZMANN (cf. Section 2.3.3).

[61] `https://www.commoncriteriaportal.org/`

example. This is however time and cost intensive and hence only few products have been formally verified[62]. After this initial step, the composition as a whole would require to be verified as well. The reason is that "even secure components can be assembled in ways that make them insecure" [PC10, p. 50]. As an example consider interactions of cryptographic protocols. Even protocols which are secure when utilized in isolation can be combined in ways such that the interaction opens up new security holes [KSW98]. This example underlines a general rule in security which is that "security does not necessarily compose" [KSW98, p. 103]. One possibility to avoid such dangers is to again formally verify that the composition is secure as well. A particular method for cryptographic protocols is the universal composability paradigm [Can01]. However, even multiple formal verifications (i. e. at the service and the composition level) are no guarantee that a system is secure since there can still be design errors or software bugs. Especially the absence of bugs can never be proved [Dij72].

In this context, the proposed method is to be seen as a tool which can be employed to assess security of distributed systems. Since assessment is performed at runtime, results are delivered immediately without requiring any manual steps. Besides of the procedure described in the previous sections which completely relies on domain experts' opinion, it is also possible to define protection functions by means of security metrics. In this case, each protection function models a single security metric and thus assesses a composition with regards to this certain metric. This leads to an assessment of compositions which is much less subjective and hence much easier to be agreed upon. In general, this variant of the proposed approach can be applied in two different ways by either employing

1. general-purpose metrics or
2. organizational security metrics.

In either case, modeling protection functions is straight-forward and is performed as described in Section 5.2.2.1. The only difference is however, that each protection function consists of a single property with a weighting of 1. As an example consider applying the modified approach to define a protection function for availability based on a single QoS property uptime probability as follows:

$$pf_{Availability} = 1.0\mathcal{Q}'^{1} \tag{5.12}$$

[62] A formal verification with *Common Criteria* corresponds to a EAL 7 certification, the highest level defined in the framework. This level is however rarely achieved by any product. Out of 1955 officially certified products only six have have been certified at this level (`https://www.commoncriteriaportal.org/products/stats/`).

Table 5.3: *Sample data for comparing the results of the modified approach and average availability*

Alternative 1		Alternative 2	
Service	Uptime Prob.	Service	Uptime Prob.
A	90	A	85
B	99	B	80
C	95	C	90
Modified approach	2.32	Modified approach	0.82
Average avail.	94.67	Average avail.	85

Assuming that appropriate aggregation schemes were chosen (e. g. addition for sequences), the results obtained by the proposed method and computing the average availability will be trivially identical in terms of preferable composition alternative. The sample data presented in Table 5.3 illustrates this.

Similarly, organization-specific metrics can be employed with the proposed approach as well. In this case however, the data is usually not publicly available but needs to be provided by either the organization itself or by a some trusted authority. Typical security-related metrics include:

- number of SLA violations
- number of reported data breaches
- average time-to-react in case of security incidents

In other words, monitoring data is being utilized to assess compositions. A hybrid approach employing general-purpose and organization-specific metrics is also possible. From the author's point of view, latter metrics are also suited to assess compositions with organization-specific risk control methods. Hence, the proposed method would enable not only to assess QoS and interdependent security, but would also allow to evaluate possible financial risks of different composition alternatives.

5.3 Compliance-Aware Service Selection

In this section an approach is presented for aligning compliance checking with service selection. First an outline of the proposed approach is given. Subsequently the single steps of the proposed approach are discussed which consist of a feasibility check, a detection method for compliance violations and a repair operation.

5.3.1 Outline of Approach

Since GAs are commonly used for service selection (cf. Section 3.2.4.4), the proposed approach is a drop-in solution. This enables to extend existing GAs for consideration of compliance aspects even if they are already used in productive systems. As a proof-of-concept, NSGA-II [DPA+02] has been extended to consider compliance aspects since evaluations proved it to be the fastest metaheuristic[63]. The resulting algorithm was called Compliance-Aware Genetic Algorithm (COMPAGA). Figure 5.3 gives a conceptual overview of the proposed approach. The modified GA works as follows:

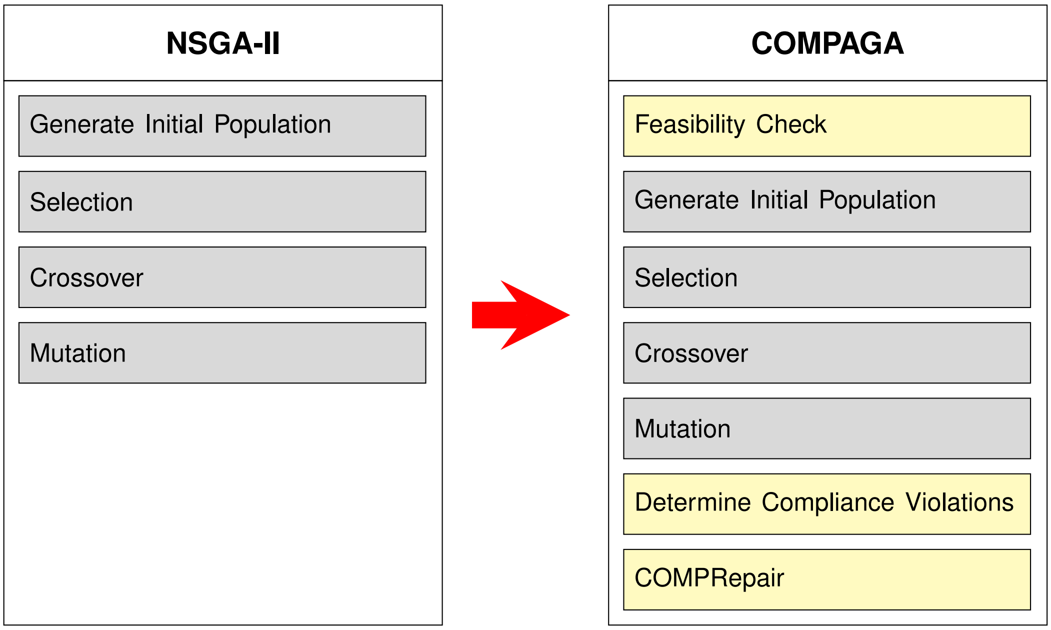


Figure 5.3: *Extending NSGA-II to COMPAGA*

1. Perform an initial feasibility check. If no compliant solution exists for the given problem instance, abort.
2. Generate an initial population.
3. Apply GA operators selection, crossover and mutation as usual on the population.
4. Check each individual solution for compliance violations.
5. If an individual solution violates compliance, apply the custom GA operator COMPRepair at a probability of p_{rep}[64].

[63] An overview of the employed metaheuristics is provided in Section 7.2.

[64] The question of how to determine a "good" probability p_{rep} is covered in Section 7.1.

6. While the convergence criterion is not met, go to step 3.

Algorithm 5.1 provides a simplifying view on COMPAGA as pseudo-code. In particular, all parts which are specific to NSGA-II have been either simplified or completely omitted in order to focus on the extensions introduced by COMPAGA. These extensions will be explained in detail in the following sections.

Algorithm 5.1: COMPAGA

```
$P$ := Process Model;
$SA$ := Set of all service alternatives;
if $feasibilityCheck(P, SA) = false$ then return $\emptyset$;
$pop$ := generateInitialPopulation();
$gen$ := 0; $cond$ := false;
while $cond = false$ do
    selection($pop$);
    crossover($pop$);
    mutation($pop$);
    foreach $solution \in pop$ do
        $SC$ := convertToComposition($solution$);
        $V$ := detectComplianceViolations($P$, $SC$);
        $rand$ := random[1..100];
        if $V.length > 0$ and $rand > p_{rep}$ then
            $SC$ := COMPRepair(P, SC, V);
            $solution$ := update($SC$);
        end
    end
    $gen$ := $gen$ + 1;
    if convergence criterion met then $cond$ := $true$;
end
return $pop$;
```

5.3.2 Feasibility Check

The initial feasibility check is the first step of COMPAGA and is performed only once. The goal of this step is to determine for a given process model and service alternatives if there is at least one compliant solution without considering user requirements. If this is not the case, it is infeasible to optimize since no solution is possible for the given problem instance, i. e. no solution will be able to meet all compliance and user requirements simultaneously. In case that at least one compliant solution exists, there is a chance that a solution fulfills both compliance and user requirements. Hence, the existence of a compliant solution is still no guarantee that for the considered problem instance at least one valid solution exists.

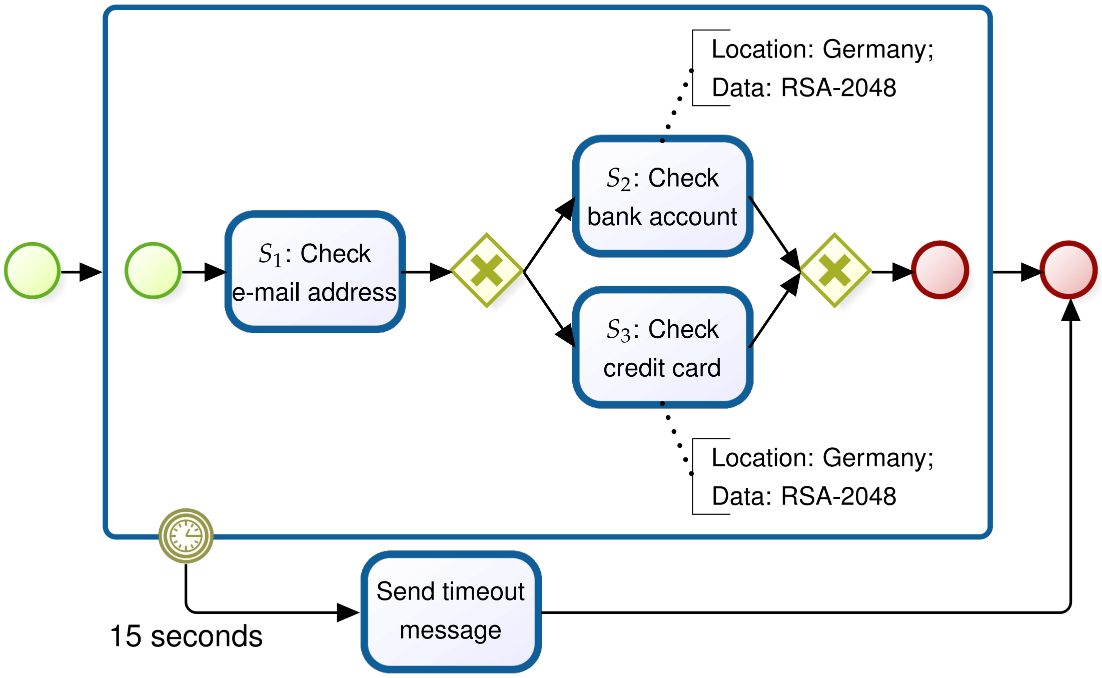


Figure 5.4: *Sample process with compliance requirements*

Table 5.4: *Service alternatives for sample process with compliance requirements*

Service Class	Service Candidate	Location	Data Encryption	Response Time
S_1	s_{11}	Germany	RSA-1024	7
	s_{12}	Ireland	RSA-1024	4
	s_{13}	Austria	RSA-2048	5
S_2	s_{21}	Belgium	RSA-4096	3
	s_{22}	Germany	RSA-3072	2
	s_{23}	France	RSA-3072	4
	s_{24}	Germany	RSA-1024	5
S_3	s_{31}	Germany	RSA-2048	4
	s_{32}	Austria	RSA-4096	4

The approach for checking feasibility will be illustrated with an example. Figure 5.4 shows the sample process with compliance requirements as explained in Section 3.4. Service classes S_2 and S_3 are each subject to two local constraints, i. e. the location must be Germany and the provided data encryption must be RSA with a key length of at least 2048 bit. The subprocess is subject to a time limit of 15 seconds. In case that this time limit is not respected, a timeout message is sent to the user (the corresponding message event is omitted here) and the process ends. Table 5.4 shows the service alternatives which were introduced in Section 3.2.1. Here, the QoS properties price and availability have been omitted. Instead, for each service alternative information is given regarding the location where the service is hosted and the encryption strength provided.

Given such a process model and a set of service alternatives, the proposed feasibility checking method works as follows:

1. Iterate all service classes and delete all service alternatives which do not fulfill local compliance constraints.
2. If any service class is empty, no compliant solution exists; abort!
3. Iterate all scopes subject to a time limit and select from each service class in the scope the service alternative with minimum response time. Aggregate the runtimes.
4. If any time limit constraint is violated, no compliant solution exists; abort!

Algorithm 5.2 shows the pseudocode for this operation. Applying the first step to the service alternatives in Table 5.4 yields the reduced sets of service alternatives as shown in Table 5.5. Since none of the service classes is empty, step 3 is applied.

Algorithm 5.2: Feasibility Check(P, SA)

```
foreach $S_i \in SA$ do
    C := data and location constraints for $S_i$ in $P$;
    foreach $s_{ij} \in S_i$ do
        if $s_{ij}$ does not meet $C$ then remove $s_{ij}$ from $S_i$;
    end
    if $|S_i| = 0$ then return $false$;
end
foreach $scope \in timeScopes$ do
    $comp$ := Partial composition for $scope$ consisting of services with minimal
      response time;
    if $comp.responseTime > scope.timeLimit$ then return $false$;
end
return $true$;
```

As only one time limit including all service classes needs to be considered, the aggregation expression for response time which was determined in Section

Table 5.5: *Reduced service classes*

Service Class	Service Candidate	Location	Data Encryption	Response Time
S_1	s_{11}	Germany	RSA-1024	7
	s_{12}	Ireland	RSA-1024	4
	s_{13}	Austria	RSA-2048	5
S_2	s_{22}	Germany	RSA-3072	2
S_3	s_{31}	Germany	RSA-2028	4

3.2.2 can be reused, i. e. $\mathcal{Q}^{\alpha} = q_{1j}^{\alpha} + max(q_{2j}^{\alpha}, q_{3j}^{\alpha})$. From each service class, the service alternative with minimum response time is selected. This yields the solution $\{s_{12}, s_{22}, s_{31}\}$ which has an aggregated response time of 8 sec. Since this is lower than the time limit defined for the subprocess, this solution is compliant and hence optimization is feasible. In particular, it can be concluded that a solution might exist fulfilling both compliance and user requirements. However, the optimal solution with respect to user requirements determined in Section 3.2.1 was $\{s_{12}, s_{24}, s_{32}\}$. Since this was also the only solution fulfilling the user requirements, this shows that for the problem instance at hand, no solution exists which fulfills compliance and user requirements at the same time. In general this illustrates that the existence of a compliant solution does not guarantee the existence of a solution fulfilling compliance and user requirements simultaneously. Hence a successful feasibility check does not spare performing optimization.

5.3.3 Detecting Compliance Violations

This step of COMPAGA is performed firstly on the initial population and afterwards on each new population of the current iteration. In order to measure the degree of compliance of single service compositions, the notion of compliance distance is utilized which was first introduced by SADIQ et al. [SGN07]. Compliance distance is a quantitative measure and in its most basic form simply counts the number of compliance violations in a process instance. Here it is adapted for service compositions and employed to count how many assigned services in a composition violate compliance requirements. Of course this is a very basic view on compliance which is based on the assumption that each violation is equally bad. As was already discussed by SADIQ et al., a more sophisticated approach would be to associate a cost with each violation [SGN07]. Compliance distance would then be the sum of all violation costs of a service composition. However, for the sake of simplicity and without any loss in generality, the basic measure will be used in the following.

Determining **Data** and **Location** violations can be performed locally. Time scopes however can be nested and may be composed of human as well as non-human tasks (cf. Section 3.4). Therefore let *timeScopes* be a data structure which contains one list *scope* per time scope in a process model as well as its *time limit*. Given a process model P and a service composition SC, the compliance distance of SC can then be determined with a two step approach. First, all tasks in P are iterated and it is checked if the services in SC fulfill **data** and **location** requirements. If this is not the case, the indices of these tasks are stored in a list V. In case that a task $p \in P$ is part of one or several nested time scopes, the index of p is stored in all corresponding lists *scope* ∈ *timeScopes*. In a second step, all lists *scope* ∈ *timeScopes* are iterated and all times allotted to human tasks as well as all response times of services in *SC* assigned to non-human tasks are summed up. If the sum of these times is greater than the *time limit* assigned to *scope*, all elements of $p \in scope\ .\ p \notin V \wedge \neg p.isHumanTask$ are added to *V*. The compliance distance of *SC* can then be obtained by counting the elements in *V*. Algorithm 5.3 shows the pseudo-code for this operation.

Algorithm 5.3: Detect Compliance Violations(*P*, *SC*)

```
V := empty list;
timeScopes := data structure with one list per time scope and its time limit;
foreach $p \in P$ do
    C := p.getConstraints;
    foreach $c \in C$ do
        if $c.isDataAnnotation \vee c.isLocationAnnotation$ then
            Check if SC[p] meets compliance requirement c;
            if SC[p] does not meet c then add p to V;
        else if c.isTimeConstraint then
            Add p to all corresponding lists in timeScopes;
        end
    end
end
foreach $scope \in timeScopes$ do
    if $\Sigma$ human processing times + $\Sigma$ response times > scope.timeLimit then
        foreach $p \in scope\ .\ p \notin V \wedge \neg p.isHumanTask$ do
            Add p to V;
        end
    end
end
return V;
```

5.3.4 Recovering Compliance

This step is performed in each iteration of COMPAGA at a probability of p_{rep} for each population member which has a compliance distance > 0. In order to replace services in case of violations efficiently with suiting services, at first a clustering is performed on the set of service alternatives with a total of three levels. On the first level, services are clustered according to their service class S_i. On the second level, services are clustered according to their **data** class. Finally, on the third level services are clustered according to their **location** (cf. Figure 5.5)[65]. This clustering allows to find suiting service alternatives in logarithmic time.

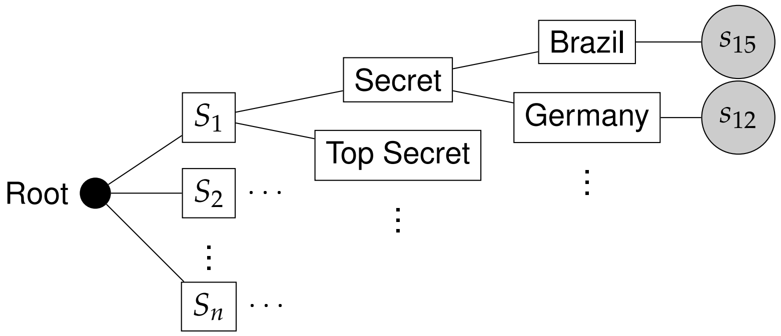


Figure 5.5: *Service clustering*

Given such a clustering, a process model P, a list of violations V as produced by algorithm 5.3 and a non-compliant service composition SC, the repair operation works as follows. The list V is iterated and for each violation $v \in V$ at first the service class S_v of v is determined as well as the set C of **data** and **location** constraints which are defined in P. Then the set of clustered services is searched for a service candidate $s_{vj} \in S_v$, which has

1. at least the **data** class as defined in P,
2. the same **location** as defined in P and
3. minimum response time.

If such a candidate exists, the service of class S_v in SC is replaced with it. Algorithm 5.4 shows the pseudo code for this operation. It should be noted that the clustering is performed right after a positive feasibility check and has been placed in algorithm 5.4 only for the sake of clarity.

Obviously, the success of this operation depends on the question if suiting service alternatives are available. Therefore it cannot be guaranteed that a non-compliant SC will become compliant after performing this repair operation. Another noteworthy aspect is that **time limit** constraints are not explicitly addressed, but implicitly by picking services with minimum response time. The reason for this decision was that checking time limits would again require a two step approach as in algorithm 5.3. However, analysis revealed that the proposed technique will result in a compliant service composition in terms of **time limits** if suiting service

[65] The second and the third level are interchangeable, i. e. one could also cluster services for **location** before **data** class.

Algorithm 5.4: Repair operation(P, SC, V)

```
Clustering := Three-level clustering of service alternatives;
foreach $v \in V$ do
    $S_v$ := service class of $v$;
    $C$ := data and location constraints for $S_v$ in $P$;
    $cand := Clustering.pick(s \in S_v \; . \; s.data \geqslant C.data \wedge s.location = C.location$
                          $\wedge\ s.responeTime = min)$;
    if $cand \neq \emptyset$ then
        $SC[S_v] := cand$;
    end
end
return $SC$;
```

alternatives exist. If this is not the case, **time limit** constraints cannot be met without violating another local constraint. Therefore a more sophisticated approach is not necessary. Since COMPAGA is a GA, it can find at most a number of elements in $\mathcal{PF}^*$ equal to the population size (cf. Section 2.5.6).

5.4 Summary

This chapter presented the two main contributions of this thesis. The first one is a method to assess the security of service compositions based on the notion of structural decomposition. Extensions to the basic model have been presented in order to consider interdependence effects between security objectives. However, the method is limited as not all structures can be captured in case of implication relationships. The second contribution is a drop-in solution for considering compliance aspects in service selection with GAs. An approach was presented which builds on top of compliance distance in order to detect and repair compliance violations in service compositions.

Chapter 6

Proof of Concept

> *"The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures."*
>
> – Frederick P. Brooks, Jr. [Bro95, p. 7]

In this chapter the Service Selection Workbench (SSW) is presented. This tool aims at illustrating the approach presented in Chapter 5. First an overview is provided of the tool's functionality (Section 6.1). Next a round-trip is given in using the tool from workspace creation to performing simulations (Section 6.2). This round-trip is followed by a discussion of the implemented features with respect to the requirements identified in Section 4.1 (Section 6.3). A summary of the main results concludes this chapter (Section 6.4).

6.1 Overview

SSW is a tool which has been developed in order to support users in building security assessment models according to the approach presented in Section 5.2. Furthermore, a simulation environment has been integrated to analyze resulting MOO problems with different GAs (including COMPAGA presented in Section 5.3) and different simulation settings. SSW basically consists of three components:

- QoS model editor
- Domain model editor
- Simulation environment

SSW has been developed on WINDOWS using Java. It uses Swing for all tasks related to the Graphical User Interface (GUI) and the jMetal framework [DN11] for simulations. The philosophy behind SSW is that it shall at first hand illustrate the approach presented in Chapter 5 by means of a real system for users with a variety of backgrounds. The way users interact with SSW, especially the models they build, tells more about the practicability of the approach than any theoretical analysis could. Another aspect is that SSW might be used as a building block to develop a plugin or a completely new system for process modeling using the approaches presented in Chapter 5. The following subsections give a brief overview of each component.

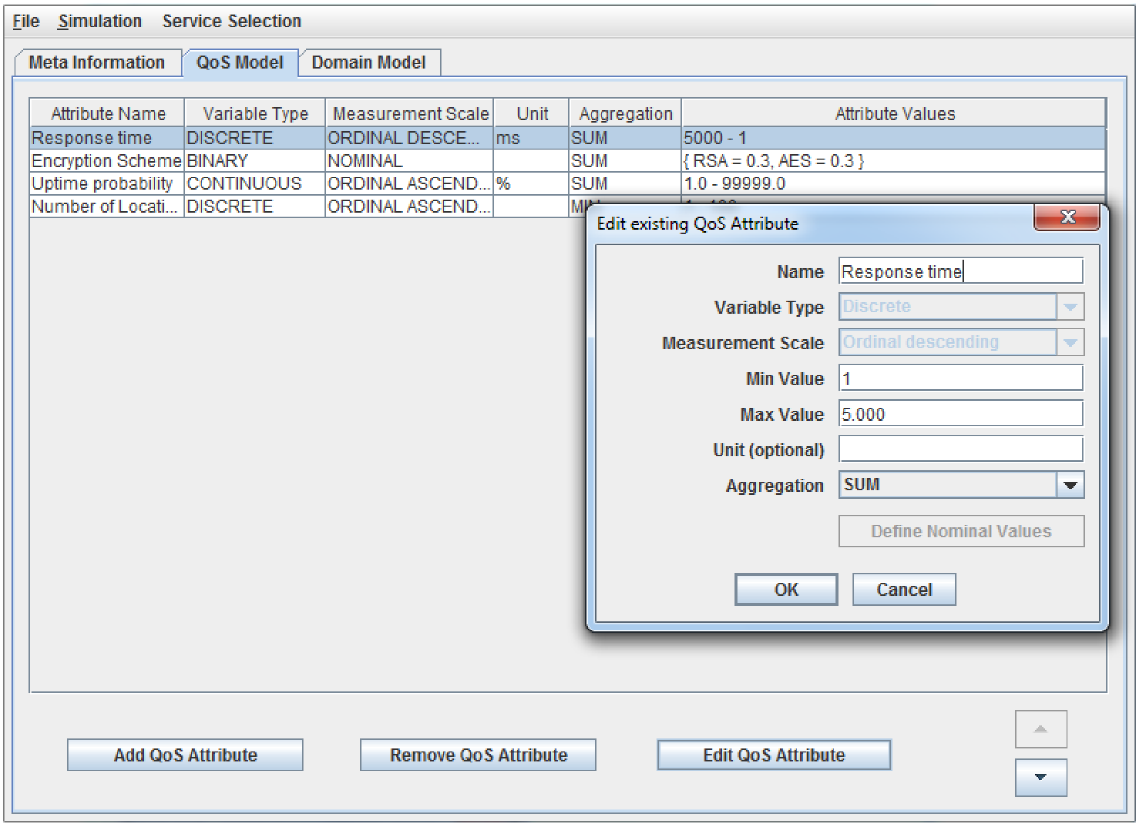
6.1.1 QoS Model Editor

This component allows users to define QoS models of any complexity (cf. Figure 6.1). It allows to add, delete and edit existing QoS attributes. For each QoS attribute the user needs to specify

- a **name**
- its **variable type**
- its **measurement scale**
- **min** and **max** values
- an optional **unit**
- its **aggregation scheme**[66]

These properties have been already explained in Sections 3.2.4 and 5.2.1 respectively. What should be additionally explained here is the implemented approach for defining nominal attribute values. As can be seen in Figure 6.1, the menu for defining new QoS attributes contains a button labeled "Define Nominal Values". In case of QoS attributes with a nominal scale, this button opens up a new dialog which enables users to specify attribute values (e. g. different encryption algorithms) and to assign a utility value to each entry (cf. Figure 6.2).

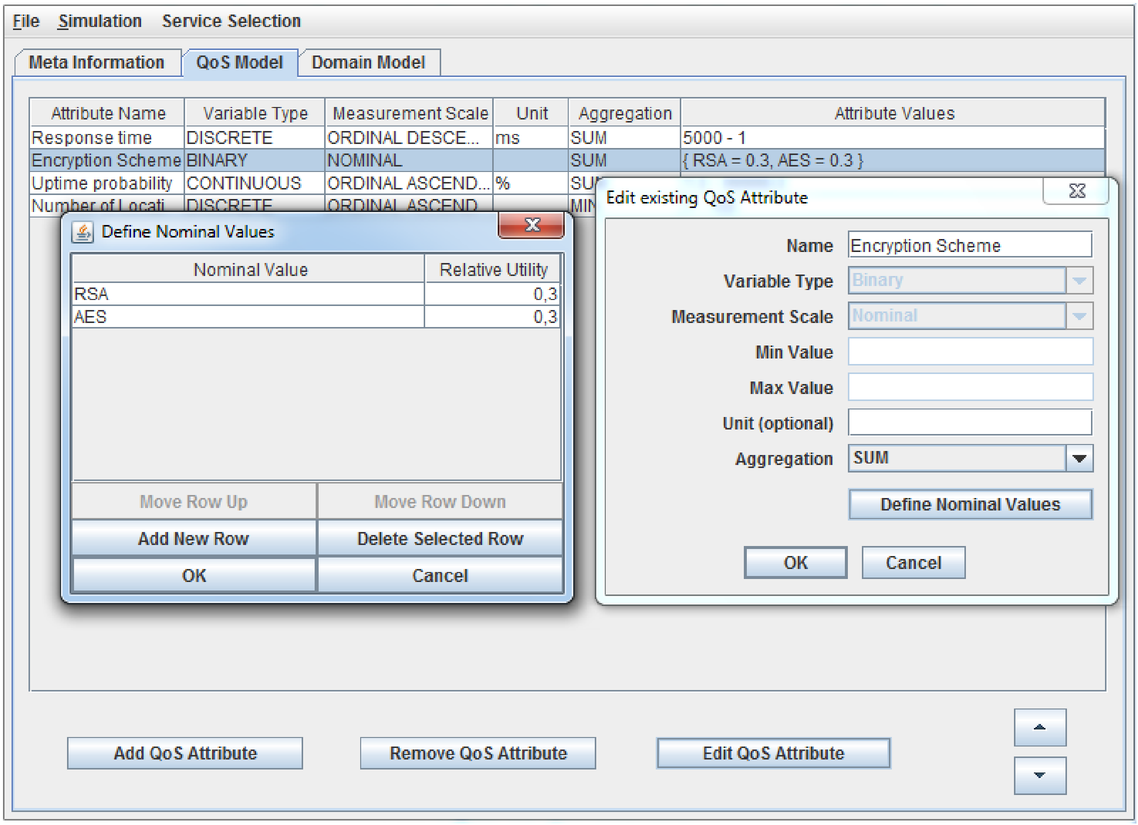
[66] For the sake of simplicity the prototype is restricted to sequential workflows. Since however a combinatorial approach is employed for problem modeling, this does not cause any loss of generality (cf. Section 3.2.4).



Attribute Name	Variable Type	Measurement Scale	Unit	Aggregation	Attribute Values
Response time	DISCRETE	ORDINAL DESCE...	ms	SUM	5000 - 1
Encryption Scheme	BINARY	NOMINAL		SUM	{ RSA = 0.3, AES = 0.3 }
Uptime probability	CONTINUOUS	ORDINAL ASCEND...	%	SUM	1.0 - 99999.0
Number of Locati...	DISCRETE	ORDINAL ASCEND...		MI	1 - 100

Name: Response time
Variable Type: Discrete
Measurement Scale: Ordinal descending
Min Value: 1
Max Value: 5.000
Unit (optional):
Aggregation: SUM

Figure 6.1: *QoS model editor*



Attribute Name	Variable Type	Measurement Scale	Unit	Aggregation	Attribute Values
Response time	DISCRETE	ORDINAL DESCE...	ms	SUM	5000 - 1
Encryption Scheme	BINARY	NOMINAL		SUM	{ RSA = 0.3, AES = 0.3 }
Uptime probability	CONTINUOUS	ORDINAL ASCEND...	%	SU	
Number of Locati...	DISCRETE	ORDINAL ASCEND...		MIN	

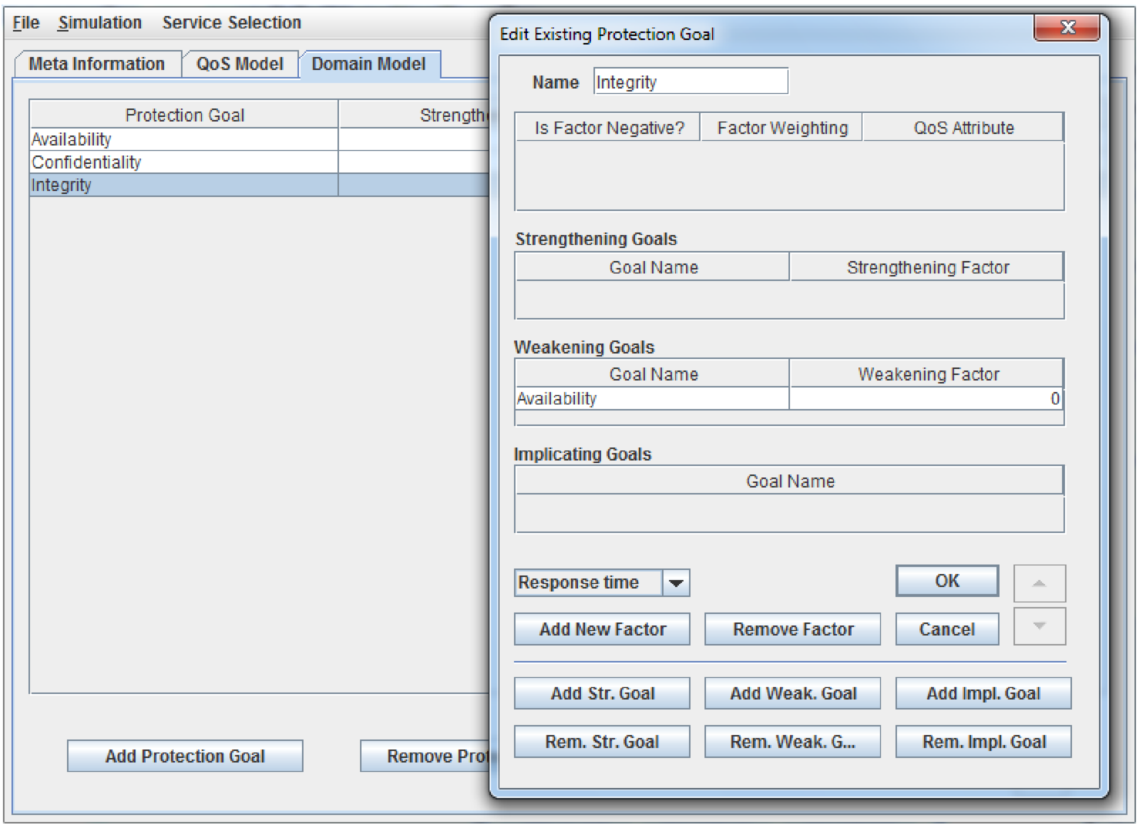
Nominal Value	Relative Utility
RSA	0,3
AES	0,3

Name: Encryption Scheme
Variable Type: Binary
Measurement Scale: Nominal
Min Value:
Max Value:
Unit (optional):
Aggregation: SUM

Figure 6.2: *Defining utility values for nominal QoS attributes*

6.1.2 Domain Model Editor

The domain model editor (cf. Figure 6.3) allows to define protection functions as well as their interdependencies as explained in Section 5.2.2.3. For each protection function users can specify a name and select QoS attributes which have an impact on that particular protection function. The corresponding list only shows QoS attributes which have already been defined. For each chosen QoS attribute users are allowed to define weightings and to specify if this attribute is a negative factor. Furthermore, the editor allows to add existing protection functions as either strengthening, weakening or implicating objectives. Implication relationships are validated according to the method presented in Section 5.2.2.3. In case that the given model does not suffice the required conditions (i. e. attribute chaining and pessimistic factor weighting), an error screen shows up to inform the user.



Protection Goal	Strength
Availability	
Confidentiality	
Integrity	

Name: Integrity

Is Factor Negative?	Factor Weighting	QoS Attribute

Strengthening Goals

Goal Name	Strengthening Factor

Weakening Goals

Goal Name	Weakening Factor
Availability	0

Implicating Goals

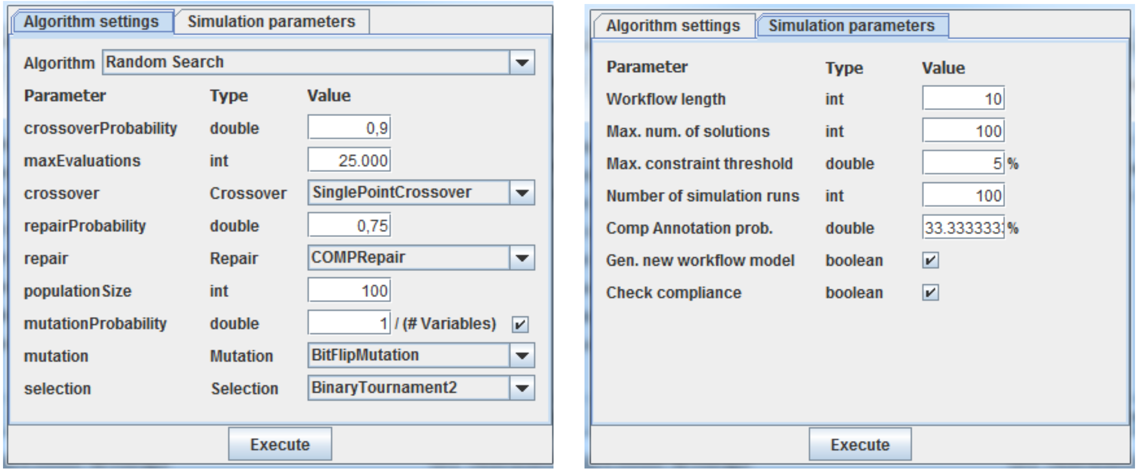
Goal Name

Response time

Figure 6.3: *Domain model editor*

6.1.3 Simulation Environment

The simulation environment consists of two tabs which allow for defining algorithm as well as simulation settings (cf. Figure 6.4). The tab "Algorithm settings" allows users to specify not only the algorithm to be used but also GA-related options such as selecting GA operators as well as their application probability, population size, etc. An exhaustive enumeration of all options is beyond the scope of this thesis. It should be noted however that depending on the GA employed, some settings might stay without effect if the respective algorithm does not support them. For instance, the random approach (cf. Section 7.2) does not employ any kind of crossover, mutation or repair operator. As such changing the types of these operators as well as their probabilities will stay without any effect[67]. This is particularly the case for the repair operation COMPRepair which is only supported by COMPAGA. Another noteworthy fact is that the layout of the tab "algorithm settings" has been strongly influenced by a support GUI for the jMetal library presented in [DN11]. Simulations are performed according to the "ideal multi-objective optimization procedure" proposed by Deb (cf. Section 2.5.4), i. e. not the complete Pareto front is determined but at most a predefined maximum of Pareto optimal solutions (cf. below).



Algorithm: Random Search

Parameter	Type	Value
crossoverProbability	double	0,9
maxEvaluations	int	25.000
crossover	Crossover	SinglePointCrossover
repairProbability	double	0,75
repair	Repair	COMPRepair
populationSize	int	100
mutationProbability	double	1 / (# Variables) ☑
mutation	Mutation	BitFlipMutation
selection	Selection	BinaryTournament2

(a) Algorithm settings

Parameter	Type	Value
Workflow length	int	10
Max. num. of solutions	int	100
Max. constraint threshold	double	5 %
Number of simulation runs	int	100
Comp Annotation prob.	double	33.333333 %
Gen. new workflow model	boolean	☑
Check compliance	boolean	☑

(b) Simulation parameters

Figure 6.4: *Simulation environment*

The tab "Simulation parameters" is employed to prepare simulations such as the ones reported in Chapter 7. It allows for defining

- workflow length
- maximum number of solutions to be determined
- the maximum constraint threshold to be accepted
- the number of simulation runs to be performed

[67] For details about the respective GAs the interested reader is referred to Section 7.2 as well as to the source code of jMetal which can be obtained at: `http://jmetal.sourceforge.net/`

- the probability at which single tasks in the generated workflow are decorated with random compliance requirements

Furthermore, the user can decide to either generate a completely new workflow model at each simulation run or to reuse a previously generated one. This option allows to compare simulation results against each other and has been extensively used when preparing the simulation results reported in Chapter 7.

Finally, the user can either turn on or off compliance checking of workflow models. This approach is necessary because checking compliance in terms of location, data and time limit requirements necessitates the QoS model to contain certain attributes. Since SSW does not claim to be a mature, feature-rich product, the required QoS attributes have been hard-coded for the sake of simplicity. If these attributes are not present in the QoS model, compliance checking cannot be performed. Hence the state of this flag is used internally to determine if the QoS model shall be checked for these attributes or not. In case that compliance checking is turned on and the model does not support the required attributes, the user is prevented from performing any simulations with an error message.

In particular, performing compliance checks requires the employed QoS model to contain the following three attributes:

1. *Encryption* with variable type binary and measurement scale nominal containing the following nominal values: { RSA = 1.0, AES = 0.9, 3DES = 0.8, None = 0.0 },

2. *Hosting location* with variable type binary and measurement scale nominal containing the following nominal values: { Europe = 1.0, Asia = 0.9, USA = 0.8 },

3. *Response Time* with variable type discrete and measurement scale ordinal descending.

If all settings are plausible, simulations start after clicking the "Execute" button. Results of simulation runs are written to several files (cf. Figure 6.5). The central file for each simulation run is `_SIMULATION.LOG` which is a log generated by the Java class `Logger`[68]. This file documents all steps taken and results generated during simulations along with timestamps and can thus be used to e. g. reconstruct all experiments in detail as well as to detect and pinpoint abnormal program

[68] Cf. the Java documentation for details: `http://docs.oracle.com/javase/6/docs/api/java/util/logging/Logger.html`

behavior. In order to ease numerical evaluations, all generated results are again written to individual files which can be separately imported to tools such as SPSS, MICROSOFT EXCEL and OPENOFFICE CALC. Some of the abbreviations used in Figure 6.5 refer to auxiliary metrics employed for the numerical evaluations performed in the scope of this thesis and are covered in Section 7.3.

`_SIMULATION.LOG`	Main log file of each simulation run
General simulation results	
`_ADF.TXT`	Fulfillment of user requirements
`_HV.TXT`	Hypervolume of generated solutions
`_NORM_L2.TXT`	Euclidian or L2 norm of function values
`_NORM_RMS.TXT`	RMS norm of function values
`_PRICE.TXT`	Price of generated solutions
`_RUNTIME.TXT`	Runtime of employed algorithm
Results for comparing COMPAGA with other GAs	
`_CD.TXT`	Compliance distance of solutions
`_ENCRYPTION.TXT`	Average encryption strength of solutions
`_RESPONSETIME.TXT`	Aggregated response time of solutions

Figure 6.5: *Files generated by SSW during simulations*

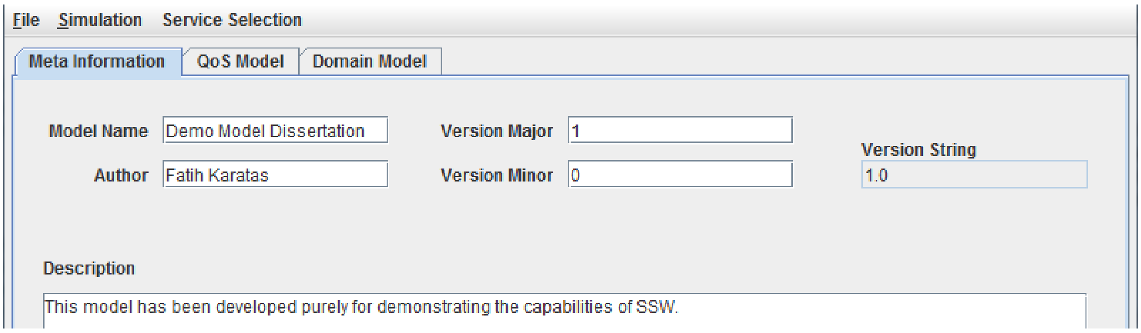
6.2 Application Example

This section briefly describes some typical use cases of SSW. Particularly, the aim is to give the reader a glimpse on the possibilities offered by SSW although its still an early step towards a mature product.

6.2.1 Workspace Creation

A workspace in SSW is a container for security assessment models as well as test data for simulations. Creating a workspace is performed over a self describing menu entry. After creation, users can provide optional meta information about the model which solely serves for description purposes (cf. Figure 6.6). As soon as the workspace is created, users can build QoS and domain models as described in Section 6.1.1 and 6.1.2 respectively.

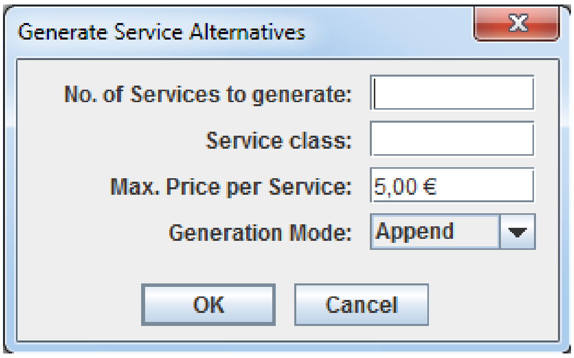
In order to perform simulations however, it is necessary to generate test data, i. e. service alternatives. For this task, SSW provides a separate dialog. After providing the service class and the maximum price per service, the required number of service alternatives are generated (cf. Figure 6.7). QoS attribute values



Model Name: Demo Model Dissertation
Author: Fatih Karatas
Version Major: 1
Version Minor: 0
Version String: 1.0
Description: This model has been developed purely for demonstrating the capabilities of SSW.

Figure 6.6: *Providing optional meta information for workspaces*

of each alternative are chosen randomly from the given range for QoS attributes employing either ordinal, interval or ratio scales. Similarly, price is randomly picked from 0 to maximum price which is 5.00 by default. In case of nominal QoS attributes, one of the defined elements is randomly picked. Effectivity vectors are also generated randomly for each service alternative. The generated service alternatives can be viewed and edited afterwards in a separate dialog (cf. Figure 6.8).



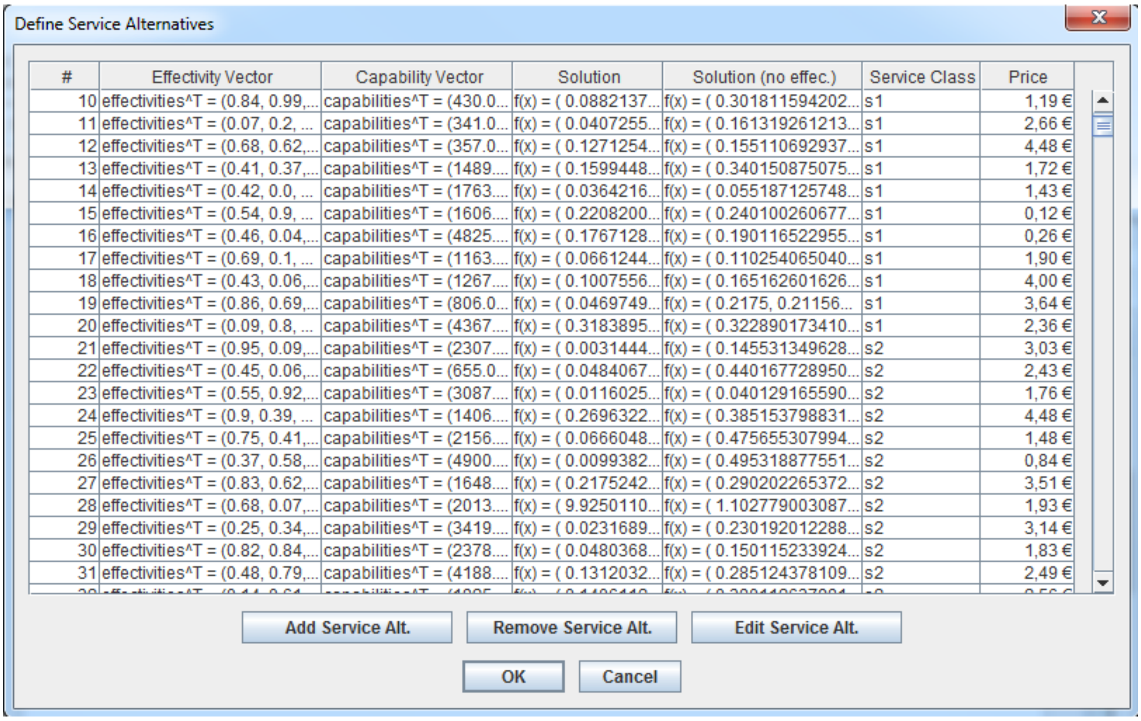
No. of Services to generate:
Service class:
Max. Price per Service: 5,00 €
Generation Mode: Append

Figure 6.7: *Generating random service alternatives for simulations*

6.2.2 Expressing Requirements and Performing Simulations

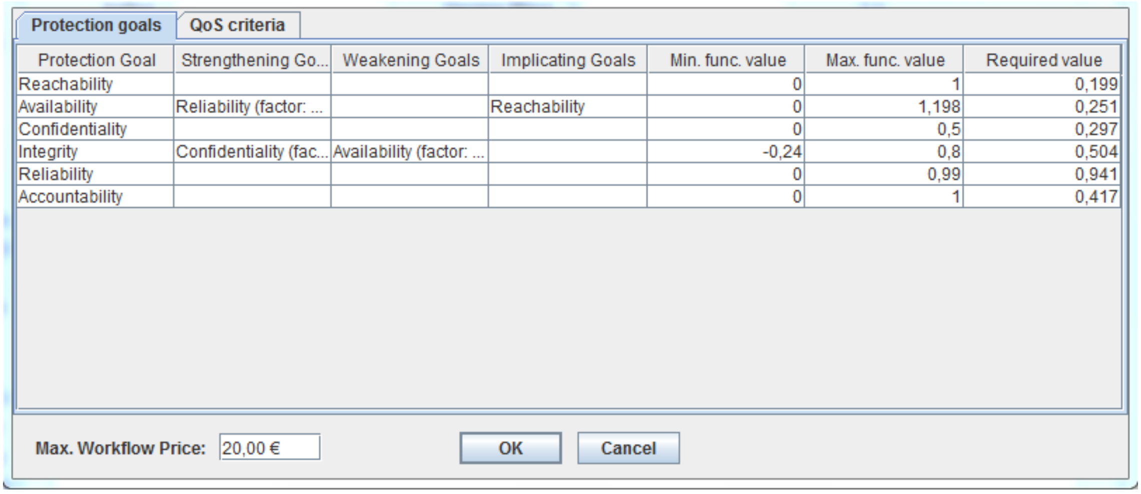
SSW supports expressing user requirements at the level of single protection functions and QoS attributes via a separate dialog (cf. Figure 6.9). Requirements can be specified either manually or generated randomly. These requirements are subsequently considered during service selection by means of constraints.

When performing simulations, users can choose between two different modes. The simulation environment presented in Section 6.1.3 considers service selection as MOO problem without taking any preferences into account with regards to security objectives. Furthermore, SSW provides users with a different mode which allows to express preferences regarding single protection functions in order to reduce the MOO problem to a SOO problem (cf. Section 2.5.4). By default, no pref-



#	Effectivity Vector	Capability Vector	Solution	Solution (no effec.)	Service Class	Price
10	effectivities^T = (0.84, 0.99,...	capabilities^T = (430.0...	f(x) = (0.0882137...	f(x) = (0.301811594202...	s1	1,19 €
11	effectivities^T = (0.07, 0.2, ...	capabilities^T = (341.0...	f(x) = (0.0407255...	f(x) = (0.161319261213...	s1	2,66 €
12	effectivities^T = (0.68, 0.62,...	capabilities^T = (357.0...	f(x) = (0.1271254...	f(x) = (0.155110692937...	s1	4,48 €
13	effectivities^T = (0.41, 0.37,...	capabilities^T = (1489....	f(x) = (0.1599448...	f(x) = (0.340150875075...	s1	1,72 €
14	effectivities^T = (0.42, 0.0, ...	capabilities^T = (1763....	f(x) = (0.0364216...	f(x) = (0.055187125748...	s1	1,43 €
15	effectivities^T = (0.54, 0.9, ...	capabilities^T = (1606....	f(x) = (0.2208200...	f(x) = (0.240100260677...	s1	0,12 €
16	effectivities^T = (0.46, 0.04,...	capabilities^T = (4825....	f(x) = (0.1767128...	f(x) = (0.190116522955...	s1	0,26 €
17	effectivities^T = (0.69, 0.1, ...	capabilities^T = (1163....	f(x) = (0.0661244...	f(x) = (0.110254065040...	s1	1,90 €
18	effectivities^T = (0.43, 0.06,...	capabilities^T = (1267....	f(x) = (0.1007556...	f(x) = (0.165162601626...	s1	4,00 €
19	effectivities^T = (0.86, 0.69,...	capabilities^T = (806.0...	f(x) = (0.0469749...	f(x) = (0.2175, 0.21156...	s1	3,64 €
20	effectivities^T = (0.09, 0.8, ...	capabilities^T = (4367....	f(x) = (0.3183895...	f(x) = (0.322890173410...	s1	2,36 €
21	effectivities^T = (0.95, 0.09,...	capabilities^T = (2307....	f(x) = (0.0031444...	f(x) = (0.145531349628...	s2	3,03 €
22	effectivities^T = (0.45, 0.06,...	capabilities^T = (655.0...	f(x) = (0.0484067...	f(x) = (0.440167728950...	s2	2,43 €
23	effectivities^T = (0.55, 0.92,...	capabilities^T = (3087....	f(x) = (0.0116025...	f(x) = (0.040129165590...	s2	1,76 €
24	effectivities^T = (0.9, 0.39, ...	capabilities^T = (1406....	f(x) = (0.2696322...	f(x) = (0.385153798831...	s2	4,48 €
25	effectivities^T = (0.75, 0.41,...	capabilities^T = (2156....	f(x) = (0.0666048...	f(x) = (0.475655307994...	s2	1,48 €
26	effectivities^T = (0.37, 0.58,...	capabilities^T = (4900....	f(x) = (0.0099382...	f(x) = (0.495318877551...	s2	0,84 €
27	effectivities^T = (0.83, 0.62,...	capabilities^T = (1648....	f(x) = (0.2175242...	f(x) = (0.290202265372...	s2	3,51 €
28	effectivities^T = (0.68, 0.07,...	capabilities^T = (2013....	f(x) = (9.9250110...	f(x) = (1.102779003087...	s2	1,93 €
29	effectivities^T = (0.25, 0.34,...	capabilities^T = (3419....	f(x) = (0.0231689...	f(x) = (0.230192012288...	s2	3,14 €
30	effectivities^T = (0.82, 0.84,...	capabilities^T = (2378....	f(x) = (0.0480368...	f(x) = (0.150115233924...	s2	1,83 €
31	effectivities^T = (0.48, 0.79,...	capabilities^T = (4188....	f(x) = (0.1312032...	f(x) = (0.285124378109...	s2	2,49 €

Figure 6.8: *Editing generated service alternatives*

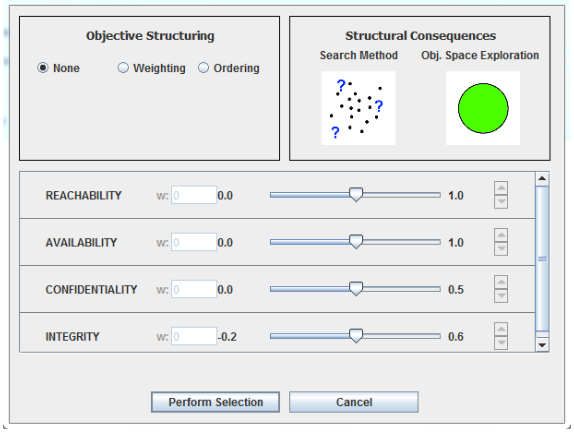


Protection Goal	Strengthening Go...	Weakening Goals	Implicating Goals	Min. func. value	Max. func. value	Required value
Reachability				0	1	0,199
Availability	Reliability (factor: ...		Reachability	0	1,198	0,251
Confidentiality				0	0,5	0,297
Integrity	Confidentiality (fac...	Availability (factor: ...		-0,24	0,8	0,504
Reliability				0	0,99	0,941
Accountability				0	1	0,417

Max. Workflow Price: 20,00 €

Figure 6.9: *Dialog for defining user requirements*

erences for any objective are given (cf. Figure 6.10). Two icons in the upper right corner symbolize that in this case the search space is being thoroughly explored at the cost of exactness. The other supported modes are objective weighting (cf. Figure 6.11(a)) and lexicographic ordering of objectives (cf. Figure 6.11(b)). In case of objective weighting, the user needs to define weights for each objective such that the sum must be 1. In case of lexicographic ordering, the user has to arrange the objectives with small arrow buttons at the right of each objective. Each objective is represented with a different coloring. The order in which objectives are evaluated goes from green to red, i. e. the green objective is evaluated first and the red objective last. In both cases again two little icons in the upper right corner inform the user that the problem can be solved exactly at the cost of restricted search space exploration. The sliders which can be seen next to each security objective allow for expressing requirements with regards to that particular objective. Upper and lower boundaries of each protection function are determined by inserting maximum and minimum values respectively for each e_{ij} and q'_{ij}.



Objective Structuring

● None ○ Weighting ○ Ordering

Structural Consequences

Search Method Obj. Space Exploration

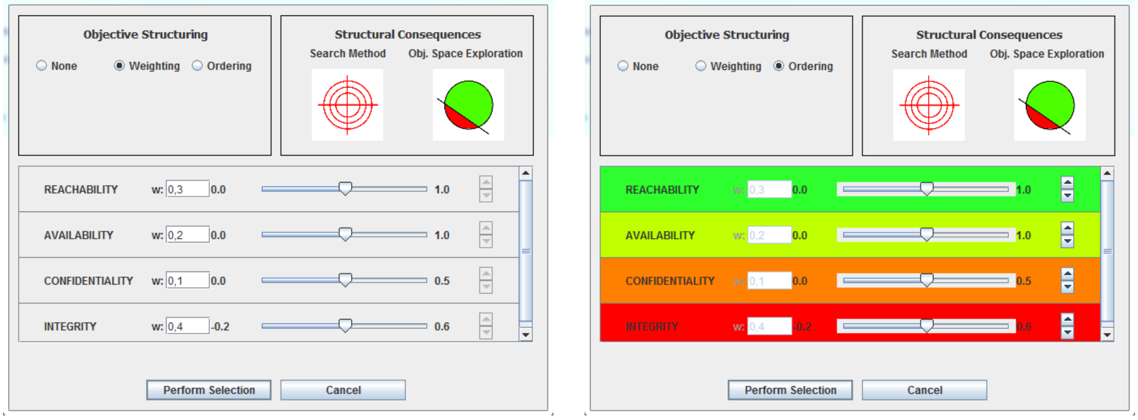
REACHABILITY w: 0 0.0 1.0

AVAILABILITY w: 0 0.0 1.0

CONFIDENTIALITY w: 0 0.0 0.5

INTEGRITY w: 0 -0.2 0.6

Figure 6.10: *Service selection with no preferences regarding objectives*



Objective Structuring

○ None ● Weighting ○ Ordering

Structural Consequences

Search Method Obj. Space Exploration

REACHABILITY w: 0,3 0.0 1.0

AVAILABILITY w: 0,2 0.0 1.0

CONFIDENTIALITY w: 0,1 0.0 0.5

INTEGRITY w: 0,4 -0.2 0.6

Objective Structuring

○ None ○ Weighting ● Ordering

Structural Consequences

Search Method Obj. Space Exploration

REACHABILITY w: 0,3 0.0 1.0

AVAILABILITY w: 0,2 0.0 1.0

CONFIDENTIALITY w: 0,1 0.0 0.5

INTEGRITY w: 0,4 -0.2 0.6

(a) Objective Weighting (b) Lexicographic Ordering

Figure 6.11: *Different methods for expressing preferences regarding objectives*

6.3 Discussion

As can be seen, SSW implements the approaches developed in Chapter 5 and hence fulfills its purpose of being a prototypical implementation for the requirements discussed in Chapter 4. In order to become a fully featured BPMS however, SSW would require facilities for the phases design, process enactment as well as diagnosis of the BPM lifecycle (cf. Section 2.1.1). These steps remain as open tasks for future version of SSW.

Two remarks shall be added here regarding the requirements formulated in Chapter 4. The first remark is about requirement **R1 (Completeness)**. Since SSW does not yet either support importing process models or modeling processes, all process models are assumed to be sequential. A methodical consequence is that currently only one aggregation scheme per QoS attribute is supported. Extending SSW to support multiple aggregation schemes per QoS attribute is a minor task from the author's point of view. The amount of time however which would be necessary to develop a modeling component or a feature for importing process models was considered too much for the given timeframe. Hence this feature has been skipped since without proper process models it would not yield any added value.

The other remark is about the fulfillment of requirement **R3 (Thorough Search Space Exploration)**. In its current state, SSW does not implement solving SOO problems since this is a common feature which does not yield any added value in the scope of this thesis. However, several Java libraries are available for this task such as Apache Commons Math[69] and CPLEX Optimizer by IBM[70] which provide implementations for ILP solving algorithms such as Simplex. From the authors point of view, integrating SSW with these libraries is a straightforward but nevertheless time consuming task and hence has been skipped in the scope of this thesis.

6.4 Summary

This chapter presented SSW, a proof of concept implementation for the approaches introduced and discussed in Chapter 5. It was shown that SSW prototypically fulfills the requirements identified in Section 4.1. Currently, SSW has two minor limitations. Firstly, it supports only sequential workflows and hence only one aggregation scheme per QoS attribute due to a lack of support for real process models. The other limitation is that in its current state SSW only supports determining Pareto optimal solutions for MOO problems employing GAs up to a user defined maximum (cf. Section 6.1.3). Solving SOO problems and completely determining the Pareto front have been skipped in order to focus on features that yield added value in the scope of this thesis. Since the source code of SSW is however freely available[71], it is to be hoped that these missing features might be

[69] http://commons.apache.org/proper/commons-math/

[70] http://www.ibm.com/support/entry/portal/product/websphere/ibm_ilog_cplex_optimization_studio?productContext=-568455760

[71] http://www.fatih-karatas.com/projects

added in the future.

Part III

Evaluation

Chapter 7

Numerical Results

> *"If you can not measure it, you can not improve it."*
>
> – Lord Kelvin

THIS CHAPTER PROVIDES an overview of the simulations conducted in order to study the performance of COMPAGA. At first an overview is given about the experimental setup (Section 7.1), employed reference algorithms (Section 7.2) and some auxiliary metrics utilized in the simulations (Section 7.3). Afterwards the question of how to choose a meaningful p_{rep} is addressed (Section 7.4) before reporting simulation results (Section 7.5). The chapter is concluded by a brief summary (Section 7.6).

7.1 Experimental Setup

All experiments reported in this chapter were performed on a machine with a 2.67 GHz Intel Core i5 CPU, 2 GB RAM and running WINDOWS 7 (32 Bit). The simulation environment is written in Java 1.6 and utilizes the jMetal 4.0 framework [DN11]. The reference algorithms selected for comparisons (cf. Section 7.2) are shipping with the framework and were not modified in any way.

Each algorithm was tested with identical simulation settings. The population size for each algorithm was always 100. Workflow length has been varied from 10 to 50 with a stepping of 10 and for each task a total of 20 service alternatives were generated. Each test case was evaluated 100 times and each algorithm was allowed to perform at most 25,000 evaluations. For each test case a workflow model and service alternatives were generated randomly. Workflow models were randomly decorated with compliance requirements. For the sake of simplicity, generated workflows are always sequential and contain at most one time limit constraint. Since a combinatorial model has been employed (cf. Section 3.2.4) this does not lead to any loss of generality. Each task is decorated with a data or location constraint at a uniform probability which is $33.\overline{3}\%$ by default (cf. Section 6.1.3).

For all simulations the same QoS model has been utilized containing a total of eight attributes (cf. Table 7.1). As explained in Section 6.2.1, each service alternative was generated with a random value in the given interval for each QoS attribute an with a random price in the interval $[0.00, 5.00]$. Based on this QoS model, a total of three domain models were generated with 2, 4 and 6 protection functions respectively. It should be noted that the QoS model as well as the domain models are not based on any inquiries but use to some extent arbitrary values and interdependencies in order to study the properties of COMPAGA unbiased and in a general fashion.

In order to determine confidence intervals, sample variance s has been employed since the actual variance σ is unknown. The sample variance is defined as follows:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{7.1}$$

where $n \neq 1$ is the sample size, x_i the i-th sample value and $\bar{x}$ the sample's mean. A sample's confidence interval is then given by:

$$\left[\bar{x} - t_{(1-\frac{\alpha}{2};n-1)}\frac{s}{\sqrt{n}}; \bar{x} + t_{(1-\frac{\alpha}{2};n-1)}\frac{s}{\sqrt{n}}\right] \tag{7.2}$$

where $n \neq 0$ is the sample size, $t_{(1-\frac{\alpha}{2};n-1)}$ the $(1-\frac{\alpha}{2})$ quantile of Student's t-distribution with $n-1$ degrees of freedom. A confidence level of 0.95 was considered as sufficient. Considering 99 degrees of freedom (since every experiment was repeated 100 times), this leads to a quantile of 1.9842.

With this setup, a total of 15 test cases (5 different workflow lengths and 3 domain models) have been simulated for COMPAGA as well as the reference algorithms (cf. Section 7.2).

7.2 Reference Algorithms

A number of state-of-the-art MOO algorithms were selected which provide overall good results for most optimization problems. Besides that, a random approach was utilized to obtain an approximate baseline for the experiments. Since each algorithm operates on a certain population of solution candidates, the maximum

Table 7.1: *QoS model used for simulations*

Attribute	Variable	Measurement	Aggregation	Attribute
Name	Type	Scale	Scheme	Value
Response Time	Discrete	Ordinal Descending	SUM	$[1, 5000]$
Encryption	Binary	Nominal	MIN	$\{\text{RSA} = 1.0, \text{AES} = 0.9, \text{3DES} = 0.8\}$
Latency	Discrete	Ordinal Descending	SUM	$[1, 5000]$
Uptime Probability	Continuous	Ordinal Ascending	PRODUCT	$[0.01, 1.00]$
Hosting location	Binary	Nominal	MIN	$\{\text{Europe} = 1.0, \text{Asia} = 0.9, \text{USA} = 0.8\}$
Scalability	Continuous	Ordinal Ascending	PRODUCT	$[0.01, 1.00]$
Throughput	Discrete	Ordinal Ascending	MIN	$[1, 1000]$
Certificate	Binary	Nominal	MIN	$\{\text{X.509} = 1.0, \text{PGP} = 0.8\}$

number of elements in $\mathcal{PF}^*$ which can be discovered by any algorithm is determined by the population size (cf. Section 2.5.6). Hence no algorithm gives a general guarantee to determine $\mathcal{PF}^*$ completely. As already stated, all these algorithms ship with the jMetal framework and remained unmodified. In the following a brief overview is given about each algorithm.

Random Approach

This algorithm randomly assigns services to each task. As no evaluation of solutions or improvements are performed, the results of this algorithm form an approximate baseline which any sophisticated algorithm should outperform. Since this algorithm does not necessarily deliver feasible solutions, it is not a reference algorithm in a strict sense. Its only purpose is to give an impression of the limits of pure random service selection and of the benefits of sophisticated approaches.

IBEA

ZITZLER and KÜNZLI introduced the Indicator-Based Evolutionary Algorithm (IBEA) which employs a binary quality indicator to compare two potential solution sets. The fitness of individual solutions is determined by comparing them against the population employing this indicator [ZK04]. This effectively extends the dominance relation introduced in Section 2.5.4.

NSGA-II

DEB et al. presented the Nondominated Sorting Genetic Algorithm-II (NSGA-II) which employs a fast sorting algorithm to assign individuals to a front. These fronts are ranked such that a solution in a front dominates all solutions in a front with a lower rank and is only dominated by solutions in a front with a higher rank. As soon as all fronts are determined, NSGA-II employs a crowded comparison approach in order to preserve diversity among the population [DPA+02].

SPEA2

ZITZLER et al. introduced the Strength Pareto Evolutionary Algorithm 2 (SPEA2) which determines individual fitness based on the number of individuals it dominates as well as the number of individuals by which it is dominated. In order to preserve diversity in the population, nearest neighbor estimation is utilized. Furthermore, SPEA2 uses an archive to preserve obtained boundary solutions [ZLT02].

7.3 Auxiliary Metrics

Some auxiliary metrics have been utilized in the experiments in order to compare results. Following is an overview and a brief explanation of each metric.

Hypervolume Ratio

In Section 2.5.6 the $\mathcal{HV}$ was introduced as an indicator for measuring the quality of a population of solutions generated by a GA. In order to compare the results of different GAs for a given MOO problem, it would be however desirable to quantify the quality of obtained solutions in a normalized interval $[0, 1]$. This would on the one hand allow to rank algorithms in terms of efficiency for different MOO problems. On the other hand, it gives an idea of how well different GAs are able to explore the search space of a certain MOO problem. Therefore, an auxiliary metric is constructed as follows. First, the $\mathcal{HV}$s of all employed algorithms for a MOO problem are merged into a maximum $\mathcal{HV}$ which is denoted $\mathcal{HV}_{max}$. The quality of a single algorithm β can then be computed by means of the hypervolume ratio $\mathcal{HR}$ as follows [CLV07]:

$$\mathcal{HR}_{\beta} = \frac{\mathcal{HV}_{\beta}}{\mathcal{HV}_{max}} \tag{7.3}$$

where $\mathcal{HV}_{max} \neq 0$ such that $\sum \mathcal{HR}_{\beta} = 1$. In other words, $\mathcal{HR}_{\beta}$ expresses for a set of algorithms the ratio at which each algorithm explored the search space. In conjunction with actual solutions this metric allows for a limited reasoning about the quality of each algorithm in comparison to each other, i. e. if a algorithm was able to explore more feasible areas than others.

Degree of Fulfillment

In order to determine the quality of obtained solutions with respect to user requirements, a simple measure is constructed. The measure is called Degree of Fulfillment (DF) and is defined as follows:

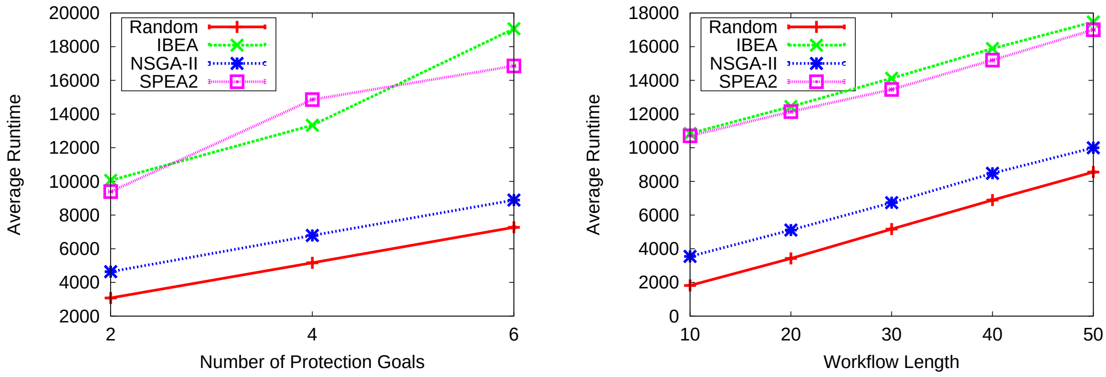
$$DF = \frac{\sum_{i=1}^{n} \frac{pf_i(Q')}{pf_{i,req}}}{n} \tag{7.4}$$

where $n \neq 0$. Here, $pf_{i,req}$ denotes a user's requirements regarding pf_i. The measure can compensate underfulfillment of a protection function with overfulfillment of another and vice versa.

This metric is employed in order to compare the solutions found by different algorithms against each other. Hence only the ratio of DF as obtained by different algorithms is of interest, but not if the actual degree is high or low. For instance, low values can occur for multiple reasons, e. g. unrealistic requirements, unsuitable service alternatives or a combination of both. Since in the conducted experiments all parameters (including user requirements and service alternatives) are generated randomly, such a situation is very probable.

7.4 Configuring COMPAGA

An important question in the context of COMPAGA was which GA to extend. The author decided to choose the GA which runs fastest for varying numbers of protection functions and workflow lengths. Since the repair function will inevitably increase the runtime of the algorithm, hope was that this would keep the runtime of the extended GA still comparable to the other reference algorithms. Hence some initial evaluations were performed in order to study the runtime of the reference algorithms with respect to different numbers of protection goals and workflow lengths without considering compliance[72]. The respective numerical data can be found in Appendix A.4. As can be seen in Figures 7.1(a) and 7.1(b), NSGA-II had a runtime which was approximately half the runtime of the other GAs and slightly higher than the runtime of the random approach. Thus NSGA-II was chosen as the base algorithm to extend.



(a) Different number of protection goals

(b) Different workflow lengths

Figure 7.1: *Runtime evaluation*

In the context of COMPAGA, a question remaining is how to choose p_{rep}? As COMPRepair (cf. Section 5.3.4) affects algorithm runtime, compliance distance as well as $\mathcal{HV}$, this is not a trivial question. Since COMPAGA is an extension of

[72] The results of these initial evaluations have been previously published in [KK13a] and [KFK15]. Here, confidence intervals were added which however cannot be seen due to their narrowness.

NSGA-II, latter algorithm will be used as reference for measuring these effects. Let $\Delta_{Runtime}$, Δ_{CD} and Δ_{HV} denote the effects of COMPRepair on algorithm runtime, compliance distance and $\mathcal{HV}$ respectively. Normalizing these variables utilizing the method of ZENG et al. (cf. Section 3.2.3) yields $\Delta'_{Runtime}$, Δ'_{CD} and Δ'_{HV}. Here, $\Delta_{Runtime}$ has been considered as ordinal descending and Δ_{CD} and Δ_{HV} as ordinal ascending. A "good" p_{rep} can then be determined by measuring the impact of these three variables for different probabilities on different workflow lengths and choosing the "best" value. For this task, the Euclidean distance is calculated for each case (denoted η_{rep}) as follows:

$$\eta_{rep} = \sqrt{\left(\Delta'_{Runtime}\right)^2 + \left(\Delta'_{CD}\right)^2 + \left(\Delta'_{HV}\right)^2} \tag{7.5}$$

This has been performed for different probabilities p_{rep} from 5 to 100 with a stepping of 5. All other settings were as outlined above. The best value in this case means $max\ \eta_{rep}$. Table 7.2 shows the results of these experiments (intermediary results regarding $\Delta'_{Runtime}$, Δ'_{CD} and Δ'_{HV} can be found in Appendix A.3).

Table 7.2: *Simulation results for different probabilities p_{rep}*

	Workflow length				
p_{rep}	10	20	30	40	50
5%	1.0248	1.0684	1.0809	1.1258	1.1912
10%	1.0116	1.0745	1.0432	0.9817	1.1949
15%	0.9941	1.0107	0.9680	1.0650	1.1687
20%	0.9712	0.9730	1.0033	1.0251	1.1301
25%	0.9535	1.0140	0.9711	0.9950	1.1049
30%	0.9550	1.0064	0.9579	0.9675	1.1223
35%	0.9255	0.9829	0.8895	0.9608	1.0649
40%	0.9236	0.9645	0.7664	0.9244	1.0834
45%	0.9125	0.8883	0.8504	0.9077	1.1474
50%	0.8960	0.9271	0.8424	0.9189	1.1170
55%	0.8846	0.8818	0.8070	0.9180	1.1668
60%	0.8685	0.8751	0.7795	0.9036	1.1913
65%	0.8586	0.8613	0.7260	0.9187	1.2155
70%	0.8484	0.8469	0.7370	0.9569	1.2744
75%	0.8243	0.8202	0.6609	0.9755	1.2895
80%	0.8367	0.8133	0.7460	0.9860	1.3138
85%	0.8107	0.7948	0.7687	0.9899	1.3369
90%	0.8035	0.8070	0.7919	1.0076	1.3636
95%	0.8010	0.8269	0.8196	1.0310	1.3846
100%	0.7981	0.9017	0.9082	1.0529	1.4143

Two trends can be seen:

1. η_{rep} increases with workflow size and

2. η_{rep} generally decreases for increasing p_{rep}. However, for workflow lengths $\geqslant 20$ there is a turning point at which η_{rep} constantly increases for increasing p_{rep}. For workflow lengths 20, 30, 40 and 50 the turning point is at $p_{rep} =$ 85%, 75%, 60% and 50% respectively.

The maximum of $\eta_{rep} = 1.4143$ is reached at $p_{rep} = 100$ for a workflow length of 50 which caught the author by surprise. Rudimentary experiments performed initially had led to the assumption that η_{rep} must have a nucleus at about $p_{rep} = 75$. This was actually the reason for calling COMPRepair probabilistically. However, this assumption has been disproved by these detailed experiments. Hence, in all subsequent experiments COMPRepair is called at a probability of $p_{rep} = 100\%$.

7.5 Results

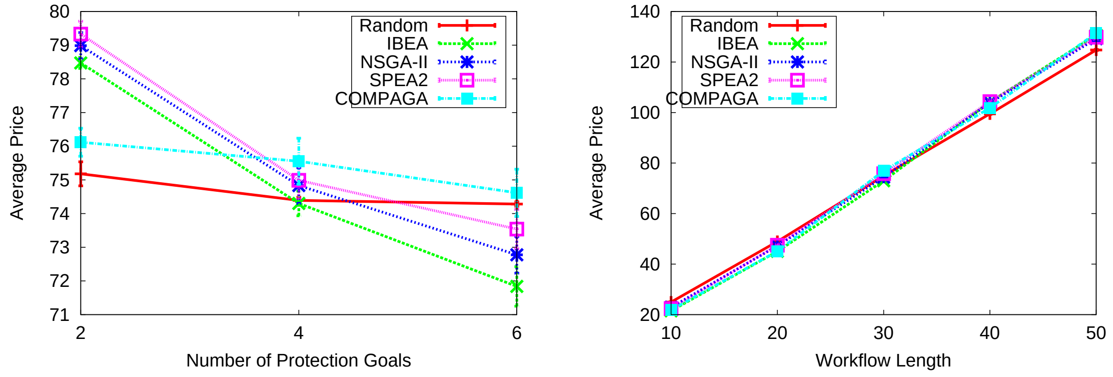
A total of five experiments have been conducted over the course of the conducted numerical evaluations. In the following the main results will be presented by means of plots. It should be noted that in each plot confidence intervals have also been charted. Due to their insignificance however, they are not visible in most cases. The numerical data for all plots can be found in Appendix A.4.

For the sake of readability, the following transformation has been performed on all simulation results reported in this section. Results for a certain workflow length are the average over all different numbers of protection goals. Analogously, results for a different number of protection goals are the average over all different workflow lengths.

7.5.1 QoS of Service Compositions

The first experiment aimed at investigating how well COMPAGA and state-of-the-art GAs can handle problem instances in terms of QoS of service compositions. For this purpose a subset of three QoS attributes was chosen from the model introduced in Section 7.1, namely price, response time (ms) and encryption strength. As has been stated, each algorithm performed a total of 100 runs per problem instance (i. e. different workflow length and different number of protection goals). At each run, the average QoS for all obtained solutions has been determined. For the numerical analysis finally, the average of all runs for a problem instance has been calculated. This leads to a total of 15 results per algorithm for different workflow lengths as well as a different number of protection functions.

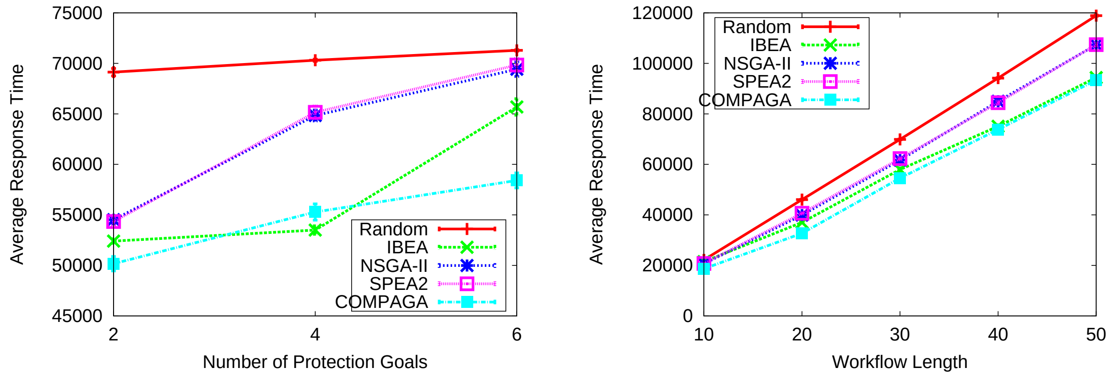
Following are the results for QoS of compositions which have been averaged over different numbers of protection goals (cf. Figures 7.2(a), 7.3(a) and 7.4(a)) as well as workflow lengths (cf. Figures 7.2(b), 7.3(b) and 7.4(b)).



(a) Price vs. number of protection goals

(b) Price vs. workflow length

Figure 7.2: *Experimental results: Price*

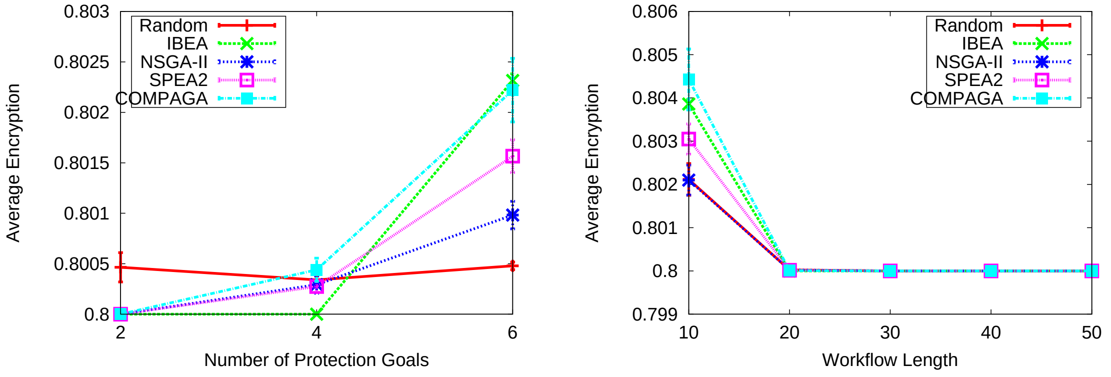


(a) Response time vs. number of protection goals

(b) Response time vs. workflow length

Figure 7.3: *Experimental results: Response time*

The results show that the random approach generally delivered the worst results as expected. A distinct exemption is however price for a low number of protection goals as well as workflow lengths $\geqslant 40$. In these cases, the random approach delivered the best results. NSGA-II and SPEA2 often performed very similar with a major exemption for encryption. For a high number of protection goals and a workflow length of 10, respective results clearly diverted. IBEA and COMPAGA fought a neck-and-neck race and generally delivered the best results. However, there were also some distinct exemptions with regards to a varying number of protection goals. For instance IBEA performed significantly worse than COMPAGA with regards to price and a low number of protection goals while performing significantly better for a high number of protection goals. The same could be observed for response time at six and for encryption at four protection



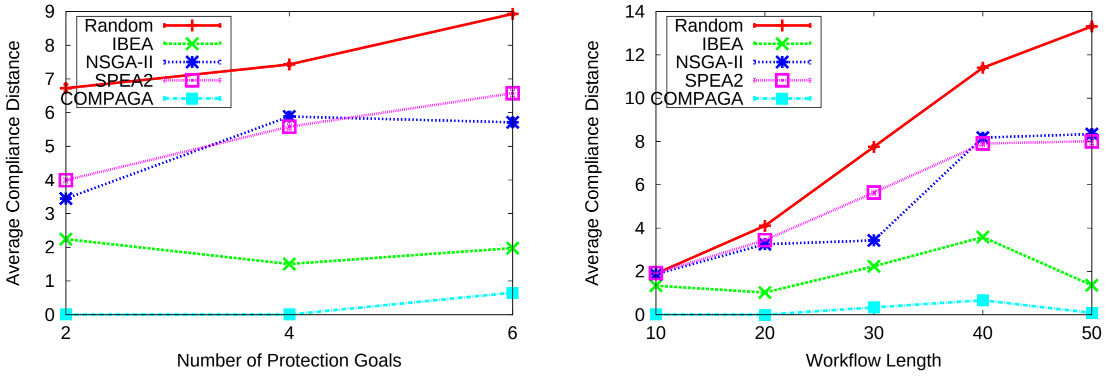
(a) Encryption vs. number of protection goals (b) Encryption vs. workflow length

Figure 7.4: *Experimental results: Encryption*

goals where the results delivered by COMPAGA were significantly better than those delivered by IBEA.

7.5.2 Compliance Distance

The goal of the second experiment was to study the behavior of the employed algorithms with regards to compliance distance. That is, the goal was to see the benefit of COMPAGA compared to the other algorithms. Again, the data has been averaged for different numbers of protection goals (cf. Figure 7.5(a)) and workflow lengths (cf. Figure 7.5(b)).



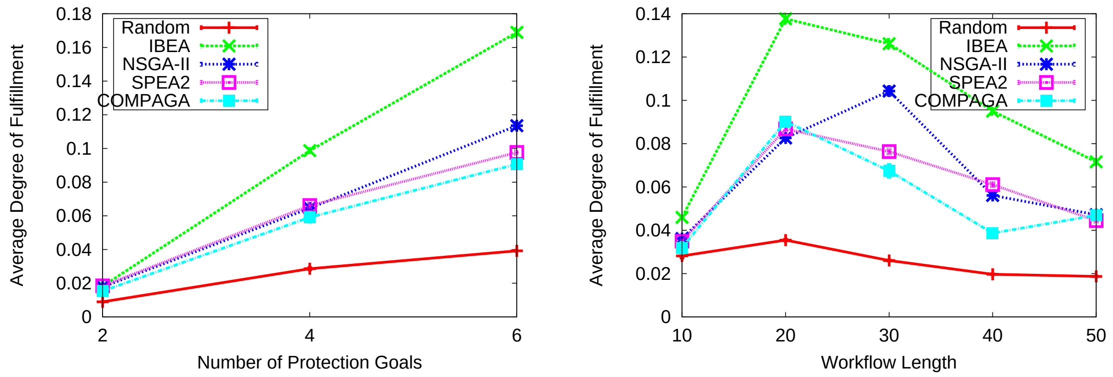
(a) CD vs. number of protection goals (b) CD vs. workflow length

Figure 7.5: *Experimental results: CD*

The results were similar to those of the first experiment. The random approach again delivered the worst results. NSGA-II and SPEA2 again performed very similar and mediocre with a significant exemption at workflow length 30. At this length, NSGA-II performed significantly better than SPEA2. From among the reference algorithms, IBEA performed best but was in every case clearly outperformed by COMPAGA which generally delivered the best results. This proves that the proposed repair operation COMPRepair (cf. Section 5.3.4) works as expected.

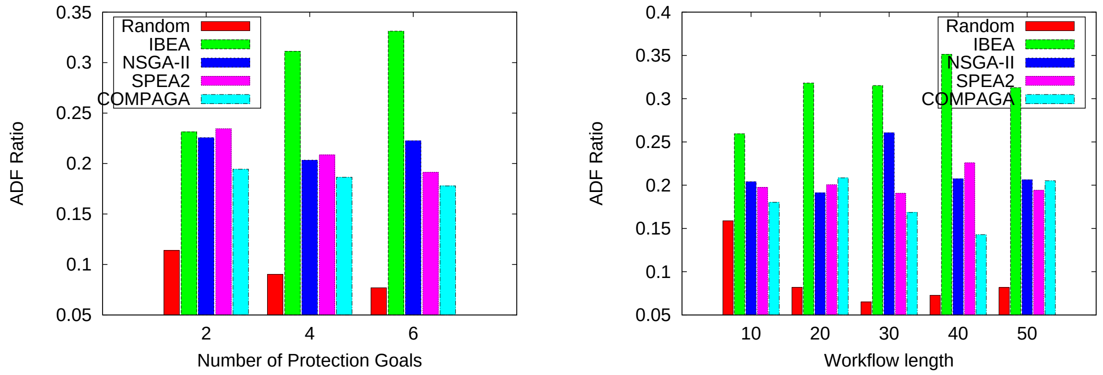
7.5.3 Fulfillment of User Requirements

For this experiment the average DF achieved by the employed algorithms was compared for different numbers of protection goals (cf. Figure 7.6(a)) and workflow lengths (cf. Figure 7.6(b)). The goal was to study how well each algorithm was able to respect user requirements while optimizing a given problem instance. To support understanding of the results, the numerical data has also been plotted as histograms presenting the average DF ratio per algorithm (cf. Figures 7.7(a) and 7.7(b)).



(a) ADF vs. number of protection goals (b) ADF vs. workflow length

Figure 7.6: *Experimental results: ADF*



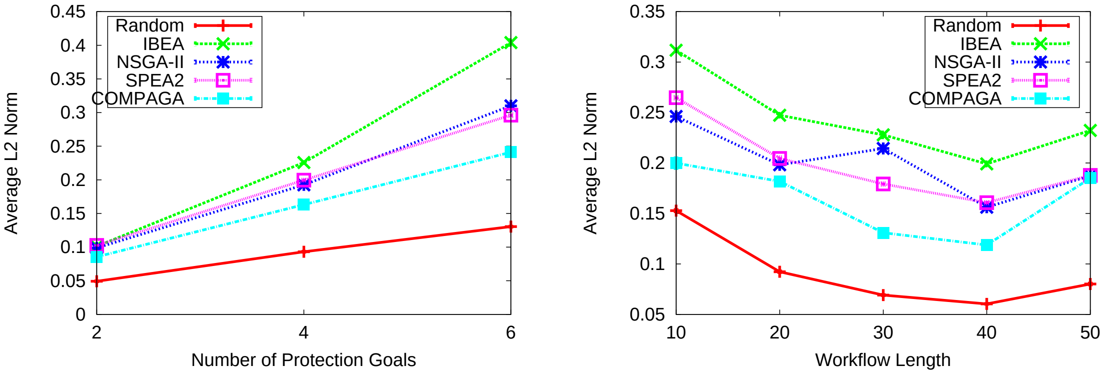
(a) ADF Ratio vs. number of protection goals (b) ADF Ratio vs. workflow length

Figure 7.7: *Histograms of ADF Ratios*

The results of IBEA were found to be superior to the results of the other algorithms, especially in case of more than 2 protection goals and for workflow lengths

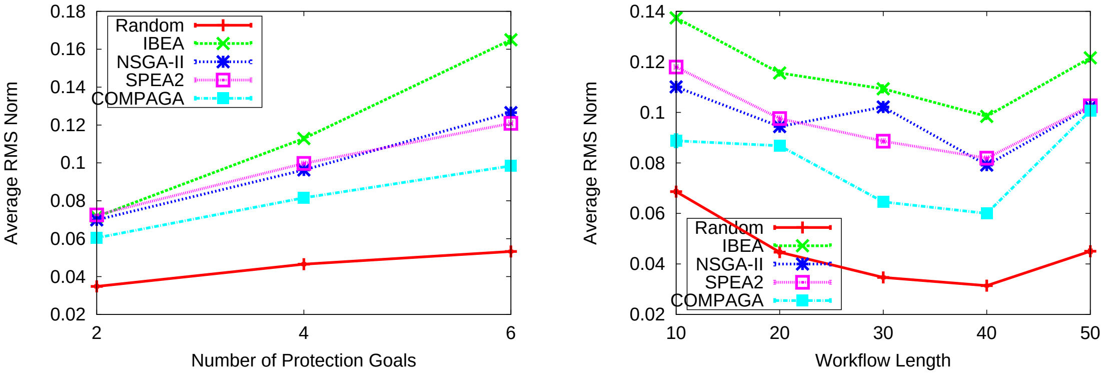
⩾ 20. While the superiority was observed to increase with an increasing number of protection goals, it was found to stay roughly equal for increasing workflow lengths. COMPAGA, NSGA-II and SPEA2 performed similarly mediocre. Again at workflow lengths 30 a sudden increase in the quality of results obtained by NSGA-II can be observed. To the best of the author's knowledge however, no convincing explanation could be found for this strange behavior. The same holds for a sudden decrease of the performance of COMPAGA at workflow length 40. As expected, the random approach generally performed worst and was clearly outperformed by all other algorithms.

The numerical results have been compared to the L2 or Euclidean norm as well as the Root Mean Square (RMS) norm of the function values obtained in the simulations. As can be seen in the Figures 7.8 and 7.9, the results are similar to the numerical data obtained by computing the average DF, however with different value ranges. Hence the trend which has been indicated by the average DF is being confirmed by both the L2 and the RMS norm.



(a) L2 norm vs. number of protection goals (b) L2 norm vs. workflow length

Figure 7.8: *Experimental results: L2 norm*

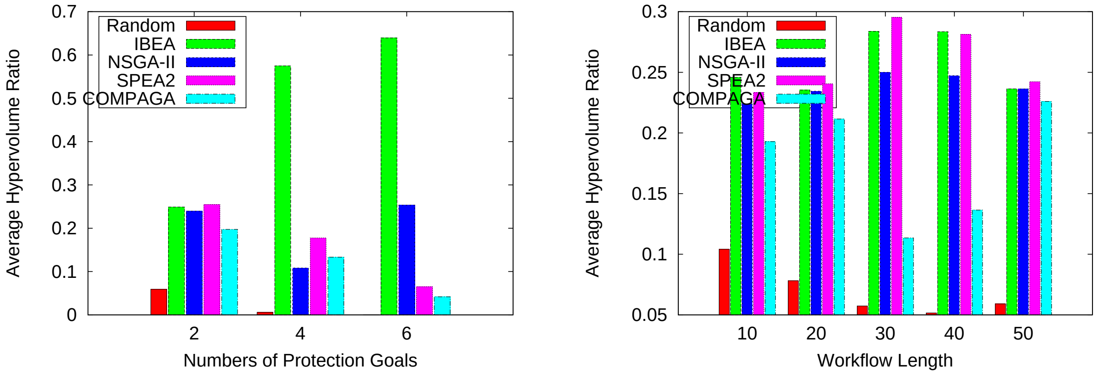


(a) RMS norm vs. number of protection goals (b) RMS norm vs. workflow length

Figure 7.9: *Experimental results: RMS norm*

7.5.4 Exploration and Exploitation of Objective Space

In this experiment, at first the $\mathcal{HV}$ was calculated for each solution obtained by the algorithms. The obtained $\mathcal{HV}$ has been averaged for different numbers of protection goals and workflow lengths (cf. Appendix A.4 for the numerical data). Based on this data, $\mathcal{HR}$ was calculated, again for different numbers of protection goals (cf. Figure 7.10(a)) and workflow lengths (cf. Figure 7.10(b)). It should be noted that the actual numerical results themselves have little meaning for the conducted experiment, i. e. if they are high or low. What is important however is their ratio towards each other since this tells how well each algorithm was able to explore the search space.



(a) HR vs. number of protection goals

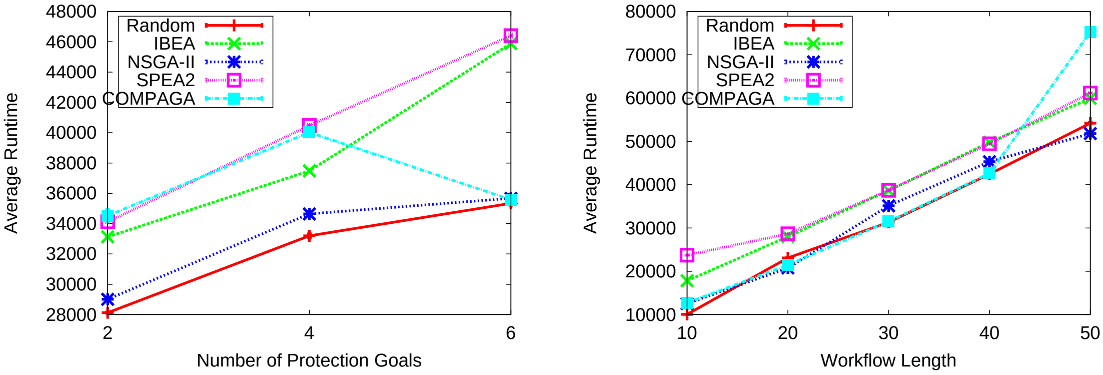
(b) HR vs. workflow length

Figure 7.10: *Experimental results: $\mathcal{HR}$*

The $\mathcal{HR}$ measure reveals that in case of more than 2 protection goals, IBEA clearly outperforms all algorithms. COMPAGA, NSGA-II and SPEA2 performed approximately equally well while the random approach delivered the worst results. Although NSGA-II experiences a sudden increase in performance at six protection goals, this does not change observed trends. The behavior of the algorithms for different workflow lengths however draws a slightly different picture. In this case, IBEA and SPEA2 fight a neck-and-neck race while NSGA-II sometimes comes close to them. The performance of COMPAGA was mediocre and subject to great fluctuations which can be seen very clearly using the $\mathcal{HR}$ measure. The random approach again performed worst.

7.5.5 Runtime

The fifth and final experiment was a classical runtime analysis measured in milliseconds (ms). Each algorithm was studied with regards to different numbers of protection goals (cf. Figure 7.11(a)) and workflow lengths (cf. Figure 7.11(b)).



(a) Runtime vs. number of protection goals

(b) Runtime vs. workflow length

Figure 7.11: *Experimental results: Runtime*

Results for the different perspectives, i. e. numbers of protection goals and workflow length, were quite similar for the reference algorithms and different for COMPAGA. Generally, the random approach was among the fastest. NSGA-II lived up its reputation of being very fast as its runtime was always close to the runtime of the random approach. SPEA2 was among the slowest algorithms while IBEA usually performed a bit faster. Results for COMPAGA were quite different. The average performance for a varying number of protection goals put COMPAGA among the slowest with a sudden decrease of runtime at 6 protection goals which made it one of the fastest in this case. It seems that on average there were only few compliant and feasible solutions for problem instances with 6 protection functions, independently from the workflow length. Hence, the repair operation could not find any suiting services for replacement which caused the runtime of COMPAGA to drop to the level of NSGA-II. This theory is being supported by data for $\mathcal{HR}$ (cf. Section 7.5.4) as well as CD (cf. Section 7.5.2) which are the lowest from all GAs in this case. However, CD is not 0 which means that COMPAGA was not able to find suiting service alternatives to further reduce it. On the contrary, the average performance of COMPAGA for different workflow lengths generally put it among the fastest algorithms with a sudden and dramatic increase of runtime at workflow length 50. Contrary to the former extreme, there seemed to be much more compliant solutions on average in this case. Taking a look again at $\mathcal{HR}$ and CD, this position can be supported. $\mathcal{HR}$ is among the highest from all GAs while CD is 0 for workflow length 50 and > 0 for workflow lengths 30 and 40.

7.5.6 Summary of Results

Compared to the reference algorithms, COMPAGA delivers solutions with slightly better QoS properties, especially with regards to response time. Latter is not surprising since response time is explicitly minimized by COMPRepair when selecting alternative services during repair operations. The compliance distance of the solutions delivered by COMPAGA was the lowest from all algorithms and always close to 0, even for a high number of protection goals and for big workflow models. On the contrary, consideration of user requirements was only below average. Due to the strong alignment of COMPAGA to compliance, search space exploration and exploitation was also below average. With respect to runtime it could be observed that COMPAGA is able to quickly find solutions, but also that the runtime can increase disproportionately in some cases. It was found that the number of protection goals generally had an impact on the QoS of obtained solutions with COMPAGA. Average DF was subject to fluctuations in the face of varying workflow length. Similarly, runtime was also subject to several fluctuations for both different numbers of protection goals as well as different workflow lengths which seems to be caused by sudden changes in the necessary effort to repair a solution.

7.6 Summary

This chapter gave an overview of the simulations performed in order to evaluate COMPAGA. It introduced the experimental setup, the reference algorithms against which COMPAGA was compared and defined some auxiliary metrics which were used for evaluation purposes. Furthermore, the question of choosing an appropriate p_{rep} has been addressed. To the author's surprise and in contradiction to earlier preliminary studies, a probability of $p_{rep} = 100\%$ proved to be the best solution. Hence COMPAGA was configured with this value for the subsequent simulations. The results proved that COMPAGA indeed solves the task it was primarily designed for, namely reducing compliance distance of service compositions much better than other GAs. However it was also revealed that, although the current state is very promising, it also leaves room for several improvements, namely more thorough objective space exploration and exploitation, better consideration of user requirements and a more constant algorithm runtime.

Chapter 8

Case Study

*"In theory, theory and practice are the same.
In practice, they are not"*

– ALBERT EINSTEIN

THE CASE STUDY presented in this chapter originates from the EU FP7 project di.me[73]. It represents a special case with regards to the approach presented in Section 5.2 since only a single service is selected instead of multiple. Compliance aspects were not considered. Hence this case study is cited in order to demonstrate how a security assessment model can be defined by experts using the methods presented in Chapter 5. Any numerical analyses have been omitted. In the following, at first a brief overview is given of the di.me project (Section 8.1). Afterwards the QoS (Section 8.2) and domain model (Section 8.3) for di.me are being presented before shortly presenting the modified service selection algorithm (Section 8.4). A summary concludes this chapter (Section 8.5).

8.1 Background

The di.me project aims for integrating personal data in a personal information sphere, providing a single, user-controlled point of access. This user-controlled server, the so called Personal Service (PS), is integrating information from several existing online social networks and implements intelligent features like semantic reasoning and privacy advisory in order to support users in managing their personal information (cf. Figure 8.1) [SGH+11; TBG+12].

Besides the integration of existing networks and services, di.me provides its

[73] `http://www.dime-project.eu/`

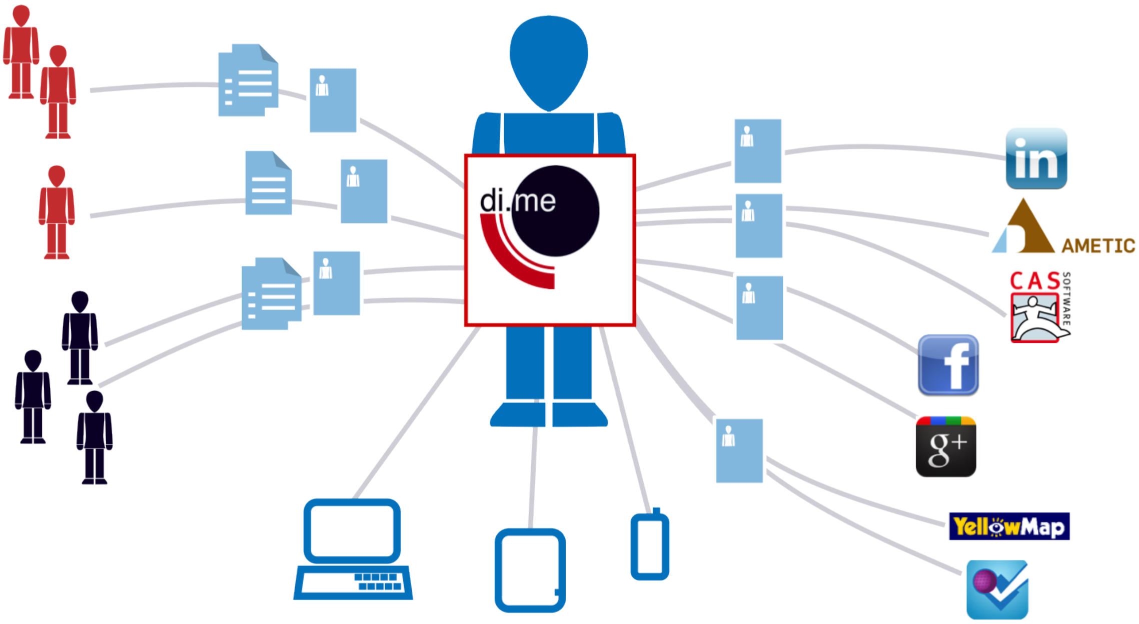


Figure 8.1: *The big picture of di.me*

own Online Social Network (OSN) functionalities not covered by any known alternative (e. g. multiple online identities, or network anonymity). Targeted are several user groups:

1. Business event visitors
2. CRM users and
3. Private individuals

For each user group a dedicated scenario was defined[74]. Important for each scenario is the ease of deployment of di.me servers for lay as well as experienced users. The case study reported in this chapter was especially motivated by the use cases *deploying a new PS* and *moving to a new PS* (e. g. migration).

8.2 QoS Model

In order to chose a fitting service provider for deploying a single-user server or to find an existing multi-tenant server meeting the users' security requirements, several different QoS properties need to be considered. These properties were identified following the AFFINE methodology [BBH+10] which enforces the earlier

[74] Detailed scenario descriptions can be found on the project website: `http://www.dime-project.eu`

consideration of multilateral security requirements along with other nonfunctional requirements (i. e. usability) during the design and development process. In the following, the identified QoS properties are presented in detail and it is discussed why they are of special relevance for the di.me setting.

Response Time

Response time refers to the total time passing between a request sent to the server and the arrival of the response. This value is influenced by several factors (e. g. distance to communication partners, complexity of internal processes, connection capabilities, dedicated RAM, CPU power, etc.). This time is of different importance for different di.me use cases. For instance, it is very important for processes where a user is actively involved, but otherwise of minor relevance when doing automated background processes or asynchronous communication (e. g. a delay of 3-5 sec for sending an E-Mail could be tolerated, but not in the case of a user browsing data with his client).

Storage Strategy

Different hosting facilities use different storage technologies (e. g. RAID), which has influence on the (physical) security of data but also on response time as well as integrity.

Downtime

Downtime is referring to the average time a server is not available per year. Since di.me is a strongly decentralized environment, downtime of a few servers would not harm the whole network, but only single users. A server being offline would mean in the best case (for a single-user deployment) the absence of the data of a single user. In case of multi-tenant deployments this would harm more users, but still less than in a centralized environment.

Single-User-Deployment

A single-user-deployment is a special kind of deployment, where a user has his/her own server. It requires of course a little more administrative work, but also offers complete control over the server and stored personal data. Besides the single-user deployment, there is also the option to use an existing multi-tenant deployment (offered by companies or expert users) to store personal information.

Network Anonymity

Since di.me supports multiple digital identities, which could be pseudonyms intended to be unlinkable, supporting anonymity on the network level is an important requirement [BHW+12]. Depending on the hosting facility, it may not be possible to use Tor[75] or proxied connections and to create Tor hidden services. Therefore this requirement is very important in order to support all di.me functionalities.

Hosting Country
Legal obligations of the country the server is located in is an important factor to be considered. For example, the *USA PATRIOT ACT* (U.S. H.R. 3162) of the U.S. government enforces hosting facilities and service providers located on U.S. ground to give access to the authorities. This is contradicting to some national data protection laws, e. g. as employed by the member states of the European Union[76]. Another legal obligation of interest is the *Russian Bill on Child Protection Law*[77], which came into effect on November 1st, 2012 and puts heavy penalties, including blocking Internet access and heavy fines, on Internet users who use anonymizer and filter-bypass tools (e. g., proxies, VPN). In order to keep private data stored on such servers confidential, additional encryption mechanisms need to be used. Besides legal issues, of course also the physical distance to the server may have minor consequences for the response time.

Number of Users
The number of users located on the same server is only of relevance for a multi-tenant deployment. It can have two different consequences: on the one hand, a large number of users can be of benefit for a single user's anonymity as the IP is no longer a reliable identifier. On the other hand, a PS with a large number of users providing a low downtime might be an indicator for a general stable system and good (administrative) support.

Antivirus
The existence of facilities for detecting viruses is important for user content (uploaded and shared data) and also at the level of the operating system where the di.me server is deployed.

Data Backup Interval
A small backup interval results in a higher availability (and restorability) of data. But it also has to be considered, that it is duplicating all personal files somewhere, maybe causing an information leak. So the backup strategy needs to be considered,

[75] `https://www.torproject.org`

[76] Cf. the EU Directive for Data Protection 2012 (DRAFT)

[77] Cf. Amendment to the Child Protection Act, passed on July 16th, 2012 and amended on September 21st, 2012 (Bill No. 89417-6).

too.

Data Storage Encryption
Complete encryption of stored data can be a requirement for some users, especially when using a (Cloud-) provider not located in the EU (cf. also **Hosting Country**). Strong encryption mechanisms may also influence the performance, e. g. in terms of response time.

Based on evaluations performed by experts from the di.me team, for each of these QoS properties the type, range and utility scores for nominal properties have been identified. The result was a QoS model following the method presented in Section 5.2.1 (cf. Table 8.1).

Table 8.1: *QoS properties for hosting a di.me Private Service*

QoS property	**Type**	**Range of Values**	**Utility Score**
Response Time	discrete	1-5000(ms)	-
Storage Strategy	nominal, enumeration	RAID 0 RAID 1 RAID 5	1 1 2
Downtime	discrete	0-525600 (minutes/year)	-
Single-User Deployment	binary	0 1	0 1
Network anonymity	nominal, enumeration	none proxy Tor	0 1 2
Hosting Country (legal issues)	nominal, enumeration	EU-Country Germany US Russia China	3 4 1 1 1
Number of users	discrete	1-1000	-
Antivirus for uploaded files	binary	0 1	0 1
Antivirus for system files	binary	0 1	0 1
Data backup interval	nominal, enumeration	no backup once a day once a week once a month	0 3 2 1
Data Storage Encryption	binary	0 1	0 1

8.3 Domain Model

In the course of intense discussions, the experts from the di.me team decided to consider the classical CIA triangle of security objectives, i. e. Confidentiality, Availability and Integrity. The influence of the single QoS attributes on these security objectives has been determined based on discussions with Table 8.2 showing the results. Each entry represents a weighting factor and in accordance with the method presented in Section 5.2.2.1, the sum of the weightings in each column must be 1. The corresponding protection functions can be seen in (8.1)-(8.3).

Table 8.2: *Influence of QoS attributes on security objectives in di.me*

QoS property (Weight)	Confidentiality	Integrity	Availability
Response Time (w_1)	0.0	0.0	0.3
Storage Strategy (w_2)	0.0	0.4	0.3
Downtime (w_3)	0.0	0.1	0.4
Single-User-Deployment (w_4)	0.1	0.0	0.0
Network Anonymity (w_5)	0.3	0.0	0.0
Hosting Country (w_6)	0.2	0.0	0.0
Number of Users (w_7)	0.1	0.0	0.0
Antivirus (w_8)	0.0	0.0	0.0
Data Backup Interval (w_9)	0.0	0.3	0.0
Data Storage Encryption (w_{10})	0.3	0.2	0.0

$$pf_C = 0.3Q'^4 + 0.3Q'^5 + 0.2Q'^6 + 0.1Q'^7 + 0.3Q'^{10} \tag{8.1}$$

$$pf_I = 0.4Q'^2 + 0.1Q'^3 + 0.3Q'^9 + 0.2Q'^{10} \tag{8.2}$$

$$pf_A = 0.3Q'^1 + 0.3Q'^2 + 0.4Q'^3 \tag{8.3}$$

In order to define interdependencies between protection functions, another round of discussions has been performed based on the protection functions defined in the first round. As a result of these discussions, the expert group came to the conclusion that confidentiality measures have a weakening impact on availability whereas integrity measures have a slightly strengthening effect. In case of integrity, the experts came to the conclusion that confidentiality measures have a much stronger strengthening effect than integrity on availability. The defined interdependencies between the protection functions along with the strengthening and weakening factors were as follows:

$$pf_{A'} = pf_A - 0.5 * pf_C + 0.1 * pf_I \tag{8.4}$$

$$pf_{I'} = pf_I + 0,8 * pf_C \tag{8.5}$$

8.4 Selecting the Best Deployment Option

To measure how well a deployment option fulfills user requirements, again the *DF* measure has been employed (cf. Section 7.3). Selection is performed with a simple algorithm (cf. Algorithm 8.1) which compares all available alternatives against the user requirements in terms of *DF* in linear time. The algorithm returns the index of the deployment option with maximum *DF* which is subsequently proposed to the user.

Algorithm 8.1: getBestDeploymentOption(*req*, *options*)

```
index := –1;
maxDF := –∞;
for j := 0 to length( options ) do
    currentDF := computeDF( req, options_j );
    if currentDF > maxDF then
        maxDF := currentDF;
        index := j;
    end
end
return index;
```

8.5 Summary

This chapter presented a real world example from the di.me project in which the proposed approach has been adapted. It was shown how the approach has been used to construct a security assessment model by a group of experts. Since the di.me project does not need to evaluate service compositions but different deployment options for the di.me userware, modifications to the proposed approach had to be performed with regards to the selection algorithm. In this case a full enumeration is performed utilizing the DF measure introduced in Section 7.3 to assess different deployment options. Afterwards the deployment option with the highest DF is picked and proposed to the user.

Part IV

Finale

Chapter 9

Conclusion & Outlook

"Every new beginning comes from some other beginning's end."

– Attributed to SENECA

THE FINAL CHAPTER at first summarizes the major contributions of this thesis (Section 9.1) before addressing some open questions which arose during the research (Section 9.2). Next an overview is given about complementary research which might use the work at hand as starting point (Section 9.3). Finally, the thesis closes with some general considerations on the current state and the future of service composition (Section 9.4).

9.1 Contributions

In the following the contributions of this work are being summarized with regards to the research questions as defined in Section 1.2.

Research Question 1 *What are the requirements for a method for service composition with respect to interdependent security and compliance?*

A total of six requirements have been identified in the course of this work, based on a literature review as well as considerations on pros and cons of different approaches to composition. It was identified that such a method at first needs to be **complete**, i. e. it needs to support all relevant composition patterns in the context of business process modeling. Furthermore, the approach should be able to **consider multiple QoS attributes**, either general or domain-specific. With regards to service selection, the search space should be **explored thoroughly** by default without prioritizing any QoS attributes over others, except if the decision maker decides otherwise. Mathematically this means formulating service selection as MOO problem by default and transforming it to a SOO problem in case the decision maker is able to formulate preferences. In order to be useful for real-world scenarios, the approach has to be **efficient**. For service selection as a MOO problem this means utilizing (meta-)heuristics such as GAs in order to obtain near-optimal solutions in polynomial time. Finally, such as method obviously must **consider interdependent security objectives** and be **compliance-aware**. A literature survey in the field of service composition revealed that interdependent security has not been covered yet. Compliance-awareness per se was also found to be a research gap. However, some approaches were found to have the potential to partly cover compliance aspects in service selection, particularly with regards to time limits. All reviewed approaches for service selection were found to only partly cover thorough search space exploration. Particularly, approaches either consider SOO or MOO problems, but do not consider transforming the problem from one representation to the other. Less than half of the approaches surveyed were found to be complete in terms of composition patterns while most works support multiple QoS attributes and employ efficient selection mechanisms.

Research Question 2 *How can interdependent security objectives be assessed in service composition?*

An approach has been presented which utilizes the concept of structural decomposition introduced by WANG and WULF for assessing security of systems. The assessment is based on the influence of single QoS properties on a security goal as determined by domain experts using decision making tools such as AHP [WW97]. Such a utility function assessing the fulfillment of a single security goal is called a protection function here. This basic model has been extended in order to cover the structures of interdependent security (namely strengthening, weakening and implicating relationships) which have been first described by WOLF and PFITZMANN [WP00]. In case of strengthening and weakening relationships between security goals, the proposed method relies on estimations by domain experts. For implicating relationships, a plausibility check is performed in order to guarantee the validity of modeled implication. In its current form however, the proposed method does not allow for cyclic dependencies between protection functions. Furthermore, in case of implicating relationships, protection functions must not have a predecessor. The result of the proposed modeling approach is a MOO problem which can be tackled with standard tools for optimization such as GAs. Since GAs are frequently used in service selection, the proposed approach is compatible to the greatest possible extent with established methods for QoS-aware service selection and hence can be used to retrofit existing systems for service composition.

The possibility to formulate constraints on protection functions corresponds to the notion of "good enough security" proposed by SANDHU [San03].

Research Question 3 *How can compliance be verified in service composition?*

Based on the notion of compliance distance which was first introduced by SADIQ et al. [SGN07], a method has been developed to track and record services in a composition that violate compliance requirements. Furthermore, a custom repair method has been presented and discussed that utilizes this information in order to exchange these services in an attempt to make the composition compliant again. Since GAs are frequently used for service selection, it has been shown how to extend existing algorithms with these methods in a drop-in fashion to make them compliance-aware. This approach has been demonstrated with NSGA-II which led to a new algorithm called COMPAGA. This algorithm further performs an initial feasibility check to see if there exists at least one compliant solution without considering user requirements. If this is not the case, COMPAGA aborts. However, a positive feasibility check is no guarantee that a solution exists which satisfies compliance as well as user requirements.

Research Question 4 *What are the impacts of considering interdependent security and compliance on service composition?*

Initial studies of problem instances on interdependent security with reference algorithms showed that NSGA-II delivers results the fastest. Based on these results, NSGA-II has been extended to COMPAGA and subsequently simulations have been performed on problem instances considering compliance. The results showed that COMPAGA is able to clearly reduce the compliance distance of service compositions. However, objective space exploration and exploitation are not as good as for the reference algorithms. The same holds for consideration of user requirements. The runtime of COMPAGA proved to be subject to disproportional fluctuations. Different numbers of protection goals were found to have an impact on QoS of obtained solutions as well as on algorithm runtime. On the other hand, different workflow lengths led to fluctuations in terms of average DF and algorithm runtime. Hence, the results show that with respect to considering interdependent security and compliance, COMPAGA is a viable first step which however yields room for improvement.

9.2 Open Questions

Open questions can be categorized into two distinct groups, namely regarding the proposed approach for assessing interdependent security (cf. Section 5.2) and for compliance-aware service selection with COMPAGA (cf. Section 5.3). In the following these questions and their implications on the work at hand will be discussed.

Assessing Interdependent Security of Service Compositions

Currently assessment is being performed on a global scope, i. e. for a service composition as a whole. The question of how this global view can be combined with a local view remains open. Such a local view would enable decision makers to either replace single services in a composition which do not provide a certain minimum of security or to remodel the underlying business process if no suiting service alternatives exist at all.

Another question remains regarding the basic model employed for assessing interdependent security. Since it relies on protection functions which need to be modeled by domain experts, there always exists the danger that these functions do not adequately reflect reality. Hence an open question remains how such protection functions can be validated with formal methods. Since to the best of the author's knowledge there exists no formal security model which considers interdependence effects in a general fashion, this question is considered a major challenge.

Compliance-Aware Service Selection with COMPAGA

As has been demonstrated in Section 5.3.2, the proposed feasibility check can guarantee that either none or at least one compliant solution exists for a given process model and service alternatives. However if a compliant solution also fulfills user requirements, is another question. In case of a few compliant solutions, this question can be answered by checking them against the requirements using e. g. full enumeration. However in case of "big" search spaces with maybe hundreds of compliant solutions, this decision becomes an optimization problem on its own. Hence the question arises how this problem can be modeled and efficiently be solved and what the adjective "big" exactly means in this context.

Another question with regards to COMPAGA is what might be the impact of using custom genetic operators for selection, crossover and mutation instead of general-purpose implementations? Due to the *No Free Lunch Theorems* of WOLPERT and MACREADY it is justified to assume that custom operators will have a positive impact on the performance of COMPAGA [WM97].

In this thesis it was assumed that compliance requirements on single tasks are binary, i. e. they either apply or do not. Often however, compliance requirements are more complex and offer a certain degree of freedom. In simple cases, this freedom consists of a choice between multiple alternatives. More complex compliance requirements allow certain actions only if a certain set of prerequisites is fulfilled. Modeling such complex requirements needs a more expressive language and subsequently raises the need to consider them during service selection. Hence it remains open

a) how such complex requirements might be modeled,

b) how these requirements can be transformed into a problem formulation and

c) how the resulting optimization problem can be solved efficiently.

Finally, it remains open how compliance-aware service selection can be extended to cover the full business process lifecycle. As has been discussed in Section 2.4, considering all compliance aspects requires service selection to be flanked with modeling and monitoring techniques. The author expects such a holistic view to yield synergy effects which might be utilized by sophisticated approaches for BPM. Hence the question is how such a holistic approach needs to be designed in order to reap maximum benefits.

9.3 Complementary Research

In this work several assumptions have been made regarding service composition for the sake of simplification (cf. Section 3.2.1). In particular, it was assumed that all services within a service class have uniform interfaces, i. e. that types, naming and ordering of input and output parameters are equal. However, in real settings this is rarely the case. Hence additional research is required to appropriately model heterogeneous service interfaces. For the selection process this effectively means that the search space can be restricted to those combinations which are functionally valid in terms of interfaces. For some business processes this might even mean that no concrete sequence of services exists which can implement the process. Similarly, consideration of heterogeneous data is a direction which requires more research.

In the literature about business process compliance (including this thesis), the notion of *location* is solely being used in a geographic sense. Due to the globalization of the economy however, such a view is not adequate. As an example consider multi-national corporations such as GOOGLE and AMAZON. Although their services are hosted in multiple countries, they are primarily subject to US law. Hence, although their service offerings are also being hosted in the EU, the services do not always comply with EU law. Hence further research is necessary to answer the question, how the notion of *location* needs to be extended in this context in order to cover these aspect of multi-national service providers.

Currently, the influence of single QoS attributes on protection functions in terms of factor weightings is being determined by domain experts employing decision support tools such as AHP. Major corporations however usually employ risk models for assessing the security of their IT infrastructure. In Germany, the risk analysis method of the Bundesamt für Sicherheit in der Informationstechnik (BSI) is being commonly used [BSI08]. Hence an interesting and important research direction would be to bridge the gap between the model proposed in this thesis and risk models. Particularly, the aim of this research would be to deduct factor weightings from risk models already established by corporations. This might not only strengthen the basis of the proposed approach, but also foster its adoption by real world corporations.

As previously stated in Chapter 6, SSW is not intended to be a mature product but a proof of concept. However, in order to enable a broad adoption of tools for service composition which implement the approaches presented in this thesis, it is necessary to perform further research. This research needs to focus on how service composition can be best supported by tools. Ideally such tool support should span the whole BPM lifecycle in order to benefit from synergy effects. An interesting question in this context is how such tools might foster the utilization of progressive optimization methods for service selection. Optimization methods such as STEM and SEMOPS usually require several steps of interaction with the decision maker but generally lead to "better" results from a user's perspective [CLV07]. Up to now however, there is to the best of the author's knowledge a lack of tool support for service composition with progressive optimization methods. Hence a study on the impact of progressive optimization methods on the quality of obtained solutions in service composition is necessary to develop an understanding about the utility of such methods for this field of study.

9.4 Final Remarks

"One more special message to go,
and then I'm done and I can go home."

– NIRVANA

This work proposed initial approaches to bridge the gap between service composition on the one hand and interdependent security as well as compliance on the other hand. Since an increasing number of apparently monolithic services are actually composed of more fine-grained components, considering these aspects becomes crucial for organizations. The trend towards business processes which one day shall (partly) configure themselves autonomously based on a (formal) description from a set of single single services, further increases the importance of these topics. It shall not be concealed here that it is still a long way to go until this goal is achieved, especially since today business processes are still being implemented "traditionally" (i. e. manually). Hence a paradigm shift in society seems to be inevitable to prepare the ground for a broad adoption of service composition by organizations. Even though it may still be a long time from now until this change takes place, several approaches addressing topics related to service composition have already been proposed for the time after. This thesis contains two of them.

Appendices

Appendix A

Support Materials

The materials presented here provide further details to topics that would have otherwise exceeded the scope of this thesis. They are intended to support the argumentation of sections referencing them.

A.1 Mapping of Composition Patterns from BPMN to BPEL

In the following it will be presented how to express the CPs presented in Section 3.2.2 as BPEL 2.0 statements. This is particularly important for model-driven transformations of BPMN process models to executable BPEL code. Due to the scope of the language however, all details cannot be covered here. Instead the interested reader is referred to the corresponding specification (cf. [OAS07]). With the available language constructs of BPEL 2.0 however, discriminators (CPs 5 and 7) cannot be expressed.

Sequence (CP 1)

For representing sequences, BPEL offers the `sequence` element. All activities within this element are performed in the order of their appearance (cf. Listing A.1).

Listing A.1: *Sequence in BPEL*

```
<sequence>
    ...
</sequence>
```

Loop (CP 2)

BPEL offers two alternatives for expressing loops. The first alternative is the `while` element which first evaluates a termination condition before executing the loop (cf. Listing A.2). The second alternative is the `repeatUntil` element which works in reverse order, i. e. execution before evaluation (cf. Listing A.3) and thus corresponds to a `do-while` loop known from languages such as C++ and Java.

Listing A.2: *Loop in BPEL: Alternative 1*

```
<while>
   <condition>...</condition>
    ...
</while>
```

Listing A.3: *Loop in BPEL: Alternative 2*

```
<repeatUntil>
    ...
   <condition>...</condition>
</repeatUntil>
```

XOR-Split followed by XOR-join (CP 3)

This pattern can be expressed in BPEL using the `if`, `elseif` and `else` elements which represent conditional branching as known from plenty of programming languages (cf. Listing A.4)

Listing A.4: *Representing XOR-split and join in BPEL*

```
<if>
   <condition>...</condition>
    ...
   <elseif>
      <condition>...</condition>
       ...
   </elseif>
   <else>
       ...
   </else>
</if>
```

AND-split followed by AND-join (CP 4)

BPEL defines the `flow` element which allows to group several activities. It completes only after all enclosed activities have been completed (cf. Listing A.5).

Listing A.5: *Representing AND-split and join in BPEL*

```
<flow>
    ...
</flow>
```

OR-split followed by OR-join (CP 6)

This pattern is not directly supported by means of a dedicated element in BPEL. However, it can be constructed by nesting one or more `link` elements in a `flow` element. A `link` element defines one or more conditional transitions between activities. Depending on the formulation of these conditions, one or more can be true at any time which corresponds to the OR-pattern (cf. Listing A.6).

Listing A.6: *Representing OR-split and join in BPEL*

```
<flow>
   <links>
      <link name="link-name" />
   </links>
   <receive>
      <sources>
         <source linkName="link-name">
            <transitionCondition>
               ...
            </transitionCondition>
         </source>
      </sources>
   </receive>
   <invoke>
      <targets>
         <target linkName="link-name" />
      </targets>
   </invoke>
</flow>
```

A.2 Comparison of Related Works on QoS-aware Service Selection

Table A.1 gives details about state-of-the-art approaches for QoS-aware service selection which have been discussed in Chapter 4. The respective works have been classified with respect to utilized problem modeling and solving approaches.

Table A.1: *Comparison of related works on QoS-aware Service Selection*

		Approach	Problem Modeling	Solving Approach	Optimization Mode	Multi-Objective Optimization
Exact Approaches	Graph-based	Zeng et al. (2003)	DAG	ILP Solver (IBM OSL)	global	SAW
		Menascé (2004)	DAG	Full Enumeration	global	supported
		Zeng et al. (2004)	DAG	local: highest utility global: ILP Solver (IBM OSL)	local; global	SAW
		Grønmo and Jaeger (2005)	Directed Graph	Full Enumeration	local and global	SAW
		Cardellini et al. (2006,2007)	Customized DAG	MatLab and SNOPT	global	SAW
		Yu et al. (2007)	DAG	MCSP, MCSP_General	global	Custom Utility Function
		Yan et al. (2012)	Customized DAG (PlanGraph)	Based on Dijskstras algorithm	global	SAW
		Heinrich and Lewerenz (2013)	DAG	MCSP	global	Custom Utility Function
	Combinatorial	Aggarwal et al. (2004)	ILP	LINDO Solver	global	Linear combination
		Yu and Lin (2004)	MCKP; DAG	Pisinger's algorithm	global	Custom Utility Function
		Ardagna and Pernici (2006)	MMKP, MIP	CPLEX Solver	local and global	SAW

Continued on next page

Table A.1 – continued from previous page

		Approach	Problem Modeling	Solving Approach	Optimization Mode	Multi-Objective Optimization
Exct. Apr.	**Comb.**	YU et al. (2007)	MMKP; ILP	MMKP: BBLP, ILP: WS_IP	global	Custom Utility Function
		MENASCÉ et al. (2010)	Non-Linear Programming	JOSes algorithm and custom heuristic	global	supported
(Meta-)Heuristic Approaches	**Heuristics**	XIAO and BOUTABA (2005)	Undirected Graph	Custom Heuristic	global	not supported (cost min.)
		BERBNER et al. (2006)	MIP	H1_RELAX_IP, H2_SWAP H3_SIM_ANNEAL	global	SAW
		YU et al. (2007)	DAG; MMKP	MCSP-K, MCSP-K_General, WS_HEU, WFlow	global	Custom Utility Function
		ALRIFAI et al. (2009, 2012)	MIP	LPSolve Solver	local and global	SAW
		LIANG et al. (2009)	Assignment Problem	Custom local search heuristic	global and local	weighting scheme
		MABROUK et al. (2009)	Decision Tree	Custom Heuristic	global	Custom Utility Function
		ARDAGNA and MIRANDOLA (2010)	Non-linear MCOP	Custom Heuristic	global	SAW
		KLEIN et al. (2011)	DAG	Heuristic based on Hill Climbing	global	Utility Function based on SAW

Continued on next page

Table A.1 – continued from previous page

		Approach	Problem Modeling	Solving Approach	Optimization Mode	Multi-Objective Optimization
(Meta-)Heuristic Approaches		Lama et al. (2012)	Dependency Graph	Heuristic based on A* search algorithm	global	not supported
	Genetic Algorithms	Canfora et al. (2005)	MMKP	custom GA	global	Custom Utility Function
		Claro et al. (2005)	MMKP	NSGA-II	local and global	supported
		Jaeger and Mühl (2007)	Custom model	custom GA	local and global	SAW
		Canfora et al. (2008)	MMKP with local constraints	custom GA	local and global	Custom Utility Function
		Wada et al. (2008)	MMKP	custom GA E^3 MOGA	local and global	supported
		Ye et al. (2011)	DAG	custom GA	local and global	SAW
		Klein et al. (2012)	graph model	custom GA NetGA	local and global	not supported
		Wagner et al. (2012)	graph model	custom GA SHUURI	local and global	supported
Semantic		Lee (2003)	MMKP	OSL Solver	global	not supported
		Arpinar et al. (2005)	Directed Graph	IMA algorithm	global	SAW
		Wang et al. (2007)	Fuzzy Logic	Fuzzy Inference	global	QoS Matching
		Lécué and Mehandjiev (2011)	CSP	custom GA	global	Fitness Function

A.3 Intermediary Results for Determining p_{rep}

This appendix shows all intermediary data which lead to the results presented in Table 7.2 (cf. Section 7.4). Tables A.2, A.3, and A.4 show the raw results for $\Delta_{Runtime}$, Δ_{CD} and Δ_{HV} while Tables A.5, A.6 and A.7 show normalized data.

Table A.2: *Simulation results:* $\Delta_{Runtime}$

	Workflow length				
p_{rep}	10	20	30	40	50
5%	10.1000	25.7500	19.5500	15.1200	24.6700
10%	15.8100	30.3300	42.5700	113.5300	49.0100
15%	25.5000	72.8600	95.3400	68.4200	77.3000
20%	36.2200	94.3800	77.1400	98.9500	126.4200
25%	45.5000	73.5500	96.2400	125.2000	169.3100
30%	44.0100	80.0800	111.5800	149.5100	181.0600
35%	58.0100	92.7500	147.1400	162.9300	244.5000
40%	58.5000	105.7500	233.7300	200.2700	270.1700
45%	65.0000	148.3500	183.4500	223.0800	273.0500
50%	72.4000	127.8700	185.6600	230.4100	326.8600
55%	77.9100	149.5700	211.3500	255.3400	335.1900
60%	84.7000	154.7400	233.0200	296.1000	379.7700
65%	89.4000	164.4600	260.0000	298.1600	371.3800
70%	94.2000	172.6500	265.6200	301.4900	390.3900
75%	106.2200	190.1200	341.2200	315.6100	399.2400
80%	98.7000	200.9100	287.5700	321.5200	410.4300
85%	112.5000	210.2000	285.3000	342.6300	436.9100
90%	115.6000	215.7900	296.3800	354.0300	444.6900
95%	117.0000	224.5900	302.8400	363.7700	508.6100
100%	116.1000	214.3900	298.3800	404.1100	501.3700

Table A.3: *Simulation results:* Δ_{CD}

	Workflow length				
p_{rep}	10	20	30	40	50
5%	0.0400	0.7273	0.6644	1.3507	5.3227
10%	0.0600	0.8356	0.7580	1.7482	6.3689
15%	0.6600	0.9333	0.9140	2.2569	6.7379
20%	0.1400	0.9472	1.0268	2.4516	7.3571
25%	0.2200	0.9911	1.1766	2.6339	7.8746
30%	0.1900	1.0323	1.1600	2.8486	8.4202
35%	0.3200	1.0623	1.3539	3.1590	8.6091
40%	0.1100	1.2073	1.3785	3.6401	9.2206
45%	0.4700	1.2288	1.5862	3.8887	10.1885
50%	0.7100	1.3032	1.7227	4.2001	10.2516
55%	0.3800	1.3544	1.8094	4.5265	10.9234
60%	0.4900	1.4044	1.9732	4.9149	11.4796
65%	0.6400	1.4867	2.0631	5.1512	11.6275
70%	0.9400	1.6075	2.4701	5.7129	12.2961
75%	0.9300	1.6728	2.7985	5.9181	12.4901
80%	0.9100	1.8007	3.2515	6.0735	12.6541
85%	1.2400	2.0021	3.7553	6.3336	12.9351
90%	1.3000	2.2757	4.3965	6.4709	13.0779
95%	1.4500	2.6310	4.9203	6.5891	13.1803
100%	1.7700	3.0549	5.4107	6.7100	13.2985

Table A.4: *Simulation results:* Δ_{HV}

	Workflow length				
p_{rep}	10	20	30	40	50
5%	-0.0196	0.0692	0.0705	0.1006	0.1152
10%	-0.0234	0.0819	0.0739	0.1133	0.1262
15%	-0.0227	0.0910	0.0881	0.1182	0.1302
20%	-0.0242	0.0905	0.0906	0.1237	0.1377
25%	-0.0236	0.0944	0.0904	0.1296	0.1451
30%	-0.0260	0.0970	0.1000	0.1336	0.1512
35%	-0.0301	0.0955	0.0921	0.1384	0.1534
40%	-0.0314	0.0975	0.0989	0.1419	0.1623
45%	-0.0294	0.0934	0.1030	0.1461	0.1741
50%	-0.0349	0.0965	0.0991	0.1531	0.1781
55%	-0.0339	0.0895	0.1006	0.1636	0.1902
60%	-0.0399	0.0904	0.1017	0.1720	0.2009
65%	-0.0422	0.0912	0.0919	0.1769	0.2084
70%	-0.0470	0.0895	0.0984	0.1885	0.2267
75%	-0.0479	0.0896	0.1017	0.2002	0.2313
80%	-0.0529	0.0949	0.1093	0.2050	0.2425
85%	-0.0559	0.0905	0.1130	0.2099	0.2511
90%	-0.0599	0.1006	0.1221	0.2197	0.2629
95%	-0.0631	0.1163	0.1315	0.2316	0.2752
100%	-0.1087	0.1465	0.1673	0.2491	0.2884

Table A.5: *Normalized simulation results:* $\Delta'_{Runtime}$

	Workflow length				
p_{rep}	10	20	30	40	50
5%	1.0000	0.9686	0.9810	0.9899	0.9708
10%	0.9885	0.9594	0.9349	0.7965	0.9219
15%	0.9691	0.8741	0.8290	0.8830	0.8652
20%	0.9476	0.8309	0.8655	0.8218	0.7667
25%	0.9290	0.8727	0.8272	0.7691	0.6806
30%	0.9320	0.8596	0.7964	0.7203	0.6571
35%	0.9039	0.8342	0.7251	0.6934	0.5298
40%	0.9029	0.8081	0.5514	0.6185	0.4783
45%	0.8899	0.7227	0.6523	0.5728	0.4725
50%	0.8750	0.7638	0.6478	0.5581	0.3646
55%	0.8640	0.7202	0.5963	0.5081	0.3479
60%	0.8504	0.7099	0.5528	0.4263	0.2585
65%	0.8409	0.6904	0.4987	0.4222	0.2753
70%	0.8313	0.6739	0.4874	0.4155	0.2371
75%	0.8072	0.6389	0.3358	0.3872	0.2192
80%	0.8223	0.6172	0.4434	0.3753	0.1969
85%	0.7946	0.5986	0.4480	0.3330	0.1438
90%	0.7884	0.5874	0.4257	0.3101	0.1282
95%	0.7856	0.5697	0.4128	0.2905	0.0000
100%	0.7874	0.5902	0.4217	0.2096	0.0145

Table A.6: *Normalized simulation results:* Δ'_{CD}

	Workflow length				
p_{rep}	10	20	30	40	50
5%	0.0000	0.0518	0.0471	0.0989	0.3984
10%	0.0015	0.0600	0.0542	0.1288	0.4773
15%	0.0468	0.0674	0.0659	0.1672	0.5052
20%	0.0075	0.0684	0.0744	0.1819	0.5519
25%	0.0136	0.0717	0.0857	0.1956	0.5909
30%	0.0113	0.0748	0.0845	0.2118	0.6321
35%	0.0211	0.0771	0.0991	0.2352	0.6463
40%	0.0053	0.0880	0.1010	0.2715	0.6924
45%	0.0324	0.0897	0.1166	0.2903	0.7654
50%	0.0505	0.0953	0.1269	0.3138	0.7702
55%	0.0256	0.0991	0.1335	0.3384	0.8209
60%	0.0339	0.1029	0.1458	0.3677	0.8628
65%	0.0453	0.1091	0.1526	0.3855	0.8740
70%	0.0679	0.1182	0.1833	0.4279	0.9244
75%	0.0671	0.1232	0.2081	0.4433	0.9390
80%	0.0656	0.1328	0.2422	0.4551	0.9514
85%	0.0905	0.1480	0.2802	0.4747	0.9726
90%	0.0950	0.1686	0.3286	0.4850	0.9834
95%	0.1063	0.1954	0.3681	0.4940	0.9911
100%	0.1305	0.2274	0.4051	0.5031	1.0000

Table A.7: *Normalized simulation results:* Δ'_{HV}

	Workflow length				
p_{rep}	10	20	30	40	50
5%	0.2242	0.4479	0.4513	0.5270	0.5637
10%	0.2148	0.4800	0.4597	0.5591	0.5915
15%	0.2165	0.5028	0.4955	0.5714	0.6016
20%	0.2128	0.5017	0.5019	0.5852	0.6204
25%	0.2142	0.5114	0.5014	0.6001	0.6390
30%	0.2081	0.5179	0.5254	0.6101	0.6544
35%	0.1979	0.5141	0.5057	0.6221	0.6600
40%	0.1945	0.5191	0.5226	0.6311	0.6823
45%	0.1995	0.5088	0.5331	0.6415	0.7122
50%	0.1858	0.5168	0.5233	0.6592	0.7222
55%	0.1884	0.4990	0.5271	0.6856	0.7528
60%	0.1732	0.5013	0.5298	0.7068	0.7797
65%	0.1674	0.5034	0.5051	0.7191	0.7986
70%	0.1552	0.4990	0.5215	0.7484	0.8447
75%	0.1531	0.4993	0.5299	0.7779	0.8561
80%	0.1403	0.5126	0.5488	0.7901	0.8843
85%	0.1328	0.5015	0.5583	0.8023	0.9060
90%	0.1229	0.5271	0.5813	0.8270	0.9359
95%	0.1147	0.5665	0.6049	0.8571	0.9668
100%	0.0000	0.6426	0.6949	0.9009	1.0000

A.4 Numerical Data of Experiments

This appendix shows the raw data used for generating the plots depicted in Chapter 7. In order to enhance readability, data has been generally rounded to four digits after decimal point except for Runtime (Tables A.8 and A.18) and Response Time (Table A.10). Since in these cases the data has at least four digits, the decimal portion has no significant impact on the measured values. Each of the following tables consists of two parts. In the upper part, simulation results have been averaged over different numbers of protection goals while the data in the lower part is averaged over different workflow lengths. The symbols $\mathrm{CI}_{\mathrm{lower}}$ and $\mathrm{CI}_{\mathrm{upper}}$ are used to denote the lower and the upper boundary of the confidence interval respectively. Since $\mathcal{HR}$ itself is not the result of simulations but of computation based on $\mathcal{HV}$, Table A.17 does not include confidence intervals. The sequence of tables follows the sequence of figures in Chapter 7.

Table A.8: *Initial runtime evaluation results*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2
2	Mean	3079	10063	4638	9395
	$\mathrm{CI}_{\mathrm{lower}}$	3067	10039	4622	9337
	$\mathrm{CI}_{\mathrm{upper}}$	3091	10087	4654	9453
4	Mean	5167	13333	6792	14859
	$\mathrm{CI}_{\mathrm{lower}}$	5150	13307	6774	14800
	$\mathrm{CI}_{\mathrm{upper}}$	5183	13359	6810	14917
6	Mean	7272	19062	8893	16854
	$\mathrm{CI}_{\mathrm{lower}}$	7241	19018	8872	16795
	$\mathrm{CI}_{\mathrm{upper}}$	7303	19106	8914	16912
Workflow Length		Random	IBEA	NSGA-II	SPEA2
10	Mean	1821	10845	3546	10703
	$\mathrm{CI}_{\mathrm{lower}}$	1815	10823	3533	10662
	$\mathrm{CI}_{\mathrm{upper}}$	1827	10867	3558	10745
20	Mean	3427	12444	5113	12144
	$\mathrm{CI}_{\mathrm{lower}}$	3411	12420	5097	12087
	$\mathrm{CI}_{\mathrm{upper}}$	3443	12468	5128	12202
30	Mean	5171	14121	6736	13459
	$\mathrm{CI}_{\mathrm{lower}}$	5146	14084	6719	13399
	$\mathrm{CI}_{\mathrm{upper}}$	5196	14158	6754	13519
40	Mean	6890	15889	8482	15200
	$\mathrm{CI}_{\mathrm{lower}}$	6863	15854	8461	15136
	$\mathrm{CI}_{\mathrm{upper}}$	6917	15924	8504	15264
50	Mean	8555	17464	9994	17005
	$\mathrm{CI}_{\mathrm{lower}}$	8530	17426	9968	16937
	$\mathrm{CI}_{\mathrm{upper}}$	8579	17503	10020	17073

Table A.9: *Simulation results: Price*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	75.1802	78.4729	78.9843	79.3291	76.1187
	$\mathrm{CI}_{\mathrm{lower}}$	74.8232	78.3090	78.5947	78.9473	75.7027
	$\mathrm{CI}_{\mathrm{upper}}$	75.5372	78.6368	79.3740	79.7109	76.5346
4	Mean	74.3905	74.3014	74.8320	74.9876	75.5534
	$\mathrm{CI}_{\mathrm{lower}}$	74.2964	73.9341	74.3014	74.4384	74.8687
	$\mathrm{CI}_{\mathrm{upper}}$	74.4847	74.6687	75.3626	75.5368	76.2380
6	Mean	74.2829	71.8377	72.7756	73.5394	74.6170
	$\mathrm{CI}_{\mathrm{lower}}$	74.2177	71.2513	72.2317	72.9559	73.9155
	$\mathrm{CI}_{\mathrm{upper}}$	74.3480	72.4242	73.3196	74.1229	75.3185
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	24.9717	21.4946	22.9352	22.3979	21.8232
	$\mathrm{CI}_{\mathrm{lower}}$	24.8664	21.4253	22.8099	22.2897	21.6279
	$\mathrm{CI}_{\mathrm{upper}}$	25.0770	21.5639	23.0606	22.5061	22.0185
20	Mean	48.9542	45.0496	47.2447	47.4798	45.1252
	$\mathrm{CI}_{\mathrm{lower}}$	48.7906	44.8806	46.9206	47.1556	44.6791
	$\mathrm{CI}_{\mathrm{upper}}$	49.1179	45.2185	47.5689	47.8041	45.5713
30	Mean	74.9350	72.9816	74.3998	75.7842	76.9354
	$\mathrm{CI}_{\mathrm{lower}}$	74.7680	72.6742	73.8850	75.2493	76.2516
	$\mathrm{CI}_{\mathrm{upper}}$	75.1019	73.2890	74.9145	76.3192	77.6191
40	Mean	99.4842	104.1472	103.8546	104.3191	101.8182
	$\mathrm{CI}_{\mathrm{lower}}$	99.2988	103.6245	103.1827	103.5980	100.9530
	$\mathrm{CI}_{\mathrm{upper}}$	99.6696	104.6699	104.5265	105.0402	102.6834
50	Mean	124.7443	130.6804	129.2190	129.7790	131.4464
	$\mathrm{CI}_{\mathrm{lower}}$	124.5050	129.8860	128.4147	128.9434	130.6332
	$\mathrm{CI}_{\mathrm{upper}}$	124.9837	131.4748	130.0233	130.6145	132.2595

Table A.10: *Simulation results: Response Time*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	69130	52412	54528	54345	50176
	$\mathrm{CI}_{\mathrm{lower}}$	68764	52191	54141	53946	49460
	$\mathrm{CI}_{\mathrm{upper}}$	69497	52634	54915	54744	50893
4	Mean	70308	53503	64842	65157	55290
	$\mathrm{CI}_{\mathrm{lower}}$	70201	53081	64295	64529	54485
	$\mathrm{CI}_{\mathrm{upper}}$	70415	53925	65389	65785	56095
6	Mean	71283	65685	69419	69835	58417
	$\mathrm{CI}_{\mathrm{lower}}$	71214	64867	68694	69183	57672
	$\mathrm{CI}_{\mathrm{upper}}$	71352	66503	70144	70487	59162
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	22197	21520	20669	20773	18701
	$\mathrm{CI}_{\mathrm{lower}}$	22072	21458	20539	20611	18296
	$\mathrm{CI}_{\mathrm{upper}}$	22321	21582	20799	20934	19107
20	Mean	46066	37051	39945	40673	32705
	$\mathrm{CI}_{\mathrm{lower}}$	45902	36788	39595	40310	32067
	$\mathrm{CI}_{\mathrm{upper}}$	46230	37314	40296	41036	33342
30	Mean	69896	57979	61695	62302	54564
	$\mathrm{CI}_{\mathrm{lower}}$	69735	57354	60917	61729	53617
	$\mathrm{CI}_{\mathrm{upper}}$	70058	58605	62473	62875	55512
40	Mean	94144	75076	85006	84404	73809
	$\mathrm{CI}_{\mathrm{lower}}$	93930	74450	84310	83601	72875
	$\mathrm{CI}_{\mathrm{upper}}$	94357	75702	85701	85207	74743
50	Mean	118899	94375	107333	107411	93359
	$\mathrm{CI}_{\mathrm{lower}}$	118658	93516	106520	106514	92506
	$\mathrm{CI}_{\mathrm{upper}}$	119140	95234	108145	108308	94212

Table A.11: *Simulation results: Encryption*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	0.8005	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{lower}}$	0.8003	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{upper}}$	0.8006	0.8000	0.8000	0.8000	0.8000
4	Mean	0.8003	0.8000	0.8003	0.8003	0.8004
	$\mathrm{CI}_{\mathrm{lower}}$	0.8003	0.8000	0.8002	0.8002	0.8003
	$\mathrm{CI}_{\mathrm{upper}}$	0.8004	0.8000	0.8004	0.8003	0.8006
6	Mean	0.8005	0.8023	0.8010	0.8016	0.8022
	$\mathrm{CI}_{\mathrm{lower}}$	0.8004	0.8022	0.8008	0.8014	0.8019
	$\mathrm{CI}_{\mathrm{upper}}$	0.8005	0.8024	0.8011	0.8017	0.8025
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	0.8021	0.8039	0.8021	0.8030	0.8044
	$\mathrm{CI}_{\mathrm{lower}}$	0.8017	0.8037	0.8018	0.8027	0.8037
	$\mathrm{CI}_{\mathrm{upper}}$	0.8025	0.8040	0.8024	0.8034	0.8051
20	Mean	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{lower}}$	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{upper}}$	0.8000	0.8000	0.8000	0.8000	0.8000
30	Mean	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{lower}}$	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{upper}}$	0.8000	0.8000	0.8000	0.8000	0.8000
40	Mean	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{lower}}$	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{upper}}$	0.8000	0.8000	0.8000	0.8000	0.8000
50	Mean	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{lower}}$	0.8000	0.8000	0.8000	0.8000	0.8000
	$\mathrm{CI}_{\mathrm{upper}}$	0.8000	0.8000	0.8000	0.8000	0.8000

Table A.12: *Simulation results: Compliance Distance*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	6.7257	2.2432	3.4492	3.9953	0.0017
	CI_{lower}	6.6175	2.2085	3.3769	3.9324	0.0001
	CI_{upper}	6.8339	2.2779	3.5215	4.0581	0.0033
4	Mean	7.4301	1.5050	5.8844	5.5823	0.0035
	CI_{lower}	7.3910	1.4743	5.7625	5.5001	0.0017
	CI_{upper}	7.4691	1.5357	6.0063	5.6646	0.0053
6	Mean	8.9298	1.9778	5.7118	6.5748	0.6558
	CI_{lower}	8.8980	1.9308	5.5993	6.4718	0.6457
	CI_{upper}	8.9617	2.0249	5.8242	6.6778	0.6658
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	1.8987	1.3391	1.8702	1.9226	0.0119
	CI_{lower}	1.8633	1.3250	1.8490	1.9080	0.0089
	CI_{upper}	1.9341	1.3531	1.8914	1.9372	0.0150
20	Mean	4.1042	1.0225	3.2549	3.4442	0.0000
	CI_{lower}	4.0560	0.9996	3.1966	3.4045	0.0000
	CI_{upper}	4.1524	1.0454	3.3132	3.4839	0.0000
30	Mean	7.7532	2.2358	3.4305	5.6364	0.3359
	CI_{lower}	7.6929	2.1901	3.3544	5.5547	0.3339
	CI_{upper}	7.8135	2.2814	3.5066	5.7181	0.3379
40	Mean	11.4063	3.5857	8.1800	7.9098	0.6691
	CI_{lower}	11.3352	3.5367	8.0155	7.7865	0.6654
	CI_{upper}	11.4774	3.6347	8.3446	8.0331	0.6727
50	Mean	13.3136	1.3604	8.3399	8.0076	0.0847
	CI_{lower}	13.2300	1.3045	8.1490	7.8535	0.0710
	CI_{upper}	13.3971	1.4163	8.5309	8.1618	0.0985

Table A.13: *Simulation results: Average DF*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	0.0090	0.0182	0.0178	0.0185	0.0153
	CI_{lower}	0.0088	0.0173	0.0177	0.0184	0.0153
	CI_{upper}	0.0091	0.0191	0.0178	0.0185	0.0153
4	Mean	0.0286	0.0987	0.0645	0.0661	0.0591
	CI_{lower}	0.0280	0.0975	0.0629	0.0645	0.0562
	CI_{upper}	0.0293	0.1000	0.0662	0.0678	0.0621
6	Mean	0.0392	0.1689	0.1135	0.0976	0.0907
	CI_{lower}	0.0389	0.1665	0.1116	0.0963	0.0880
	CI_{upper}	0.0396	0.1713	0.1154	0.0990	0.0934
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	0.0281	0.0460	0.0362	0.0350	0.0320
	CI_{lower}	0.0272	0.0454	0.0351	0.0340	0.0294
	CI_{upper}	0.0291	0.0466	0.0372	0.0360	0.0345
20	Mean	0.0355	0.1377	0.0826	0.0868	0.0902
	CI_{lower}	0.0350	0.1355	0.0815	0.0858	0.0884
	CI_{upper}	0.0359	0.1399	0.0838	0.0879	0.0921
30	Mean	0.0260	0.1261	0.1043	0.0763	0.0674
	CI_{lower}	0.0258	0.1241	0.1024	0.0751	0.0645
	CI_{upper}	0.0263	0.1282	0.1062	0.0775	0.0704
40	Mean	0.0197	0.0950	0.0561	0.0611	0.0387
	CI_{lower}	0.0195	0.0936	0.0551	0.0601	0.0373
	CI_{upper}	0.0199	0.0965	0.0571	0.0621	0.0401
50	Mean	0.0187	0.0716	0.0472	0.0445	0.0470
	CI_{lower}	0.0185	0.0703	0.0463	0.0436	0.0463
	CI_{upper}	0.0189	0.0729	0.0480	0.0453	0.0476

Table A.14: *Simulation results: L2 Norm*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	0.0493	0.1012	0.0989	0.1026	0.0855
	CI_{lower}	0.0486	0.1011	0.0988	0.1025	0.0854
	CI_{upper}	0.0499	0.1014	0.0991	0.1027	0.0855
4	Mean	0.0931	0.2257	0.1925	0.1995	0.1632
	CI_{lower}	0.0924	0.2242	0.1911	0.1980	0.1591
	CI_{upper}	0.0939	0.2272	0.1940	0.2010	0.1673
6	Mean	0.1304	0.4041	0.3100	0.2961	0.2411
	CI_{lower}	0.1297	0.3997	0.3066	0.2937	0.2359
	CI_{upper}	0.1312	0.4084	0.3133	0.2985	0.2463
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	0.1528	0.3117	0.2462	0.2648	0.1999
	CI_{lower}	0.1518	0.3103	0.2447	0.2633	0.1946
	CI_{upper}	0.1538	0.3131	0.2476	0.2662	0.2052
20	Mean	0.0923	0.2474	0.1982	0.2045	0.1818
	CI_{lower}	0.0915	0.2454	0.1967	0.2028	0.1783
	CI_{upper}	0.0931	0.2493	0.1998	0.2063	0.1853
30	Mean	0.0692	0.2279	0.2144	0.1792	0.1309
	CI_{lower}	0.0686	0.2250	0.2117	0.1778	0.1278
	CI_{upper}	0.0697	0.2308	0.2171	0.1806	0.1339
40	Mean	0.0604	0.1991	0.1562	0.1607	0.1188
	CI_{lower}	0.0600	0.1970	0.1548	0.1595	0.1162
	CI_{upper}	0.0607	0.2012	0.1576	0.1620	0.1213
50	Mean	0.0801	0.2323	0.1874	0.1877	0.1849
	CI_{lower}	0.0792	0.2307	0.1862	0.1867	0.1837
	CI_{upper}	0.0810	0.2339	0.1885	0.1888	0.1862

Table A.15: *Simulation results: RMS Norm*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	0.0348	0.0716	0.0699	0.0725	0.0605
	$\mathrm{CI}_{\mathrm{lower}}$	0.0344	0.0715	0.0698	0.0724	0.0604
	$\mathrm{CI}_{\mathrm{upper}}$	0.0353	0.0717	0.0700	0.0726	0.0605
4	Mean	0.0466	0.1129	0.0963	0.0998	0.0816
	$\mathrm{CI}_{\mathrm{lower}}$	0.0462	0.1121	0.0955	0.0990	0.0795
	$\mathrm{CI}_{\mathrm{upper}}$	0.0470	0.1136	0.0970	0.1005	0.0836
6	Mean	0.0533	0.1650	0.1265	0.1209	0.0984
	$\mathrm{CI}_{\mathrm{lower}}$	0.0529	0.1632	0.1252	0.1199	0.0963
	$\mathrm{CI}_{\mathrm{upper}}$	0.0536	0.1667	0.1279	0.1219	0.1006
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	0.0686	0.1374	0.1101	0.1180	0.0888
	$\mathrm{CI}_{\mathrm{lower}}$	0.0681	0.1368	0.1094	0.1173	0.0863
	$\mathrm{CI}_{\mathrm{upper}}$	0.0691	0.1380	0.1107	0.1186	0.0912
20	Mean	0.0447	0.1156	0.0944	0.0975	0.0868
	$\mathrm{CI}_{\mathrm{lower}}$	0.0443	0.1147	0.0937	0.0968	0.0853
	$\mathrm{CI}_{\mathrm{upper}}$	0.0451	0.1165	0.0951	0.0983	0.0883
30	Mean	0.0347	0.1093	0.1021	0.0886	0.0646
	$\mathrm{CI}_{\mathrm{lower}}$	0.0344	0.1081	0.1009	0.0880	0.0631
	$\mathrm{CI}_{\mathrm{upper}}$	0.0350	0.1106	0.1033	0.0892	0.0660
40	Mean	0.0314	0.0984	0.0791	0.0819	0.0600
	$\mathrm{CI}_{\mathrm{lower}}$	0.0312	0.0975	0.0785	0.0813	0.0589
	$\mathrm{CI}_{\mathrm{upper}}$	0.0316	0.0993	0.0798	0.0825	0.0611
50	Mean	0.0450	0.1216	0.1022	0.1026	0.1006
	$\mathrm{CI}_{\mathrm{lower}}$	0.0445	0.1209	0.1016	0.1021	0.1001
	$\mathrm{CI}_{\mathrm{upper}}$	0.0456	0.1223	0.1027	0.1031	0.1012

Table A.16: *Simulation results: $\mathcal{HV}$*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	0.0017	0.0072	0.0069	0.0074	0.0057
	CI_{lower}	0.0017	0.0072	0.0069	0.0074	0.0057
	CI_{upper}	0.0017	0.0072	0.0070	0.0074	0.0057
4	Mean	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{lower}	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{upper}	0.0000	0.0000	0.0000	0.0000	0.0000
6	Mean	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{lower}	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{upper}	0.0000	0.0000	0.0000	0.0000	0.0000
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{lower}	0.0000	0.0000	0.0000	0.0000	0.0000
	CI_{upper}	0.0000	0.0000	0.0000	0.0000	0.0000
20	Mean	0.0003	0.0008	0.0008	0.0008	0.0007
	CI_{lower}	0.0003	0.0008	0.0008	0.0008	0.0007
	CI_{upper}	0.0003	0.0008	0.0008	0.0008	0.0007
30	Mean	0.0003	0.0016	0.0014	0.0016	0.0006
	CI_{lower}	0.0003	0.0016	0.0014	0.0016	0.0006
	CI_{upper}	0.0003	0.0016	0.0014	0.0016	0.0006
40	Mean	0.0004	0.0023	0.0020	0.0022	0.0011
	CI_{lower}	0.0004	0.0023	0.0020	0.0022	0.0011
	CI_{upper}	0.0004	0.0023	0.0020	0.0023	0.0011
50	Mean	0.0018	0.0074	0.0074	0.0076	0.0071
	CI_{lower}	0.0018	0.0074	0.0074	0.0076	0.0071
	CI_{upper}	0.0019	0.0074	0.0074	0.0076	0.0071

Table A.17: *Mean $\mathcal{HR}$ of simulation results*

# Prot. Functions	Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	0.0591	0.2490	0.2397	0.2548	0.1974
4	0.0059	0.5749	0.1083	0.1777	0.1332
6	0.0006	0.6393	0.2535	0.0649	0.0417
Workflow Length	Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	0.1041	0.2459	0.2238	0.2333	0.1929
20	0.0781	0.2354	0.2345	0.2405	0.2115
30	0.0574	0.2838	0.2500	0.2954	0.1134
40	0.0516	0.2835	0.2473	0.2813	0.1364
50	0.0590	0.2364	0.2363	0.2423	0.2260

Table A.18: *Simulation results: Runtime*

# Prot. Functions		Random	IBEA	NSGA-II	SPEA2	COMPAGA
2	Mean	28127	33129	29008	34123	34521
	CI_{lower}	28085	33086	28940	34075	34109
	CI_{upper}	28170	33171	29077	34171	34933
4	Mean	33197	37482	34641	40472	40024
	CI_{lower}	33068	37409	34511	40395	39882
	CI_{upper}	33326	37554	34772	40550	40166
6	Mean	35339	45883	35673	46405	35582
	CI_{lower}	35213	45827	35621	46293	35499
	CI_{upper}	35465	45939	35726	46516	35666
Workflow Length		Random	IBEA	NSGA-II	SPEA2	COMPAGA
10	Mean	10061	17796	12538	23727	12708
	CI_{lower}	10037	17768	12516	23660	12657
	CI_{upper}	10085	17825	12560	23794	12759
20	Mean	23092	27991	20781	28658	21457
	CI_{lower}	22956	27953	20747	28608	21412
	CI_{upper}	23228	28030	20815	28708	21503
30	Mean	31313	38653	35133	38699	31507
	CI_{lower}	31236	38565	35071	38586	31440
	CI_{upper}	31390	38741	35194	38811	31575
40	Mean	42445	49782	45303	49432	42602
	CI_{lower}	42326	49716	45069	49353	42529
	CI_{upper}	42563	49848	45536	49510	42675
50	Mean	54194	59932	51784	61151	75271
	CI_{lower}	54054	59868	51715	61064	74447
	CI_{upper}	54335	59996	51853	61239	76096

Appendix B

Publications

The following is a chronological list of all publications in which the author participated over the course of his PhD studies, divided into publication types. All of these works have been accepted for publication. Except of the entries in sections B.5 and B.6, all works are peer-reviewed.

B.1 Journals

1. F. Karatas, L. Fischer, and D. Kesdogan. “Service Composition with Consideration of Interdependent Security Objectives”. In: *Science of Computer Programming* 97 (2015), pp. 183–201. DOI: 10.1016/j.scico.2014.06.016

B.2 Book Chapters

2. F. Karatas, M. Bourimi, D. Kesdogan, P. G. Villanueva, and H. M. Fardoun. “Evaluating Usability and Privacy in Collaboration Settings with DUIs: Problem Analysis and Case Studies”. In: *Distributed User Interfaces: Usability and Collaboration*. Ed. by M. D. Lozano, J. A. Gallud, R. Tesoriero, and V. M. R. Penichet. Human-Computer Interaction Series. London: Springer-Verlag, 2013. Chap. 10, pp. 119–127. DOI: 10.1007/978-1-4471-5499-0_10

3. R. Tesoriero, M. Bourimi, F. Karatas, T. Barth, P. G. Villanueva, and P. Schwarte. “Model-Driven Privacy and Security in Multi-modal Social Media UIs”. In: *Modeling and Mining Ubiquitous Social Media*. Ed. by M. Atzemueller, A. Chin, D. Helic, and A. Hotho. Vol. 7472. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 158–181. DOI: 10.1007/978-3-642-33684-3_9

B.3 Conferences

4. F. Karatas and D. Kesdogan. "An Approach for Compliance-Aware Service Selection with Genetic Algorithms". In: *Proceedings of the 11th International Conference on Service Oriented Computing*. Ed. by S. Basu, C. Pautasso, L. Zhang, and X. Fu. ICSOC '11. Berlin, Heidelberg: Springer, 2013, pp. 465–473. DOI: `10.1007/978-3-642-45005-1_35`

5. F. Karatas, M. Bourimi, and D. Kesdogan. "Towards Visual Configuration Support for Interdependent Security Goals". In: *Online Communities and Social Computing*. Ed. by A. A. Ozok and P. Zaphiris. Vol. 8029. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 375–384. DOI: `10.1007/978-3-642-39371-6_42`

6. F. Karatas, M. Heupel, M. Bourimi, D. Kesdogan, and S. Wrobel. "Considering Interdependent Protection Goals in Domain-Specific Contexts: The di.me Case Study". In: *Proceedings of the 10th International Conference on Information Technology - New Generations*. ITNG '13. Washington, DC: IEEE, 2013, pp. 27–32. DOI: `10.1109/ITNG.2013.12`

7. F. Karatas and D. Kesdogan. "A Flexible Approach For Considering Interdependent Security Objectives in Service Composition". In: *Proceedings of the 28th Symposium on Applied Computing*. SAC '13. New York, NY, USA: ACM, 2013, pp. 1919–1926. DOI: `10.1145/2480362.2480717`

8. M. Bourimi, M. Heupel, B. Westermann, D. Kesdogan, R. Gimenez, M. Planaguma, F. Karatas, and P. Schwarte. "Towards Transparent Anonymity for User-Controlled Servers Supporting Collaborative Scenarios". In: *2012 Ninth International Conference on Information Technology: New Generations (ITNG)*. Apr. 2012, pp. 102–108. DOI: `10.1109/ITNG.2012.23`

9. J. Gulden, T. Barth, D. Kesdogan, and F. Karatas. "Erhöhung der Sicherheit von Lebensmittelwarenketten durch Modell-getriebene Prozess-Implementierung". In: *Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI) 2012*. 2012, pp. 2061–2072

10. M. Bourimi, R. Tesoriero, P. G. Villanueva, F. Karatas, and P. Schwarte. "Privacy and Security in Multi-modal User Interface Modeling for Social Media". In: *Proceedings of the 2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International conference on Social Computing*. Washington, D. C., Oct. 2011, pp. 1364–1371. DOI: `10.1109/PASSAT/SocialCom.2011.49`

B.4 Workshops

11. M. Bourimi, F. Karatas, P. G. Villanueva, Y. Daanoun, and M. Miran. "Towards Better Support For Collaborative Research By Using DUIs With Mobile Devices: SocialTV Navigation Design Case Study". In: *Proceedings of the 2nd Workshop on Distributed User Interfaces (DUI) in conjunction with ACM CHI 2012*. 2012, pp. 9–12

12. F. Karatas, T. Barth, D. Kesdogan, H. M. Fardoun, and P. G. Villanueva. "Using Distributed User Interfaces to Evaluate Decision Making in Cloud Deployment". In: *Proceedings of the 2nd Workshop on Distributed User Interfaces (DUI) in conjunction with ACM CHI 2012*. 2012, pp. 17–21

13. F. Karatas, M. Bourimi, T. Barth, D. Kesdogan, R. Gimenez, W. Schwittek, and M. Planaguma. "Towards Secure and At-Runtime Tailorable Customer-Driven Public Cloud Deployment". In: *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. Mar. 2012, pp. 124–130. DOI: 10.1109/PerComW.2012.6197463

B.5 Technical Reports

14. M. Bourimi, S. Scerri, M. Planaguma, M. Heupel, F. Karatas, and P. Schwarte. *A Two-Level Approach to Ontology-Based Access Control in Pervasive Personal Servers*. Tech. rep. *urn:nbn:de:hbz:467-5789*. Dec. 2011. URL: http://dokumentix.ub.uni-siegen.de/opus/volltexte/2011/578

B.6 Professional Journals

15. F. Karatas, T. Barth, D. Kesdogan, H. M. Fardoun, and P. G. Villaneava. "Uso de interface distribuída para avaliar a tomada de decisão na nuvem". In: *RTI Magazine, Brazil* (Jan. 2013). URL: http://www.arandanet.com.br/midiaonline/rti/2013/janeiro/index.html

Bibliography

[Aal98] W. M. P. van der AALST. "The Application of Petri Nets to Workflow Management". In: *The Journal of Circuits, Systems and Computers* 8.1 (1998), pp. 21–66. DOI: 10.1142/S0218126698000043.

[ABD05] W. M. P. van der AALST, H. T. de BEER, and B. F. van DONGEN. "Process Mining and Verification of Properties: An Approach Based on Temporal Logic". In: *Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems - Part I*. OTM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 130–147. DOI: 10.1007/11575771_11.

[ACD+07] A. ANDRIEUX, K. CZAJKOWSKI, A. DAN, K. KEAHEY, H. LUDWIG, T. NAKATA, J. PRUYNE, J. ROFRANO, S. TUECKE, and M. XU. *Web Services Agreement Specification (WS-Agreement)*. Accessed 27 December 2015. Mar. 2007. URL: https://www.ogf.org/documents/GFD.107.pdf.

[ACG+08] M. ALBERTI, F. CHESANI, M. GAVANELLI, E. LAMMA, P. MELLO, M. MONTALI, and P. TORRONI. "Expressing and Verifying Business Contracts with Abductive Logic Programming". In: *International Journal of Electronic Commerce* 12.4 (July 2008), pp. 9–38. DOI: 10.2753/JEC1086-4415120401.

[ACP+11] W. ARSAC, L. COMPAGNA, G. PELLEGRINO, and S. E. PONTA. "Security Validation of Business Processes via Model-checking". In: *Proceedings of the Third International Conference on Engineering Secure Software and Systems*. ESSoS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 29–42. DOI: 10.1007/978-3-642-19125-1_3.

[ADW08] A. AWAD, G. DECKER, and M. WESKE. "Efficient Compliance Checking Using BPMN-Q and Temporal Logic". In: *Proceedings of the 6th International Conference on Business Process Management*. Ed. by M. Dumas, M. Reichert, and M.-C. Shan. BPM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 326–341. DOI: 10.1007/978-3-540-85758-7_24.

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing*. Tech. rep. UCB/EECS-2009-28. Accessed 15 September 2009. EECS Department, University of California, Berkeley, Feb. 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[AH02a] W. M. P. van der Aalst and K. van Hee. *Workflow Management: Models, Methods and Systems*. Cambridge, MA: MIT Press, 2002.

[AH02b] W. M. P. van der Aalst and A. H. M. ter Hofstede. *YAWL: Yet Another Workflow Language*. Tech. rep. FIT-TR-2002-06. Accessed 27 December 2015. Queensland University of Technology, 2002. URL: http://www.yawlfoundation.org/sites/default/files/yawl.pdf.

[AHK+03] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. "Workflow Patterns". In: *Distributed and Parallel Databases* 14.1 (2003), pp. 5–51. DOI: 10.1023/A:1022883727209.

[AHW03] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. "Business Process Management: A Survey". In: *Proceedings of the 2003 International Conference on Business Process Management*. BPM'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 1–12. DOI: 10.1007/3-540-44895-0_1.

[AJK+06] R. Agrawal, C. Johnson, J. Kiernan, and F. Leymann. "Taming Compliance with Sarbanes-Oxley Internal Controls Using Database Technology". In: *Proceedings of the 22nd International Conference on Data Engineering*. ICDE '06. Washington, DC: IEEE Computer Society, 2006, p. 92. DOI: 10.1109/ICDE.2006.155.

[AL08] W. M. P. van der Aalst and K. B. Lassen. "Translating Unstructured Workflow Processes to Readable BPEL: Theory and Implementation". In: *Information and Software Technology* 50.3 (2008), pp. 131–159. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2006.11.004.

[Ala92] J. T. Alander. "On Optimal Population Size of Genetic Algorithms". In: *Proceedings of the 1992 IEEE International Conference on Computer Systems and Software Engineering*. CompEuro '92. 1992, pp. 65–70. DOI: 10.1109/CMPEUR.1992.218485.

[ALR+04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Jan. 2004), pp. 11–33. doi: 10.1109/TDSC.2004.2.

[ALS11] R. Accorsi, L. Lowis, and Y. Sato. "Automated Certification for Compliant Cloud-based Business Processes". In: *Business & Information Systems Engineering* 3.3 (2011), pp. 145–154. doi: 10.1007/s12599-011-0155-7.

[AM10] D. Ardagna and R. Mirandola. "Per-Flow Optimal Service Selection for Web Services Based Processes". In: *The Journal of Systems and Software* 83.8 (Aug. 2010), pp. 1512–1523. issn: 0164-1212. doi: 10.1016/j.jss.2010.03.045.

[AMS+01] M. M. Akbar, E. G. Manning, G. C. Shoja, and S. Khan. "Heuristic Solutions for the Multiple-Choice Multi-dimension Knapsack Problem". In: *Proceedings of the International Conference on Computational Science*. Ed. by V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan. ICCS '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 659–668. doi: 10.1007/3-540-45718-6_71.

[And08] R. Anderson. *Security Engineering – A Guide to Building Dependable Distributed Systems*. 2nd. Indianapolis, IN: Wiley & Sons, 2008.

[AP06] D. Ardagna and B. Pernici. "Global and Local QoS Guarantee in Web Service Selection". In: *Proceedings of the Third International Conference on Business Process Management*. Ed. by C. J. Bussler and A. Haller. BPM'05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 32–46. doi: 10.1007/11678564_4.

[AP07] D. Ardagna and B. Pernici. "Adaptive Service Composition in Flexible Processes". In: *IEEE Transactions on Software Engineering* 33.6 (June 2007), pp. 369–384. doi: 10.1109/TSE.2007.1011.

[AR09] M. Alrifai and T. Risse. "Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. New York, NY: ACM, 2009, pp. 881–890. doi: 10.1145/1526709.1526828.

[ARK+06] M. M. Akbar, M. S. Rahman, M. Kaykobad, E. G. Manning, and G. C. Shoja. "Solving the Multidimensional Multiple-choice Knapsack Problem by Constructing Convex Hulls". In: *Computers & Operations Research* 33.5 (2006), pp. 1259–1273. doi: 10.1016/j.cor.2004.09.016.

[ARN12] M. Alrifai, T. Risse, and W. Nejdl. “A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints”. In: *ACM Transactions on the Web* 6.2 (June 2012), 7:1–7:31. DOI: 10.1145/2180861.2180864.

[AVM+04] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. “Constraint Driven Web Service Composition in METEOR-S”. In: *Proceedings of the 2004 IEEE International Conference on Services Computing*. SCC '04. Washington, DC: IEEE Computer Society, 2004, pp. 23–30. DOI: 10.1109/SCC.2004.1357986.

[AZA+05] I. B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko. “Ontology-Driven Web Services Composition Platform”. In: *Information Systems and e-Business Management* 3.2 (2005), pp. 175–199. DOI: 10.1007/s10257-005-0055-9.

[BAC+11] J. Becker, C. Ahrendt, A. Coners, B. Weiss, and A. Winkelmann. “Modeling and Analysis of Business Process Compliance”. In: *Governance and Sustainability in Information Systems. Managing the Transfer and Diffusion of IT*. Ed. by M. Nüttgens, A. Gadatsch, K. Kautz, I. Schirmer, and N. Blinn. Vol. 366. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2011, pp. 259–269. DOI: 10.1007/978-3-642-24148-2_17.

[Bäc96] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford: Oxford University Press, Inc., 1996.

[Bal04] Y. Balzer. *Improve your SOA Project Plans*. http://www.ibm.com/developerworks/webservices/library/ws-improvesoa/. Accessed 11 September 2013. July 2004.

[BBB+12] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. *Recommendation for Key Management - Part 1: General (Revision 3)*. Special Publication 800-57. Accessed 27 December 2015. July 2012. URL: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[BBD+11] J. Becker, P. Bergener, P. Delfmann, M. Eggert, and Weiss. “Supporting Business Process Compliance in Financial Institutions – A Model-Driven Approach”. In: *Proceedings of the 10th International Conference on Wirtschaftsinformatik*. 2011, pp. 355–364.

[BBG+07] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini. “Validation of Web Service Compositions”. In: *Software, IET* 1.6 (2007), pp. 219–232. DOI: 10.1049/iet-sen:20070027.

[BBG05] R. Bhatti, E. Bertino, and A. Ghafoor. "A Trust-Based Context-Aware Access Control Model for Web-Services". In: *Distributed and Parallel Databases* 18.1 (2005), pp. 83–105. DOI: 10.1007/s10619-005-1075-7.

[BBH+10] M. Bourimi, T. Barth, J. M. Haake, B. Ueberschär, and D. Kesdogan. "AFFINE for Enforcing Earlier Consideration of NFRs and Human Factors when Building Socio-technical Systems Following Agile Methodologies". In: *Proceedings of the Third International Conference on Human-centred Software Engineering*. HCSE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 182–189. DOI: 10.1007/978-3-642-16488-0_15.

[BBM+08] G. Bernardi, M. Bugliesi, D. Macedonio, and S. Rossi. "A Theory of Adaptable Contract-Based Service Composition". In: *Proceedings of the 2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. SYNASC '08. Washington, DC: IEEE Computer Society, 2008, pp. 327–334. DOI: 10.1109/SYNASC.2008.38.

[BCC+04] T. Bellwood, S. Capell, L. Clement, J. Colgrave, M. J. Dovey, D. Feygin, A. Hately, R. Kochman, P. Macias, M. Novotny, M. Paolucci, C. von Riegen, T. Rogers, K. Sycara, P. Wenzel, and Z. Wu. *UDDI Version 3.0.2*. Accessed 16 September 2013. Oct. 2004. URL: http://uddi.org/pubs/uddi_v3.htm.

[BCH+10] R. L. Baskerville, M. Cavallari, K. Hjort-Madsen, and J. Pries-Heje. "The Strategic Value of SOA: A Comparative Case Study in the Banking Sector". In: *International Journal of Information Technology and Management* 9.1 (Nov. 2010), pp. 30–53. DOI: 10.1504/IJITM.2010.029433.

[BDD+13] S. Bräuer, P. Delfmann, H.-A. Dietrich, and M. Steinhorst. "Using a Generic Model Query Approach to Allow for Process Model Compliance Checking – An Algorithmic Perspective". In: *Proceedings of the 11tth International Conference on Wirtschaftsinformatik*. Ed. by R. Alt and B. Franczyk. WI2013. 2013, pp. 1245–1259.

[BDF06] M. Bartoletti, P. Degano, and G. L. Ferrari. "Security Issues in Service Composition". In: *Proceedings of the 8th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems*. FMOODS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 1–16. DOI: 10.1007/11768869_1.

[BDM+08] J. Branke, K. Deb, K. Miettinen, and R. Słowiński, eds. *Multiobjective Optimization – Interactive and Evolutionary Approaches*. Vol. 5252. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008. doi: 10.1007/978-3-540-88908-3.

[BDS08] D. Benslimane, S. Dustdar, and A. Sheth. "Services Mashups: The New Generation of Web Applications". In: *IEEE Internet Computing* 12.5 (2008), pp. 13–15. doi: 10.1109/MIC.2008.110.

[BEK+00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winter. *Simple Object Access Protocol (SOAP) 1.1*. Accessed 15 September 2013. May 2000. url: http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[Ber02] S. Berezin. "Model Checking and Theorem Proving: a Unified Framework". Accessed 27 December 2015. PhD thesis. Carnegie Mellon University, 2002. url: http://reports-archive.adm.cs.cmu.edu/anon/2002/CMU-CS-02-100.pdf.

[Ber07] R. Berbner. "Dienstgüteunterstützung für Service-orientierte Workflows". Accessed 27 December 2015. PhD thesis. Universität Darmstadt, Germany, 2007. url: http://elib.tu-darmstadt.de/diss/000838.

[BFG+08] K. Bhargavan, C. Fournet, A. D. Gordon, and S. Tse. "Verified Interoperable Implementations of Security Protocols". In: *ACM Transactions on Programming Languages and Systems* 31.1 (Dec. 2008), 5:1–5:61. doi: 10.1145/1452044.1452049.

[BGK04] E. K. Burke, S. Gustafson, and G. Kendall. "Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness". In: *IEEE Transactions on Evolutionary Computation* 8.1 (Feb. 2004), pp. 47–62. doi: 10.1109/TEVC.2003.819263.

[BHW+12] M. Bourimi, M. Heupel, B. Westermann, D. Kesdogan, R. Gimenez, M. Planaguma, F. Karatas, and P. Schwarte. "Towards Transparent Anonymity for User-Controlled Servers Supporting Collaborative Scenarios". In: *2012 Ninth International Conference on Information Technology: New Generations (ITNG)*. Apr. 2012, pp. 102–108. doi: 10.1109/ITNG.2012.23.

[Bib75] K. J. Biba. *Integrity Consideration for Secure Computer Systems*. Tech. rep. MTR-3153. Accessed 13 January 2015. MITRE Corporation, June 1975. url: http://seclab.cs.ucdavis.edu/projects/history/papers/biba75.pdf.

[Bis05] M. BISHOP. *Introduction to Computer Security*. Boston, MA: Addison Wesley, 2005.

[Bis09] J. BISKUP. *Security in Computing Systems - Challenges, Approaches and Solutions*. Berlin, Heidelberg: Springer, 2009.

[BJL+01] F. BESSON, T. JENSEN, D. LE MÉTAYER, and T. THORN. "Model Checking Security Properties of Control Flow Graphs". In: *Journal of Computer Security* 9.3 (Jan. 2001), pp. 217–250.

[BK02] S. BOSWORTH and M. E. KABAY, eds. *Computer Security Handbook*. 4th ed. Hoboken, NJ: Wiley & Sons, 2002.

[BK05] E. K. BURKE and G. KENDALL, eds. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Berlin, Heidelberg: Springer, 2005.

[BKN+11] C. BAUN, M. KUNZE, J. NIMIS, and S. TAI. *Cloud Computing: Webbasierte dynamische IT-Services*. Informatik im Fokus. Berlin, Heidelberg: Springer, 2011. DOI: 10.1007/978-3-642-18436-9.

[BKV+12] M. BOURIMI, F. KARATAS, P. G. VILLANUEVA, Y. DAANOUN, and M. MIRAN. "Towards Better Support For Collaborative Research By Using DUIs With Mobile Devices: SocialTV Navigation Design Case Study". In: *Proceedings of the 2nd Workshop on Distributed User Interfaces (DUI) in conjunction with ACM CHI 2012*. 2012, pp. 9–12.

[BL73] D. E. BELL and L. J. LAPADULA. *Secure Computer Systems: Mathematical Foundations*. Tech. rep. MTR-2547, Vol. I. Accessed 13 January 2015. MITRE Corporation, Nov. 1973. URL: www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0770768.

[Bla01] B. BLANCHET. "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules". In: *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*. CSFW '01. Washington, DC: IEEE Computer Society, 2001, pp. 82–96. DOI: 10.1109/CSFW.2001.930138.

[Bla09] B. BLAU. "Coordination in Service Value Networks - A Mechanism Design Approach". Accessed 27 December 2015. PhD thesis. Fakultät für Wirtschaftswissenschaften, Universität Karlsruhe (TH), 2009. URL: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000012353.

[BLN+09] J. BLANTON, S. LESKI, B. NICKS, and T. TIRZAMAN. “Making SOA Work in a Healthcare Company”. In: *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. OOPSLA ’09. New York, NY: ACM, 2009, pp. 589–596. DOI: 10.1145/1639950.1639953.

[BMV05] D. BASIN, S. MÖDERSHEIM, and L. VIGANÒ. “OFMC: A Symbolic Model Checker for Security Protocols”. In: *International Journal of Information Security* 4.3 (2005), pp. 181–208. DOI: 10.1007/s10207-004-0055-7.

[Bra06] M. BRAMBILLA. “Generation of WebML Web Application Models from Business Process Specifications”. In: *Proceedings of the 6th International Conference on Web Engineering*. ICWE ’06. New York, NY: ACM, 2006, pp. 85–86. DOI: 10.1145/1145581.1145597.

[Bra07] S. BRAHE. “BPM on Top of SOA: Experiences from the Financial Industry”. In: *Proceedings of the 5th International Conference on Business Process Management*. BPM’07. Springer-Verlag. Berlin, Heidelberg, 2007, pp. 96–111. DOI: 10.1007/978-3-540-75183-0_8.

[Bro95] F. P. BROOKS Jr. *The Mythical Man-Month: Essays on Software Engineering*. Anniversary Edition. Boston, MA: Addison Wesley, 1995.

[BS06] J. BLOOMBERG and R. SCHMELZER. *Service Orient or Be Doomed!: How Service Orientation Will Change Your Business*. Hoboken, NJ: Wiley & Sons, Inc., 2006.

[BSD14] A. BOUGUETTAYA, Q. Z. SHENG, and F. DANIEL, eds. *Web Services Foundations*. Berlin, Heidelberg: Springer, 2014. DOI: 10.1007/978-1-4614-7518-7.

[BSI08] BSI. *BSI-Standard 100-3 - Risikoanalyse auf der Basis von IT-Grundschutz*. Accessed 22 September 2014. 2008. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1003_pdf.pdf?__blob=publicationFile.

[BSP+11] M. BOURIMI, S. SCERRI, M. PLANAGUMA, M. HEUPEL, F. KARATAS, and P. SCHWARTE. *A Two-Level Approach to Ontology-Based Access Control in Pervasive Personal Servers*. Tech. rep. *urn:nbn:de:hbz:467-5789*. Dec. 2011. URL: http://dokumentix.ub.uni-siegen.de/opus/volltexte/2011/578.

[BSR+06] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. "Heuristics for QoS-aware Web Service Composition". In: *Proceedings of the IEEE International Conference on Web Services*. ICWS '06. Washington, DC: IEEE Computer Society, 2006, pp. 72–82. doi: 10.1109/ICWS.2006.69.

[BSR+07] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. "Dynamic Replanning of Web Service Workflows". In: *Proceedings of the Inaugural IEEE International Conference on Digital Ecosystems and Technologies*. IEEE DEST '07. 2007, pp. 211–216. doi: 10.1109/DEST.2007.371972.

[BTV+11] M. Bourimi, R. Tesoriero, P. G. Villanueva, F. Karatas, and P. Schwarte. "Privacy and Security in Multi-modal User Interface Modeling for Social Media". In: *Proceedings of the 2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International conference on Social Computing*. Washington, D. C., Oct. 2011, pp. 1364–1371. doi: 10.1109/PASSAT/SocialCom.2011.49.

[Bun13a] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. Tech. rep. BSI TR-02102. Accessed 25 April 2013. Jan. 2013. url: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=1.

[Bun13b] Bundesnetzagentur (BNetzA). *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung*. Accessed 27 December 2015. Feb. 2013. url: https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/2013Algorithmenkatalog.pdf?__blob=publicationFile&v=1.

[Bun90] Bundestag. *Bundesdatenschutzgesetz (BDSG)*. Accessed 5 October 2013. Dec. 1990. url: http://www.gesetze-im-internet.de/bundesrecht/bdsg_1990/gesamt.pdf.

[Bur00] S. Burbeck. *The Tao of E-Business Services*. http://www.ibm.com/developerworks/webservices/library/ws-tao/. Accessed 25 September 2013. Oct. 2000.

[BYL+11] A. Bouguettaya, Q. Yu, X. Liu, and Z. Malik. "Service-Centric Framework for a Digital Government Application". In: *IEEE Transactions on Services Computing* 4.1 (2011), pp. 3–16. doi: 10.1109/TSC.2010.36.

[BZ07] M. BRAVETTI and G. ZAVATTARO. "A Theory for Strong Service Compliance". In: *Proceedings of the 9th International Conference on Coordination Models and Languages*. COORDINATION'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 96–112. DOI: `10.1007/978-3-540-72794-1_6`.

[CAH+97] D. CAMINER, J. ARIS, P. HERMON, and F. LAND. *L.E.O.: The Incredible Story of the World's First Business Computer*. McGraw-Hill, 1997.

[CAH05] D. B. CLARO, P. ALBERS, and J.-K. HAO. "Selecting Web Services for Optimal Composition". In: *Proceedings of the 2nd International Workshop on Semantic and Dynamic Web Processes*. 2005.

[Can01] R. CANETTI. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. FOCS '01. Washington, DC: IEEE Computer Society, 2001, pp. 136–145. DOI: `10.1109/SFCS.2001.959888`.

[Car02] A. J. S. CARDOSO. "Quality of Service and Semantic Composition of Workflows". Accessed 2 July 2013. PhD thesis. The University of Georgia, Aug. 2002. URL: `http://hdl.handle.net/10724/6222`.

[Car05] N. G. CARR. "The End of Corporate Computing". In: *MIT Sloan Management Review* 46.3 (2005), pp. 67–73.

[CCG+06] V. CARDELLINI, E. CASALICCHIO, V. GRASSI, and R. MIRANDOLA. "A Framework for Optimal Service Selection in Broker-Based Architectures with Multiple QoS Classes". In: *Proceedings of the IEEE Services Computing Workshops*. SCW '06. Washington, DC: IEEE Computer Society, 2006, pp. 105–112. DOI: `10.1109/SCW.2006.1`.

[CCG+07] V. CARDELLINI, E. CASALICCHIO, V. GRASSI, and F. LO PRESTI. "Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes". In: *IEEE International Conference on Web Services*. ICWS '07. 2007, pp. 743–750. DOI: `10.1109/ICWS.2007.91`.

[CCM+01] E. CHRISTENSEN, F. CURBERA, G. MEREDITH, and S. WEERAWARANA. *Web Services Description Language (WSDL) 1.1*. Accessed 15 September 2013. Mar. 2001. URL: `http://www.w3.org/TR/wsdl`.

[CDE+05] G. CANFORA, M. DI PENTA, R. ESPOSITO, and M. L. VILLANI. "An Approach for QoS-aware Service Composition based on Genetic Algorithms". In: *Proceedings of the 2005 conference on Genetic and* *evolutionary computation (GECCO)*. 2005, pp. 1069–1075. DOI: 10.1145/1068009.1068189.

[CDE+08] G. CANFORA, M. DI PENTA, R. ESPOSITO, and M. L. VILLANI. "A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services". In: *The Journal of Systems and Software* 81.10 (Oct. 2008), pp. 1754–1769. DOI: 10.1016/j.jss.2007.12.792.

[CES83] E. M. CLARKE, E. A. EMERSON, and A. P. SISTLA. "Automatic Verification of Finite State Concurrent System Using Temporal Logic Specifications: A Practical Approach". In: *Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '83. New York, NY: ACM, 1983, pp. 117–126. DOI: 10.1145/567067.567080.

[CFH06] B. CARMINATI, E. FERRARI, and P. C. K. HUNG. "Security Conscious Web Service Composition". In: *Proceedings of the IEEE International Conference on Web Services*. ICWS '06. Washington, DC: IEEE Computer Society, 2006, pp. 489–496. DOI: 10.1109/ICWS.2006.115.

[CHM+12] I. CRÉVITS, S. HANAFI, R. MANSI, and C. WILBAUT. "Iterative Semi-Continuous Relaxation Heuristics for the Multiple-Choice Multidimensional Knapsack Problem". In: *Computers & Operations Research* 39.1 (2012), pp. 32–41. DOI: 10.1016/j.cor.2010.12.016.

[CKO92] B. CURTIS, M. I. KELLNER, and J. OVER. "Process Modeling". In: *Communications of the ACM* 35.9 (Sept. 1992), pp. 75–90. DOI: 10.1145/130994.130998.

[Cla08] E. M. CLARKE. "The Birth of Model Checking". In: *25 Years of Model Checking*. Ed. by O. Grumberg and H. Veith. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–26. DOI: 10.1007/978-3-540-69850-0_1.

[CLR+09] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, and C. STEIN. *Introduction to Algorithms*. 3rd ed. Cambridge, MA: The MIT Press, 2009.

[CLV07] C. A. COELLO COELLO, G. B. LAMONT, and D. A. VAN VELDHUIZEN. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Ed. by D. E. Goldberg and J. R. Koza. 2nd. Genetic and Evolutionary Computation Series. Springer, 2007. DOI: 10.1007/978-0-387-36797-2.

[Cor08] G. CORNUÉJOLS. "Valid Inequalities for Mixed Integer Linear Programs". In: *Mathematical Programming* 112.1 (Mar. 2008), pp. 3–44. DOI: 10.1007/s10107-006-0086-0.

[CP09] M. COMUZZI and B. PERNICI. “A Framework for QoS-Based Web Service Contracting”. In: *ACM Transactions on the Web* 3.3 (July 2009), 10:1–10:52. DOI: 10.1145/1541822.1541825.

[CRR10] C. CABANILLAS, M. RESINAS, and A. RUIZ-CORTES. “III Taller de Procesos de Negocio e Ingenieria de Servicios PNIS10 in JISBD10”. In: vol. 4. 4. 2010. Chap. Hints on how to face business process compliance, pp. 26–32.

[CSN+13] A. CAMERON, M. STUMPTNER, N. NANDAGOPAL, W. MAYER, and T. MANSELL. “Performance Analysis of a Rule-based SOA Component for Real-time Applications”. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC ’13. New York, NY, USA: ACM, 2013, pp. 1877–1884. DOI: 10.1145/2480362.2480711.

[CW87] D. D. CLARK and D. R. WILSON. “A Comparison of Commercial and Military Computer Security Policies”. In: *1987 IEEE Symposium on Security and Privacy* (1987). DOI: 10.1109/SP.1987.10001.

[DD07] W. DOMSCHKE and A. DREXL. *Einführung in Operations Research*. 7th ed. Berlin, Heidelberg: Springer, 2007.

[Deb01] K. DEB. *Multi-Objective Optimization using Evolutionary Algorithms*. New York, NY: Wiley & Sons, 2001.

[Den82] D. E. R. DENNING. *Cryptography and Data Security*. Boston: Addison Wesley, 1982.

[DG92] J.-C. DELAUNAY and J. GADREY. *Services in Economic Thought: Three Centuries of Debate*. Kluwer Academic Publishers, 1992. DOI: 10.1007/978-94-011-2960-2.

[DGM+12] D. D’APRILE, L. GIORDANO, A. MARTELLI, G. L. POZZATO, D. ROGNONE, and D. T. DUPRÉ. “Business Process Compliance Verification: An Annotation Based Approach with Commitments”. In: *Information Systems: Crossroads for Organization, Management, Accounting and Engineering*. Ed. by M. De Marco, D. Te’eni, V. Albano, and S. Za. Physica-Verlag HD, 2012, pp. 563–570. DOI: 10.1007/978-3-7908-2789-7_61.

[DHS05] Á. DARVAS, R. HÄHNLE, and D. SANDS. “A Theorem Proving Approach to Analysis of Secure Information Flow”. In: *Proceedings of the Second International Conference on Security in Pervasive Computing*. Ed. by D. Hutter and M. Ullmann. SPC’05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 193–209. DOI: 10.1007/978-3-540-32004-3_20.

[Dij59] E. W. DIJKSTRA. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/BF01386390.

[Dij72] E. W. DIJKSTRA. "The Humble Programmer". In: *Communications of the ACM* 15.10 (Oct. 1972), pp. 859–866. DOI: 10.1145/355604.361591.

[Dij75] E. W. DIJKSTRA. "Guarded Commands, Nondeterminacy and Formal Derivation of Programs". In: *Communications of the ACM* 18.8 (Aug. 1975), pp. 453–457. DOI: 10.1145/360933.360975.

[Dij82] E. W. DIJKSTRA. "On the Role of Scientific Thought". In: *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982, pp. 60–66.

[DK75] F. DEREMER and H. KRON. "Programming-in-the-Large Versus Programming-in-the-Small". In: *Proceedings of the International Conference on Reliable Software*. New York, NY: ACM, 1975, pp. 114–121. DOI: 10.1145/800027.808431.

[DKL+07] G. DECKER, O. KOPP, F. LEYMANN, and M. WESKE. "BPEL4Chor: Extending BPEL for Modeling Choreographies". In: *Proceedings of the 2007 IEEE International Conference on Web Services*. ICWS '07. 2007, pp. 296–303. DOI: 10.1109/ICWS.2007.59.

[DN11] J. J. DURILLO and A. J. NEBRO. "jMetal: A Java framework for multi-objective optimization". In: *Advances in Engineering Software* 42.10 (2011), pp. 760–771. DOI: 10.1016/j.advengsoft.2011.05.014.

[DPA+02] K. DEB, A. PRATAB, S. AGARWAL, and T. MEYARIVAN. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. DOI: 10.1109/4235.996017.

[DS97] B. DUTERTRE and S. SCHNEIDER. "Using a PVS Embedding of CSP to Verify Authentication Protocols". In: *Theorem Proving in Higher Order Logics*. Ed. by E. L. Gunter and A. Felty. Vol. 1275. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1997, pp. 121–136. DOI: 10.1007/BFb0028390.

[DY81] D. DOLEV and A. C. YAO. "On the Security of Public Key Protocols". In: *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. SFCS '81. Washington, DC: IEEE Computer Society, 1981, pp. 350–357. DOI: 10.1109/SFCS.1981.32.

[Eck12] C. ECKERT. *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. 7th ed. Munich: Oldenbourg Verlag, 2012.

[EMS+08] M. EL KHARBILI, A. K. A. de MEDEIROS, S. STEIN, and W. M. P. van der AALST. "Business Process Compliance Checking: Current State and Future Challenges". In: *MobIS*. Ed. by P. Loos, M. Nüttgens, K. Turowski, and D. Werth. Vol. 141. LNI. GI, 2008, pp. 107–113.

[Erl05] T. ERL. *Service-Oriented Architecture; Concepts, Technology, and Design*. Upper Saddle River, NJ: Prentice Hall, 2005.

[Erl07] T. ERL. *SOA: Principles of Service Design*. Upper Saddle River, NJ: Prentice Hall, 2007.

[FF93] C. M. FONSECA and P. J. FLEMING. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization". In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1993, pp. 416–423.

[FFH+03] C. FACCIORUSSO, S. FIELD, R. HAUSER, Y. HOFFNER, R. HUMBEL, R. PAWLITZEK, W. RJAIBI, and C. SIMINITZ. "A Web Services Matchmaking Engine for Web Services". In: *Proceedings of the 4th International Conference on E-Commerce and Web Technologies*. Ed. by K. Bauknecht, A. M. Tjoa, and G. Quirchmayr. EC-Web '03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 37–49. DOI: 10.1007/978-3-540-45229-4_5.

[FGM+99] R. FIELDING, J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH, and T. BERNERS-LEE. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Accessed 27 December 2015. June 1999. URL: http://tools.ietf.org/html/rfc2616.

[FHJ+74] G. J. FEENEY, R. D. HILTON, R. L. JOHNSON, T. J. O'ROURKE, and T. E. KURTZ. "Utility Computing: A Superior Alternative?" In: *Proceedings of the National Computer Conference, 1974*. AFIPS '74. New York, NY: ACM, 1974, pp. 1003–1004. DOI: 10.1145/1500175.1500370.

[Fie00] R. FIELDING. "Architectural Styles and the Design of Network-based Software Architectures". Accessed 14 September 2013. PhD thesis. University of California, Irvine, 2000. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[Fra06] U. FRANK. *Towards a pluralistic conception of research methods in information systems research.* ICB Research Report 7. Accessed 27 December 2015. Dec. 2006. URL: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReport07.pdf.

[Fra07] U. FRANK. "Ein Vorschlag zur Konfiguration von Forschungsmethoden in der Wirtschaftsinformatik". In: *Wissenschaftstheoretische Fundierung nud wissenschaftliche Orientierung der Wirtschaftsinformatik.* Ed. by F. Lehner and S. Zelewski. Berlin: GITO-Verlag, 2007, pp. 158–185.

[Fra98] U. FRANK. *The MEMO Object Modelling Language (MEMO-OML).* Tech. rep. Accessed 29 July 2013. D-56075 Koblenz: Institut für Wirtschaftsinformatik, Universität Koblenz-Landau, June 1998. URL: http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Arbeitsberichte_Koblenz/Nr10.pdf.

[Gad00] J. GADREY. "The Characterization of Goods and Services: An Alternative Approach". In: *Review of Income and Wealth* 46.3 (2000), pp. 369–387.

[Gar03] T. GARDNER. "UML Modelling of Automated Business Processes with a Mapping to BPEL4WS". In: *Proceedings of the First European Workshop on Object Orientation and Web Services at ECOOP.* 2003.

[GBK+12] J. GULDEN, T. BARTH, D. KESDOGAN, and F. KARATAS. "Erhöhung der Sicherheit von Lebensmittelwarenketten durch Modellgetriebene Prozess-Implementierung". In: *Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI) 2012.* 2012, pp. 2061–2072.

[GG13] H. GARAVEL and S. GRAF. *Formal Methods for Safe and Secure Computers Systems.* Tech. rep. BSI Study 875. Accessed 27 December 2015. Bonn: Federal Office for Information Security, 2013. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/formal_methods_study_875/formal_methods_study_875.pdf?__blob=publicationFile&v=1.

[GHS+09] G. GOVERNATORI, J. HOFFMANN, S. SADIQ, and I. WEBER. "Detecting Regulatory Compliance for Business Process Models through Semantic Annotations". In: *Business Process Management Workshops.* Ed. by D. Ardagna, M. Mecella, and J. Yang. Vol. 17. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 5–17. DOI: 10.1007/978-3-642-00328-8_2.

[GJ05] R. Grønmo and M. C. Jaeger. "Model-Driven Methodology for Building QoS-Optimised Web Service Compositions". In: *Proceedings of the 5th IFIP WG 6.1 international conference on Distributed Applications and Interoperable Systems*. Ed. by L. Kutvonen and N. Alonistioti. DAIS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 68–82. DOI: 10.1007/11498094_7.

[GK07] A. Ghose and G. Koliadis. "Auditing Business Process Compliance". In: *Proceedings of the 5th International Conference on Service-Oriented Computing*. ICSOC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 169–180. DOI: 10.1007/978-3-540-74974-5_14.

[GLG+06] C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. "SOCK: A Calculus for Service Oriented Computing". In: *Proceedings of the 4th International Conference on Service-Oriented Computing*. ICSOC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 327–338. DOI: 10.1007/11948148_27.

[GMP06] C. Giblin, S. Müller, and B. Pfitzmann. *From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation*. Tech. rep. RZ 3662. IBM Research GmbH, Oct. 2006.

[Gov93] Government of Canada. *The Canadian Trusted Computer Product Evaluation Criteria*. Jan. 1993.

[GPV+96] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. "Simple on-the-fly Automatic Verification of Linear Temporal Logic". In: *Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV*. London: Chapman & Hall, Ltd., 1996, pp. 3–18.

[GR11] T. Ghasemi and M. Razzazi. "Development of Core to Solve the Multidimensional Multiple-Choice Knapsack Problem". In: *Computers & Industrial Engineering* 60.2 (2011), pp. 349–360. DOI: 10.1016/j.cie.2010.12.001.

[GR95] B. Guttman and E. A. Roback. *An Introduction to Computer Security: the NIST Handbook*. Tech. rep. SP 800-12. Gaithersburg, MD: National Institute of Standards & Technology, 1995.

[Gro09] R. L. Grossman. "The Case for Cloud Computing". In: *IT Professional* 11.2 (Mar. 2009), pp. 23–27. DOI: 10.1109/MITP.2009.40.

[GV06] S. Goedertier and J. Vanthienen. "Designing Compliant Business Processes with Obligations and Permissions". In: *Proceedings of the 2006 International Conference on Business Process Management Work-*

shops. BPM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 5–14. DOI: 10.1007/11837862_2.

[HC01] G. T. HEINEMAN and W. T. COUNCILL. *Component-Based Software Engineering: Putting the Pieces Together*. Boston, MA: Addison-Wesley Professional, 2001.

[HHA04] A. HUNSTAD, J. HALLBERG, and R. ANDERSSON. "Measuring IT Security – A Method Based on Common Criteria's Security Functional Requirements". In: *Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop*. Washington, DC: IEEE Computer Society, 2004, pp. 226–233. DOI: 10.1109/IAW.2004.1437821.

[Hil77] T. P. HILL. "On Goods and Services". In: *Review of Income and Wealth* 23.4 (1977), pp. 315–338.

[Hil99] T. P. HILL. "Tangibles, Intangibles and Services: A New Taxonomy for the Classification of Output". In: *Canadian Journal of Economics* 32.2 (Apr. 1999), pp. 426–446.

[HL01] F. S. HILLIER and G. J. LIEBERMAN. *Introduction to Operations Research*. 7th ed. Boston: McGraw-Hill, 2001.

[HL13] B. HEINRICH and L. LEWERENZ. "QoS-Aware Service Selection Considering Potential Service Failures". In: *Wirtschaftsinformatik Proceedings*. Paper 26. 2013.

[Hoa69] C. A. R. HOARE. "An Axiomatic Basis for Computer Programming". In: *Communications of the ACM* 12.10 (Oct. 1969), pp. 576–580. DOI: 10.1145/363235.363259.

[HW03] G. HOHPE and B. WOOLF. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Amsterdam: Addison-Wesley Longman, 2003.

[HZ12] B. HEINRICH and S. ZIMMERMANN. "Granularity Metrics for IT Services". In: *Proceedings of the 33rd International Conference on Information Systems*. ICIS '12. 2012, pp. 1–19.

[IBR10] S. IQBAL, M. F. BARI, and M. S. RAHMAN. "Solving the Multi-dimensional Multi-choice Knapsack Problem with the Help of Ants". In: *Proceedings of the 7th International Conference on Swarm Intelligence*. ANTS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 312–323. DOI: 10.1007/978-3-642-15461-4_27.

[ISO08] ISO/IEC. *Security Techniques - Information Security Management Systems - Requirements (ISO/IEC 27001:2005)*. Sept. 2008.

[ISO11] ISO/IEC/IEEE. *Systems and Software Engineering – Architecture Description (ISO/IEC/IEEE 42010:2011)*. Dec. 2011.

[JD88] A. K. JAIN and R. C. DUBES. *Algorithms for Clustering Data*. Eaglewood Cliffs, New Jersey 07632: Prentice Hall, 1988.

[JF09] M. JENSEN and S. FEJA. "A Security Modeling Approach for Web-Service-based Business Processes". In: *Proceedings of the 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS)*. ECBS '09. Washington, DC: IEEE Computer Society, 2009, pp. 340–347. DOI: 10.1109/ECBS.2009.14.

[JM07] M. C. JAEGER and G. MÜHL. "QoS-based Selection of Services: The Implementation of a Genetic Algorithm". In: *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*. 2007, pp. 1–12.

[JRM04] M. C. JAEGER, G. ROJEC-GOLDMANN, and G. MÜHL. "QoS Aggregation for Web Service Composition using Workflow Patterns". In: *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference*. EDOC '04. Washington, DC: IEEE Computer Society, 2004, pp. 149–159. DOI: 10.1109/EDOC.2004.1342512.

[JRM05] M. C. JAEGER, G. ROJEC-GOLDMANN, and G. 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) on e-Technology, e-Commerce and e-Service*. EEE '05. Washington, DC: IEEE Computer Society, 2005, pp. 181–185. DOI: 10.1109/EEE.2005.110.

[Kar08] D. KARAGIANNIS. "A Business Process-Based Modelling Extension for Regulatory Compliance". In: *MKWI08 - Multikonferenz Wirtschaftsinformatik*. Ed. by M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, B. Speitkamp, and P. Wolf. Berlin: GITO-Verlag, 2008, pp. 1159–1173.

[Kau09] L. M. KAUFMAN. "Data Security in the World of Cloud Computing". In: *IEEE Security & Privacy* 7.4 (July 2009), pp. 61–64. DOI: 10.1109/MSP.2009.87.

[KBB+12] F. KARATAS, M. BOURIMI, T. BARTH, D. KESDOGAN, R. GIMENEZ, W. SCHWITTEK, and M. PLANAGUMA. "Towards Secure and At-Runtime Tailorable Customer-Driven Public Cloud Deployment". In: *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. Mar. 2012, pp. 124–130. DOI: 10.1109/PerComW.2012.6197463.

[KBK+12] F. KARATAS, T. BARTH, D. KESDOGAN, H. M. FARDOUN, and P. G. VILLANUEVA. "Using Distributed User Interfaces to Evaluate Decision Making in Cloud Deployment". In: *Proceedings of the 2nd Workshop on Distributed User Interfaces (DUI) in conjunction with ACM CHI 2012*. 2012, pp. 17–21.

[KBK+13a] F. KARATAS, T. BARTH, D. KESDOGAN, H. M. FARDOUN, and P. G. VILLANUEAVA. "Uso de interface distribuída para avaliar a tomada de decisão na nuvem". In: *RTI Magazine, Brazil* (Jan. 2013). URL: `http://www.arandanet.com.br/midiaonline/rti/2013/janeiro/index.html`.

[KBK+13b] F. KARATAS, M. BOURIMI, D. KESDOGAN, P. G. VILLANUEVA, and H. M. FARDOUN. "Evaluating Usability and Privacy in Collaboration Settings with DUIs: Problem Analysis and Case Studies". In: *Distributed User Interfaces: Usability and Collaboration*. Ed. by M. D. Lozano, J. A. Gallud, R. Tesoriero, and V. M. R. Penichet. Human-Computer Interaction Series. London: Springer-Verlag, 2013. Chap. 10, pp. 119–127. DOI: `10.1007/978-1-4471-5499-0_10`.

[KBK13] F. KARATAS, M. BOURIMI, and D. KESDOGAN. "Towards Visual Configuration Support for Interdependent Security Goals". In: *Online Communities and Social Computing*. Ed. by A. A. Ozok and P. Zaphiris. Vol. 8029. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 375–384. DOI: `10.1007/978-3-642-39371-6_42`.

[KBS04] D. KRAFZIG, K. BANKE, and D. SLAMA. *Enterprise SOA: Service-Oriented Architecture Best Practices*. The Coad Series. Upper Saddle River, NJ: Prentice Hall, Nov. 2004.

[KC03] J. O. KEPHART and D. M. CHESS. "The Vision of Autonomic Computing". In: *Computer* 36.1 (Jan. 2003), pp. 41–50. DOI: `10.1109/MC.2003.1160055`.

[Kes00] D. KESDOĞAN. *Privacy im Internet*. Braunschweig, Wiesbaden: Vieweg, 2000.

[KFK15] F. KARATAS, L. FISCHER, and D. KESDOGAN. "Service Composition with Consideration of Interdependent Security Objectives". In: *Science of Computer Programming* 97 (2015), pp. 183–201. DOI: `10.1016/j.scico.2014.06.016`.

[KG06] M. Klusch and A. Gerber. “Evaluation of Service Composition Planning with OWLS-XPlan”. In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. WI-IATW ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 117–120. doi: 10.1109/WI-IATW.2006.68.

[KH03] H. Kunreuther and G. Heal. “Interdependent Security”. In: *The Journal of Risk and Uncertainty* 26.2-3 (2003), pp. 231–249.

[Kha98] M. S. Khan. “Quality Adaptation in a Multisession Multimedia System: Model, Algorithms, and Architecture”. PhD thesis. University of Victoria, 1998.

[KHB+13] F. Karatas, M. Heupel, M. Bourimi, D. Kesdogan, and S. Wrobel. “Considering Interdependent Protection Goals in Domain-Specific Contexts: The di.me Case Study”. In: *Proceedings of the 10th International Conference on Information Technology - New Generations*. ITNG ’13. Washington, DC: IEEE, 2013, pp. 27–32. doi: 10.1109/ITNG.2013.12.

[KHH+11] A. Krammer, B. Heinrich, M. Henneberger, and F. Lautenbacher. “Granularity of Services: An Economic Analysis”. In: *Business & Information Systems Engineering* 3.6 (2011), pp. 345–358. doi: 10.1007/s12599-011-0189-x.

[KIH11] A. Klein, F. Ishikawa, and S. Honiden. “Efficient Heuristic Approach with Improved Time Complexity for Qos-Aware Service Composition”. In: *Proceedings of the 2011 IEEE International Conference on Web Services*. ICWS ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 436–443. doi: 10.1109/ICWS.2011.60.

[KIH12] A. Klein, F. Ishikawa, and S. Honiden. “Towards Network-aware Service Composition in the Cloud”. In: *Proceedings of the 21st International Conference on World Wide Web*. WWW ’12. New York, NY, USA: ACM, 2012, pp. 959–968. doi: 10.1145/2187836.2187965.

[KK13a] F. Karatas and D. Kesdogan. “A Flexible Approach For Considering Interdependent Security Objectives in Service Composition”. In: *Proceedings of the 28th Symposium on Applied Computing*. SAC ’13. New York, NY, USA: ACM, 2013, pp. 1919–1926. doi: 10.1145/2480362.2480717.

[KK13b] F. KARATAS and D. KESDOGAN. "An Approach for Compliance-Aware Service Selection with Genetic Algorithms". In: *Proceedings of the 11th International Conference on Service Oriented Computing*. Ed. by S. Basu, C. Pautasso, L. Zhang, and X. Fu. ICSOC '11. Berlin, Heidelberg: Springer, 2013, pp. 465–473. DOI: `10.1007/978-3-642-45005-1_35`.

[Kle69] L. KLEINROCK. *UCLA to be First Station in Nationwide Computer Network*. UCLA Press Release. July 1969.

[KM05] A. KOSCHMIDER and M. MEVIUS. "A Petri Net Based Approach for Process Model Driven Deduction of BPEL Code". In: *Proceedings of the 2005 OTM Confederated International Conference on On the Move to Meaningful Internet Systems*. OTM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 495–505. DOI: `10.1007/11575863_69`.

[Kno03] E. KNORR. *2004: The Year of Web Services*. `http://www.cio.com/article/32050/2004_The_Year_of_Web_Services`. Accessed 14 September 2013. Dec. 2003.

[KP06] D. KESDOĞAN and C. C. PALMER. "Technical Challenges of Network Anonymity". In: *Computer Communications* 29.3 (Feb. 2006), pp. 306–324. DOI: `10.1016/j.comcom.2004.12.011`.

[KPM14] KPMG. *Cloud Monitor 2014: Cloud Computing in Deutschland – Status quo und Perspektiven*. `http://www.kpmg.com/DE/de/Documents/cloudmonitor-2014-kpmg.pdf`. Accessed 19 April 2014. 2014.

[KPP04] H. KELLERER, U. PFERSCHY, and D. PISINGER. *Knapsack Problems*. Springer Berlin Heidelberg, 2004. DOI: `10.1007/978-3-540-24777-7`.

[KR11] C. KURZ and F. RIEGER. *Die Datenfresser*. Frankfurt a. M.: S. Fischer, 2011.

[KS12] K. KITTEL and S. SACKMANN. "Flexible Controls for Compliance in Catastrophe Management Processes". In: *Proceedings of Multikonferenz Wirtschaftsinformatik*. MKWI'12. Berlin: GITO-Verlag, 2012, pp. 1675–1687.

[KSW98] J. KELSEY, B. SCHNEIER, and D. WAGNER. "Protocol Interactions and the Chosen Protocol Attack". In: *Proceedings of the 5th International Workshop on Security Protocols*. Ed. by B. Christianson, B. Crispo, M. Lomas, and M. Roe. Vol. 1361. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 1998, pp. 91–104. DOI: `10.1007/BFb0028162`.

[KTV+10] J. Kreiker, A. Tarlecki, M. Y. Vardi, and R. Wilhelm. "Modeling, Analysis, and Verification - The Formal Methods Manifesto 2010 (Dagstuhl Perspectives Workshop 10482)". In: *Dagstuhl Manifestos* 1.1 (2010), pp. 21–40. DOI: 10.4230/DagMan.1.1.21.

[LAP06] A. Lazovik, M. Aiello, and M. Papazoglou. "Planning and Monitoring the Execution of Web Service Requests". In: *International Journal on Digital Libraries* 6.3 (June 2006), pp. 235–246. DOI: 10.1007/s00799-006-0002-5.

[LC11] T. Lavarack and M. Coetzee. "Web Services Security Policy Assertion Trade-offs". In: *Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security*. ARES '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 535–540. DOI: 10.1109/ARES.2011.80.

[LD60] A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (July 1960), pp. 497–520. DOI: 10.2307/1910129.

[LDS+10] M. Li, T. Deng, H. Sun, H. Guo, and X. Liu. "GOS: A Global Optimal Selection Approach for QoS-Aware Web Services Composition". In: *Proceedings of the 2010 Fifth IEEE International Symposium on Service Oriented System Engineering*. SOSE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 7–14. DOI: 10.1109/SOSE.2010.47.

[Lee03] J. Lee. "Matching Algorithms for Composing Business Process Solutions with Web Services". In: *Proceedings of the 4th International Conference on E-Commerce and Web Technologies*. Ed. by K. Bauknecht, A. M. Tjoa, and G. Quirchmayr. EC-Web '03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 393–402. DOI: 10.1007/978-3-540-45229-4_38.

[Len06] A. K. Lenstra. "Key Lengths". In: *Handbook of Information Security, Volume 2: Information Warfare; Social, Legal and International Issues; and Security Foundations*. Ed. by H. Bidgoli. Hoboken, New Jersey: John Wiley & Sons, Inc., 2006, pp. 617–635.

[Ley01] F. Leymann. *Web Services Flow Language (WSFL 1.0)*. Accessed 27 December 2015. May 2001. URL: http://xml.coverpages.org/WSFL-Guide-200110.pdf.

[LH07] C. Legner and R. Heutschi. "SOA Adoption in Practice - Findings from Early SOA Implementations". In: *Proceedings of the 15th European Conference on Information Systems*. ECIS'07. 2007, pp. 1643–1654.

[LJL+03] K.-C. LEE, J.-H. JEON, W.-S. LEE, S.-H. JEONG, and S.-W. PARK. *QoS for Web Services: Requirements and Possible Approaches*. W3C Note. Accessed 27 December 2015. W3C, Nov. 2003. URL: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[LKD+03] H. LUDWIG, A. KELLER, A. DAN, R. P. KING, and R. FRANCK. *Web Service Level Agreement (WSLA) Language Specification*. wsla-2003/01/28. Accessed 27 December 2015. 2003. URL: http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf.

[LKS08] N. LIN, U. KUTER, and E. SIRIN. "Web Service Composition with User Preferences". In: *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications*. Ed. by S. Bechhofer, M. Hauswirth, J. Hoffman, and M. Koubarakis. ESWC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 629–643. DOI: 10.1007/978-3-540-68234-9_46.

[LM11] F. LÉCUÉ and N. MEHANDJIEV. "Seeking Quality of Web Service Composition in a Semantic Dimension". In: *IEEE Transactions on Knowledge and Data Engineering* 23.6 (June 2011), pp. 942–959. DOI: 10.1109/TKDE.2010.237.

[LMH+06] I. M. LLORENTE, R. S. MONTERO, E. HUEDO, and K. LEAL. "A Grid Infrastructure for Utility Computing". In: *Proceedings of the 15th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. WETICE '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 163–168. DOI: 10.1109/WETICE.2006.7.

[LMX07] Y. LIU, S. MÜLLER, and K. XU. "A Static Compliance-Checking Framework for Business Process Models". In: *IBM Systems Journal* 46.2 (2007), pp. 335–361. DOI: 10.1147/sj.462.0335.

[LQS12] A. LOMUSCIO, H. QU, and M. SOLANKI. "Towards Verifying Contract Regulated Service Composition". In: *Autonomous Agents and Multi-Agent Systems* 24.3 (May 2012), pp. 345–373. DOI: 10.1007/s10458-010-9152-3.

[LRG+12] L. T. LY, S. RINDERLE-MA, K. GÖSER, and P. DADAM. "On Enabling Integrated Process Compliance with Semantic Constraints in Process Management Systems". In: *Information Systems Frontiers* 14.2 (Apr. 2012), pp. 195–219. DOI: 10.1007/s10796-009-9185-9.

[LRK+11] L. T. LY, S. RINDERLE-MA, D. KNUPLESCH, and P. DADAM. "Monitoring Business Process Compliance using Compliance Rule Graphs". In: *On the Move to Meaningful Internet Systems: OTM 2011*. OTM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 82–99. DOI: 10.1007/978-3-642-25109-2_7.

[LRM+12] M. LAMA, P. RODRIGUEZ-MIER, M. MUCIENTES, and J. C. VIDAL. "An Optimal and Complete Algorithm for Automatic Web Service Composition". In: *International Journal of Web Services Research* 9.2 (Apr. 2012), pp. 1–20. DOI: 10.4018/jwsr.2012040101.

[LRS02] F. LEYMANN, D. ROLLER, and M.-T. SCHMIDT. "Web Services and Business Process Management". In: *IBM Systems Journal* 41.2 (Apr. 2002), pp. 198–211. DOI: 10.1147/sj.412.0198.

[LV00] A. K. LENSTRA and E. R. VERHEUL. "Selecting Cryptographic Key Sizes". In: *Public Key Cryptography*. Ed. by H. Imai and Y. Zheng. Vol. 1751. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000. DOI: 10.1007/978-3-540-46588-1_30.

[LV01] A. K. LENSTRA and E. R. VERHEUL. "Selecting Cryptographic Key Sizes". In: *Journal of Cryptology* 14.4 (2001), pp. 255–293. DOI: 10.1007/s00145-001-0009-4.

[LWL09] Q. LIANG, X. WU, and H. C. LAU. "Optimizing Service Systems Based on Application-Level QoS". In: *IEEE Transactions on Services Computing* 2.2 (Apr. 2009), pp. 108–121. DOI: 10.1109/TSC.2009.13.

[Man09] A. T. MANES. *SOA is Dead; Long Live Services*. http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html. Accessed 7 September 2013. Jan. 2009.

[MBK+09] N. B. MABROUK, S. BEAUCHE, E. KUZNETSOVA, N. GEORGANTAS, and V. ISSARNY. "QoS-aware Service Composition in Dynamic Service Oriented Environments". In: *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*. Ed. by J. M. Bacon and B. F. Cooper. Middleware '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 123–142. DOI: 10.1007/978-3-642-10445-9_7.

[MCD10] D. A. MENASCÉ, E. CASALICCHIO, and V. DUBEY. "On Optimal Service Selection in Service Oriented Architectures". In: *Performance Evaluation* 67.8 (Aug. 2010), pp. 659–675. DOI: 10.1016/j.peva.2009.07.001.

[MCF87] J. K. MILLEN, S. C. CLARK, and S. B. FREEMAN. "The Interrogator: Protocol Secuity Analysis". In: *IEEE Transactions on Software Engineering* 13.2 (Feb. 1987), pp. 274–288. DOI: `10.1109/TSE.1987.233151`.

[McK05] J. MCKENDRICK. *Survey: JBOWS will be around for a long time*. `http://www.zdnet.com/blog/service-oriented/survey-jbows-will-be-around-for-a-long-time/445`. Accessed 18 September 2013. Oct. 2005.

[McK08] J. MCKENDRICK. *Study: Only one out of five SOA efforts bearing fruit*. `http://www.zdnet.com/article/study-only-one-out-of-five-soa-efforts-bearing-fruit/`. Accessed 29 December 2015. July 2008.

[MDC11] E. MARTIŃEZ, G. DIÁZ, and M. E. CAMBRONERO. "Contractually Compliant Service Compositions". In: *Proceedings of the 9th international conference on Service-Oriented Computing*. ICSOC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 636–644. DOI: `10.1007/978-3-642-25535-9_50`.

[Mea96] C. MEADOWS. "The NRL Protocol Analyzer: An Overview". In: *The Journal of Logic Programming* 26.2 (1996), pp. 113–131. DOI: `10.1016/0743-1066(95)00095-X`.

[Mee08] M. MEEHAN. *SOA Adoption Marked by Broad Failure and Wild Success*. `http://searchsoa.techtarget.com/news/1319609/SOA-adoption-marked-by-broad-failure-and-wild-success`. Accessed 18 September 2013. July 2008.

[Men02] D. A. MENASCÉ. "QoS Issues in Web Services". In: *IEEE Internet Computing* 6.6 (Nov. 2002), pp. 72–75. DOI: `10.1109/MIC.2002.1067740`.

[Men04] D. A. MENASCÉ. "Composing Web Services: A QoS View". In: *IEEE Internet Computing* 8.6 (Nov. 2004), pp. 80–90. DOI: `10.1109/MIC.2004.71`.

[MG11] P. MELL and T. GRANCE. *The NIST Definition of Cloud Computing*. Special Publication 800-145. Accessed 27 December 2015. Sept. 2011. URL: `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

[Mic] MICROSOFT. *XLANG/s Language*. `http://msdn.microsoft.com/en-us/library/aa577463.aspx`. Accessed 1 August 2013.

[Mil93] R. MILNER. "The Polyadic π-Calculus: a Tutorial". In: *Logic and Algebra of Specification*. Ed. by F. Bauer, W. Brauer, and H. Schwichtenberg. Vol. 94. NATO ASI Series. Berlin, Heidelberg: Springer-Verlag, 1993, pp. 203–246. DOI: 10.1007/978-3-642-58041-3_6.

[MJD+02] Z. MILOSEVIC, A. J"SANG, T. DIMITRAKOS, and M. A. PATTON. "Discretionary Enforcement of Electronic Contracts". In: *Proceedings of the Sixth International Enterprise Distributed Object Computing Conference (EDOC'02)*. EDOC '02. 2002, pp. 39–50. DOI: 10.1109/EDOC.2002.1137695.

[MKL+09] G. MONAKOVA, O. KOPP, F. LEYMANN, S. MOSER, and K. SCHÄFERS. "Verifying Business Rules Using an SMT Solver for BPEL Processes". In: *BPSC*. Ed. by W. Abramowicz, L. A. Maciaszek, R. Kowalczyk, and A. Speck. Vol. 147. LNI. GI, 2009, pp. 81–94.

[MKL09] T. MATHER, S. KUMARASWAMY, and S. LATIF. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. Sebastopol, CA: O'Reilley, 2009.

[MLB+11] S. MARSTON, Z. LI, S. BANDYOPADHYAY, J. ZHANG, and A. GHALSASI. "Cloud Computing – The Business Perspective". In: *Decision Support Systems* 51.1 (Apr. 2011), pp. 176–189. DOI: 10.1016/j.dss.2010.12.006.

[MN02] A. MANI and A. NAGARAJAN. *Understanding Quality of Service for Web Services*. http://www.ibm.com/developerworks/webservices/library/ws-quality/. Accessed 13 November 2013. Jan. 2002.

[Mon03] J.-F. MONIN. *Understanding Formal Methods*. Ed. by M. G. Hinchey. London: Springer-Verlag, 2003. DOI: 10.1007/978-1-4471-0043-0.

[Mon13] F. MONTESI. "Process-aware Web Programming with Jolie". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. New York, NY: ACM, 2013, pp. 761–763. DOI: 10.1145/2480362.2480507.

[MOV97] A. J. MENEZES, P. C. van OORSCHOT, and S. A. VANSTONE. *Handbook of Applied Cryptography*. Boca Raton: CRC Press, 1997.

[MS04] E. M. MAXIMILIEN and M. P. SINGH. "A Framework and Ontology for Dynamic Web Services Selection". In: *IEEE Internet Computing* 8.5 (Sept. 2004), pp. 84–93. DOI: 10.1109/MIC.2004.27.

[MSB10] H. S. Meda, A. K. Sen, and A. Bagchi. "On Detecting Data Flow Errors in Workflows". In: *ACM Journal of Data and Information Quality* 2.1 (July 2010), 4:1–4:31. doi: 10.1145/1805286.1805290.

[MSD] MSDN. *Principles of Service Oriented Design*. http://msdn.microsoft.com/en-us/library/bb972954.aspx. Accessed 11 September 2013.

[MSR10] N. Moebius, K. Stenzel, and W. Reif. "Formal Verification of Application-specific Security Properties in a Model-driven Approach". In: *Proceedings of the Second International Conference on Engineering Secure Software and Systems*. ESSoS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 166–181. doi: 10.1007/978-3-642-11747-3_13.

[MT87] S. Martello and P. Toth. "Algorithms for Knapsack Problems". In: *Annals of Discrete Mathematics* 31 (1987), pp. 213–258.

[Mur13] S. Murer. *15 Years of Service Oriented Architecture at Credit Suisse: Lessons Learned – Remaining Challenges*. http://www.sei.cmu.edu/library/assets/presentations/murer-saturn2013.pdf. Accessed 17 September 2013. May 2013.

[Nec98] G. C. Necula. "Compiling wih Proofs". Accessed 27 December 2015. PhD thesis. Carnegie Mellon University, Sept. 1998. url: https://www.cs.cmu.edu/~rwh/theses/necula.pdf.

[NL12] A. Nieto and J. Lopez. "Security and QoS Tradeoffs: Towards a FI Perspective". In: *Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 2012, pp. 745–750. doi: 10.1109/WAINA.2012.204.

[NM02] S. Narayanan and S. A. McIlraith. "Simulation, Verification and Automated Composition of Web Services". In: *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. New York, NY, USA: ACM, 2002, pp. 77–88. doi: 10.1145/511446.511457.

[NS07] K. Namiri and N. Stojanovic. "Pattern-based Design and Validation of Business Process Compliance". In: *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*. OTM'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 59–76. doi: 10.1007/978-3-540-76848-7_6.

[NSA13] NSA. *NSA Suite B Cryptography*. Accessed 24 April 2013. Feb. 2013. url: http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[NTI+05] Y. Nakamura, M. Tatsubori, T. Imamura, and K. Ono. "Model-Driven Security Based on a Web Services Security Architecture". In: *Proceedings of the 2005 IEEE International Conference on Services Computing - Volume 01*. SCC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 7–15. DOI: 10.1109/SCC.2005.66.

[Oan07] O. I. Oanea. "Verification of Soundness and Other Properties of Business Processes". PhD thesis. Technische Universiteit Eindhoven, 2007. DOI: 10.6100/IR631345.

[OAS06] OASIS. *Reference Model for Service Oriented Architecture 1.0*. Accessed 27 December 2015. Aug. 2006. URL: https://www.oasis-open.org/committees/download.php/19679/.

[OAS07] OASIS. *Web Services Business Process Execution Language Version 2.0*. Accessed 1 August 2013. Apr. 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[ÖBF+10] H. Österle, J. Becker, U. Frank, T. Hess, D. Karagiannis, H. Krcmar, P. Loos, P. Mertens, A. Oberweis, and E. J. Sinz. "Memorandum zur gestaltungsorientierten Wirtschaftsinformatik". In: *Zeitschrift für betriebswirtschaftliche Forschung* 11 (2010), pp. 664–669.

[ODH+06] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst. "From BPMN Process Models to BPEL Web Services". In: *Proceedings of the IEEE International Conference on Web Services*. ICWS '06. Washington, DC: IEEE Computer Society, 2006, pp. 285–292. DOI: 10.1109/ICWS.2006.67.

[OEH02] J. O'Sullivan, D. Edmond, and A. ter Hofstede. "What's in a Service? Towards Accurate Description of Non-Functional Service Properties". In: *Distributed and Parallel Databases* 12.2 (2002), pp. 117–133. DOI: 10.1023/A:1016547000822.

[OH04] H. Orman and P. Hoffman. *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*. RFC 3766. Accessed 27 December 2015. Internet Engineering Task Force, Apr. 2004. URL: http://www.ietf.org/rfc/rfc3766.txt.

[OMG03] OMG. *MDA Guide Version 1.0.1*. Accessed 25 September 2013. June 2003. URL: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf.

[OMG11] OMG. *Business Process Model and Notation (BPMN) Version 2.0*. Accessed 17 February 2011. Jan. 2011. URL: http://www.omg.org/spec/BPMN/2.0.

[PA06] M. PESIC and W. M. P. van der AALST. "A Declarative Approach for Flexible Business Processes Management". In: *Proceedings of the 2006 International Conference on Business Process Management Workshops*. BPM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 169–180. DOI: 10.1007/11837862_18.

[Pau97] L. C. PAULSON. "Proving Properties of Security Protocols by Induction". In: *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*. CSFW '97. Washington, DC: IEEE Computer Society, 1997, pp. 70–83. DOI: 10.1109/CSFW.1997.596788.

[PC10] S. L. PFLEEGER and R. K. CUNNINGHAM. "Why Measuring Security is Hard". In: *IEEE Security & Privacy* 8.4 (July 2010), pp. 46–54. DOI: 10.1109/MSP.2010.60.

[Pfi00] A. PFITZMANN. *Sicherheit in Rechnernetzen: Mehrseitige Sicherheit in verteilten und durch verteilte Systeme*. http://dud.inf.tu-dresden.de/~pfitza/DSuKrypt.pdf. Accessed 1 March 2014. Oct. 2000.

[PG03] M. P. PAPAZOGLOU and D. GEORGAKOPOULOS. "Service-Oriented Computing". In: *Communications of the ACM* 46.10 (Oct. 2003), pp. 24–28. DOI: 10.1145/944217.944233.

[Pis95] D. PISINGER. "A Minimal Algorithm for the Multiple-Choice Knapsack Problem". In: *European Journal of Operational Research* 83.2 (1995), pp. 394–410. DOI: 10.1016/0377-2217(95)00015-I.

[POM08] F. PACI, M. OUZZANI, and M. MECELLA. "Verification of Access Control Requirements in Web Services Choreography". In: *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 1*. SCC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 5–12. DOI: 10.1109/SCC.2008.116.

[Pon13] PONEMON INSTITUTE, LLC. *2013 Cost of Data Breach Study: Global Analysis*. Tech. rep. Accessed 22 April 2014. May 2013. URL: http://www.ponemon.org/local/upload/file/2013%20Report%20GLOBAL%20CODB%20FINAL%205-2.pdf.

[PR91] M. PADBERG and G. RINALDI. "A Branch-and-cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems". In: *SIAM Review* 33.1 (Feb. 1991), pp. 60–100. DOI: 10.1137/1033004.

[PS06] A. Piszcz and T. Soule. “Genetic Programming: Optimal Population Sizes for Varying Complexity Problems”. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’06. New York, NY, USA: ACM, 2006, pp. 953–954. doi: 10.1145/1143997.1144166.

[PS97] P. P. Puschner and A. V. Schedl. “Computing Maximum Task Execution Times – A Graph-Based Approach”. In: *Real-Time Systems* 13 (July 1997), pp. 67–91. doi: 10.1023/A:1007905003094.

[PTD+07] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. “Service-Oriented Computing: State of the Art and Research Challenges”. In: *Computer* 40.11 (Nov. 2007), pp. 38–45. doi: 10.1109/MC.2007.400.

[PW05] F. Puhlmann and M. Weske. “Using the π-calculus for Formalizing Workflow Patterns”. In: *Proceedings of the 3rd International Conference on Business Process Management*. BPM’05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 153–168. doi: 10.1007/11538394_11.

[RA08] A. Rozinat and W. M. P. van der Aalst. “Conformance Checking of Processes Based on Monitoring Real Behavior”. In: *Information Systems* 33.1 (2008), pp. 64–95. issn: 0306-4379. doi: 10.1016/j.is.2007.07.001.

[Rap04] M. A. Rappa. “The Utility Business Model and the Future of Computing Services”. In: *IBM Systems Journal* 43.1 (Jan. 2004), pp. 32–42. doi: 10.1147/sj.431.0032.

[RCL10] B. P. Rimal, E. Choi, and I. Lumb. “A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems”. In: *Cloud Computing: Principles, Systems and Applications*. Ed. by N. Antonopoulos and L. Gillam. Computer Communications and Networks. London: Springer, 2010, pp. 21–46. doi: 10.1007/978-1-84996-241-4_2.

[RG09] M. Razzazi and T. Ghasemi. “An Exact Algorithm for the Multiple-Choice Multidimensional Knapsack Based on the Core”. In: *Proceedings of the 13th International CSI Computer Conference*. Ed. by H. Sarbazi-Azad, B. Parhami, S.-G. Miremadi, and S. Hessabi. CSICC’08. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 275–282. doi: 10.1007/978-3-540-89985-3_34.

[RHA+06] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar. *Workflow Control-Flow Patterns: A Revised View*. Tech. rep. BPM Center Report BPM-06-22. Accessed 28 February 2014.

BPMcenter.org, 2006. URL: http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf.

[Ric53] H. G. Rice. "Classes of Recursively Enumerable Sets and their Decision Problems". In: *Transactions of the American Mathematical Society* 74 (1953), pp. 358–366. DOI: 10.1090/S0002-9947-1953-0053041-6.

[RK02] R. T. Rust and P. K. Kannan, eds. *E-Service: New Directions in Theory and Practice*. Armonk, NY: M. E. Sharpe, Inc., 2002.

[RM06] J. Recker and J. Mendling. "On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Language". In: *Proceedings of the 18th International Conference on Advanced Information Systems Engineering*. Ed. by T. Latour and M. Petit. 2006, pp. 521–532.

[RM09] S. Rossi and D. Macedonio. "Information Flow Security for Service Compositions". In: *Proceedings of the 2009 International Conference on Ultra Modern Telecommunications Workshops*. ICUMT '09. 2009, pp. 1–8. DOI: 10.1109/ICUMT.2009.5345455.

[RSD+10] C. Rodriǵuez, P. Silveira, F. Daniel, and F. Casati. "Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction". In: *Proceedings of the 10th International Conference on Current Trends in Web Engineering*. Ed. by F. Daniel and F. M. Facca. ICWE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 277–288. DOI: 10.1007/978-3-642-16985-4_25.

[RW12] M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems*. Berlin, Heidelberg: Springer, 2012. DOI: 10.1007/978-3-642-30409-5.

[SA09] R. M. Savola and H. Abie. "Identification of Basic Measurable Security Components for a Distributed Messaging System". In: *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*. SECURWARE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 121–128. DOI: 10.1109/SECURWARE.2009.26.

[Saa90] T. L. Saaty. "How to make a decision: The Analytic Hierarchy Process". In: *European Journal of Operational Research* 48 (1990), pp. 9–26.

[SAL+09] D. SCHLEICHER, T. ANSTETT, F. LEYMANN, and R. MIETZNER. "Maintaining Compliance in Customizable Process Models". In: *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*. OTM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 60–75. DOI: 10.1007/978-3-642-05148-7_7.

[San03] R. SANDHU. "Good-Enough Security". In: *IEEE Internet Computing* 7.1 (Jan. 2003), pp. 66–68. DOI: 10.1109/MIC.2003.1167341.

[SAP05] SAP NEWS DESK. *Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort*. http://soa.sys-con.com/node/164624. Accessed 16 September 2013. Dec. 2005.

[SBH+10] H. SUN, S. BASU, V. HONAVAR, and R. LUTZ. "Automata-Based Verification of Security Requirements of Composite Web Services". In: *Proceedings of the 2010 IEEE 21st International Symposium on Software Reliability Engineering*. ISSRE '10. Washington, DC: IEEE Computer Society, 2010, pp. 348–357. DOI: 10.1109/ISSRE.2010.20.

[Sbi07] A. SBIHI. "A Best First Search Exact Algorithm for the Multiple-Choice Multidimensional Knapsack Problem". In: *Journal of Combinatorial Optimization* 13.4 (2007), pp. 337–351. DOI: 10.1007/s10878-006-9035-3.

[SCD+97] B. SABATA, S. CHATTERJEE, M. DAVIS, J. J. SYDIR, and T. F. LAWRENCE. "Taxonomy for QoS Specifications". In: *Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems*. 1997, pp. 100–107. DOI: 10.1109/WORDS.1997.609931.

[Sch01] J. B. SCHMITT. *Heterogeneous Network Quality of Service Systems*. Vol. 622. The Springer International Series in Engineering and Computer Science. Berlin, Heidelberg: Springer-Verlag, 2001. DOI: 10.1007/978-1-4615-1419-0.

[Sch99] A.-W. SCHEER. *ARIS – Business Process Frameworks*. 3rd. Heidelberg: Springer-Verlag, 1999.

[SCK11] A. SQUICCIARINI, B. CARMINATI, and S. KARUMANCHI. "A Privacy-Preserving Approach for Web Service Selection and Provisioning". In: *Proceedings of the 2011 IEEE International Conference on Web Services*. ICWS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 33–40. DOI: 10.1109/ICWS.2011.120.

[SCM+07] A. Svirskas, C. Courbis, R. Molva, and J. Bedzinskas. "Compliance Proofs for Collaborative Interactions using Aspect-Oriented Approach". In: *Proceedings of the 2007 IEEE Congress on Services*. SERVICES 2007. 2007, pp. 33–40. doi: 10.1109/SERVICES.2007.23.

[SGH+11] S. Scerri, R. Gimenez, F. Herman, M. Bourimi, and S. Thiel. "digital.me – Towards an Integrated Personal Information Sphere". In: *Proceedings of the Federated Social Web Summit*. 2011.

[SGN07] S. Sadiq, G. Governatori, and K. Namiri. "Modeling Control Objectives for Business Process Compliance". In: *Business Process Management*. Ed. by G. Alonso, P. Dadam, and M. Rosemann. Vol. 4714. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 149–164. doi: 10.1007/978-3-540-75183-0_12.

[SI07] S. Stein and K. Ivanov. "EPK nach BPEL Transformation als Voraussetzung für praktische Umsetzung einer SOA". In: *Software Engineering*. Ed. by S. Böttinger, L. Theuvsen, S. Rank, and M. Morgenstern. 2007, pp. 75–82.

[SIM+07] M. Srivatsa, A. Iyengar, T. Mikalsen, I. Rouvellou, and J. Yin. "An Access Control System for Web Service Compositions". In: *Proceedings of the 2007 International Conference on Web Services*. ICWS '07. July 2007, pp. 1–8. doi: 10.1109/ICWS.2007.31.

[Sim12] K. M. Sim. "Agent-Based Cloud Computing". In: *IEEE Transactions on Services Computing* 5.4 (Jan. 2012), pp. 564–577. doi: 10.1109/TSC.2011.52.

[SK08] S. Sackmann and M. Kähmer. "ExPDT: A Policy-based Approach for Automating Compliance". In: *WIRTSCHAFTSINFORMATIK* 50.5 (2008), pp. 366–374. doi: 10.1007/s11576-008-0078-1.

[SLI12] C. Shi, D. Lin, and T. Ishida. "User-Centered QoS Computation for Web Service Selection". In: *Proceedings of the 2012 IEEE 19th International Conference on Web Services*. ICWS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 456–463. doi: 10.1109/ICWS.2012.18.

[Sma12] N. Smart. *ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)*. Tech. rep. Revision 1.0. Accessed 27 December 2015. Katholieke Universiteit Leuven (KUL), Sept. 2012. url: http://cordis.europa.eu/docs/projects/cnect/6/216676/080/deliverables/002-DSPA20.pdf.

[SN96a] R. W. SCHULTE and Y. V. NATIS. *SSA Research Note SPA-401-068, Service Oriented Architectures, Part 1*. Tech. rep. The Gartner Group, 1996.

[SN96b] R. W. SCHULTE and Y. V. NATIS. *SSA Research Note SPA-401-069, Service Oriented Architectures, Part 2*. Tech. rep. The Gartner Group, 1996.

[SPG+11] D. SCHULLER, A. POLYVYANYY, L. GARCIÁ-BAÑUELOS, and S. SCHULTE. "Optimization of Complex QoS-Aware Service Compositions". In: *Proceedings of the 9th International Conference on Service-Oriented Computing*. Ed. by G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad. ICSOC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 452–466. DOI: 10.1007/978-3-642-25535-9_30.

[SPW+04] E. SIRIN, B. PARSIA, D. WU, J. HENDLER, and D. NAU. "HTN planning for Web Service composition using SHOP2". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1.4 (Oct. 2004), pp. 377–396. DOI: 10.1016/j.websem.2004.06.005.

[SSS11] B. SULEIMAN, C. E. da SILVA, and S. SAKR. "One Size Does Not Fit All: A Group-Based Service Selection for Web-Based Business Processes". In: *Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*. WAINA '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 253–260. DOI: 10.1109/WAINA.2011.106.

[Ste46] S. S. STEVENS. "On the Theory of Scales of Measurement". In: *Science* 103.2684 (June 1946), pp. 677–680.

[STK+10] D. SCHUMM, O. TURETKEN, N. KOKASH, A. ELGAMMAL, F. LEYMANN, and W.-J. VAN DEN HEUVEL. "Business Process Compliance Through Reusable Units of Compliant Processes". In: *Proceedings of the 10th International Conference on Current Trends in Web Engineering*. ICWE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 325–337. DOI: 10.1007/978-3-642-16985-4_29.

[SYT+10] W. SHE, I.-L. YEN, B. THURAISINGHAM, and E. BERTINO. "Policy-Driven Service Composition with Information Flow Control". In: *Proceedings of the 2010 IEEE International Conference on Web Services*. ICWS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 50–57. DOI: 10.1109/ICWS.2010.37.

[SZ01] E. A. STOHR and J. L. ZHAO. "Workflow Automation: Overview and Research Issues". In: *Information Systems Frontiers* 3.3 (2001), pp. 281–296. DOI: 10.1023/A:1011457324641.

[Tar71] R. TARJAN. "Depth-first Search and Linear Graph Algorithms". In: *Proceedings of the 12th Annual Symposium on Switching and Automata Theory*. Oct. 1971, pp. 114–121. DOI: 10.1109/SWAT.1971.10.

[TAS09] N. TRČKA, W. M. P. van der AALST, and N. SIDOROVA. "Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows". In: *Proceedings of the 21st International Conference on Advanced Information Systems Engineering*. CAiSE '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 425–439. DOI: 10.1007/978-3-642-02144-2_34.

[TBG+12] S. THIEL, M. BOURIMI, R. GIMÉNEZ, S. SCERRI, A. SCHULLER, M. VALLA, S. WROBEL, C. FRÀ, and F. HERMANN. "A Requirements-Driven Approach Towards Decentralized Social Networks". In: *Future Information Technology, Application, and Service*. Ed. by J. J. (Jong Hyuk) Park, V. C. Leung, C.-L. Wang, and T. Shon. Vol. 164. Lecture Notes in Electrical Engineering. Springer Netherlands, 2012, pp. 709–718. DOI: 10.1007/978-94-007-4516-2_75.

[TBK+12] R. TESORIERO, M. BOURIMI, F. KARATAS, T. BARTH, P. G. VILLANUEVA, and P. SCHWARTE. "Model-Driven Privacy and Security in Multi-modal Social Media UIs". In: *Modeling and Mining Ubiquitous Social Media*. Ed. by M. Atzemueller, A. Chin, D. Helic, and A. Hotho. Vol. 7472. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 158–181. DOI: 10.1007/978-3-642-33684-3_9.

[TF09] O. THOMAS and M. FELLMANN. "Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes". In: *Business & Information Systems Engineering* 1.6 (2009), pp. 438–451. DOI: 10.1007/s12599-009-0078-8.

[The13a] THE JOLIE TEAM. *Jolie Programming Language - Official Website*. http://www.jolie-lang.org/. Accessed 1 August 2013. 2013.

[The13b] THE STANDISH GROUP. *Chaos Manifesto 2013*. Tech. rep. Accessed 27 December 2015. 2013. URL: https://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf.

[TSB10] W.-T. TSAI, X. SUN, and J. BALASOORIYA. "Service-Oriented Cloud Computing Architecture". In: *Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations*. ITNG '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 684–689. DOI: 10.1109/ITNG.2010.214.

[US 02] US CONGRESS. *Sarbanes-Oxley Act of 2002*. Accessed 5 October 2013. July 2002. URL: http://www.sec.gov/about/laws/soa2002.pdf.

[US 96] US CONGRESS. *Health Insurance Portability and Accountability Act of 1996*. Accessed 5 October 2013. July 1996. URL: http://www.gpo.gov/fdsys/pkg/CRPT-104hrpt736/pdf/CRPT-104hrpt736.pdf.

[VHA05] L.-H. VU, M. HAUSWIRTH, and K. ABERER. "QoS-Based Service Selection and Ranking with Trust and Reputation Management". In: *Proceedings of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part I*. Ed. by R. Meersman and Z. Tari. OTM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 466–483. DOI: 10.1007/11575771_30.

[VK83] V. L. VOYDOCK and S. T. KENT. "Security Mechanisms in High-Level Network Protocols". In: *ACM Computing Surveys* 15.2 (June 1983), pp. 135–171. DOI: 10.1145/356909.356913.

[VVE10] A. T. VELTE, T. J. VELTE, and R. ELSENPETER. *Cloud Computing – A Practical Approach*. New York: McGraw-Hill, 2010.

[VW93] P. F. VELLEMAN and L. WILKINSON. "Nominal, Ordinal, Interval, and Ratio Typologies Are Misleading". In: *The American Statistician* 47.1 (Feb. 1993), pp. 65–72. DOI: 10.2307/2684788.

[VWB09] J. VYKOUKAL, M. WOLF, and R. BECK. "Service Grids in Industry – On-Demand Provisioning and Allocation of Grid-Based Business Services". In: *Business & Information Systems Engineering* 1.2 (Apr. 2009), pp. 177–184. DOI: 10.1007/s12599-008-0009-0.

[W3C] W3C. *Web Services Glossary*. http://www.w3.org/TR/ws-gloss/. Accessed 16 July 2013.

[W3C04a] W3C. *RDF Primer*. Accessed 25 September 2013. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[W3C04b] W3C. *Web Services Architecture*. http://www.w3.org/TR/ws-arch/. Accessed 7 September 2013. Feb. 2004.

[W3C05] W3C. *Web Services Choreography Description Language Version 1.0*. Accessed 21 September 2013. Nov. 2005. URL: http://www.w3.org/TR/ws-cdl-10/.

[W3C06] W3C. *Web Services Policy 1.2 - Framework (WS-Policy)*. Accessed 12 March 2014. Apr. 2006. URL: http://www.w3.org/Submission/WS-Policy/.

[W3C07a] W3C. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Accessed 15 September 2013. Apr. 2007. URL: http://www.w3.org/TR/soap12-part1/.

[W3C07b] W3C. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Accessed 15 September 2013. June 2007. URL: http://www.w3.org/TR/wsdl20/.

[W3C12] W3C. *OWL 2 Web Ontology Language Primer (Second Edition)*. Accessed 25 September 2013. Dec. 2012. URL: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[WAB+09] C. WEINHARDT, A. ANANDASIVAM, B. BLAU, N. BORISSOV, T. MEINL, W. MICHALK, and J. STÖSSER. "Cloud Computing – A Classification, Business Models, and Research Directions". In: *Business & Information Systems Engineering* 1.5 (2009), pp. 391–399. DOI: 10.1007/s12599-009-0071-2.

[WCL+05] S. WEERAWARANA, F. CURBERA, F. LEYMANN, T. STOREY, and D. F. FERGUSON. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Upper Saddle River, NJ: Prentice Hall, 2005.

[WCS+08] H. WADA, P. CHAMPRASERT, J. SUZUKI, and K. OBA. "Multiobjective Optimization of SLA-Aware Service Composition". In: *Proceedings of the 2008 IEEE Congress on Services - Part I*. SERVICES '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 368–375. DOI: 10.1109/SERVICES-1.2008.77.

[Weba] WEBRATIO. *IFML: The Interaction Flow Modeling Language*. http://www.ifml.org/. Accessed 2 September 2013.

[Webb] WEBRATIO. *The Web Modeling Language*. http://www.webml.org/webml/page1.do. Accessed 29 August 2013.

[Wes07] M. WESKE. *Business Process Management: Concepts, Languages, Architectures*. Berlin, Heidelberg: Springer, 2007.

[WfM99] WFMC. *Workflow Management Coalition Terminology & Glossary*. WFMC-TC-1011. Accessed 27 December 2015. Feb. 1999. URL: http://www.wfmc.org/docs/TC-1011_term_glossary_v3.pdf.

[WH02] H. WEIGAND and W.-J. van den HEUVEL. "Cross-Organizational Workflow Integration using Contracts". In: *Decision Support Systems* 33.3 (July 2002), pp. 247–265. DOI: 10.1016/S0167-9236(02)00015-5.

[WH06] T. Wilde and T. Hess. *Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung*. Tech. rep. 2/2006. Accessed 31 January 2015. Ludwig-Maximilians-Universität München, 2006. URL: http://www.wim.bwl.uni-muenchen.de/download/epub/ab_2006_02.pdf.

[Win90] J. M. Wing. "A Specifier's Introduction to Formal Methods". In: *Computer* 23.9 (Sept. 1990), pp. 8–23. DOI: 10.1109/2.58215.

[WKI+12] F. Wagner, B. Kloepper, F. Ishikawa, and S. Honiden. "Towards Robust Service Compositions in the Context of Functionally Diverse Services". In: *Proceedings of the 21st international conference on World Wide Web (WWW)*. 2012, pp. 969–978. DOI: 10.1145/2187836.2187966.

[WLB+09] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. "Formal Methods: Practice and Experience". In: *ACM Computing Surveys* 41.4 (Oct. 2009), 19:1–19:36. DOI: 10.1145/1592434.1592436.

[WLH07] H.-C. Wang, C.-S. Lee, and T.-H. Ho. "Combining Subjective and Objective QoS Factors for Personalized Web Service Selection". In: *Expert Systems with Applications* 32.2 (Feb. 2007), pp. 571–584. DOI: 10.1016/j.eswa.2006.01.034.

[WM97] D. H. Wolpert and W. G. Macready. "No Free Lunch Theorems for Optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 67–82. DOI: 10.1109/4235.585893.

[WMS+09] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. "Model-driven Business Process Security Requirement Specification". In: *Journal of Systems Architecture* 55.4 (Apr. 2009), pp. 211–223. DOI: 10.1016/j.sysarc.2008.10.002.

[WP00] G. Wolf and A. Pfitzmann. "Properties of Protection Goals and Their Integration into a User Interface". In: *Computer Networks* 32 (2000), pp. 685–699.

[WW97] C. Wang and W. Wulf. "Towards a Framework for Security Measurements". In: *Proceedings of the 20th National Information Systems Security Conference*. NISSC '97. 1997, pp. 522–533.

[XB05] J. Xiao and R. Boutaba. "QoS-Aware Service Composition and Adaptation in Autonomic Communication". In: *IEEE Journal on Selected Areas in Communications* 23.12 (Sept. 2005), pp. 2344–2360. DOI: 10.1109/JSAC.2005.857212.

[XCZ+08] H. XIAO, B. CHAN, Y. ZOU, J. W. BENAYON, B. O'FARRELL, E. LITANI, and J. HAWKINS. "A Framework for Verifying SLA Compliance in Composed Services". In: *Proceedings of the 2008 IEEE International Conference on Web Services*. ICWS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 457–464. DOI: 10.1109/ICWS.2008.26.

[YCY12] Y. YAN, M. CHEN, and Y. YANG. "Anytime QoS Optimization over the PlanGraph for Web Service Composition". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing (ACM SAC)*. 2012, pp. 1968–1975. DOI: 10.1145/2245276.2232101.

[YH95] K. P. YOON and C.-L. HWANG. *Multiple Attribute Decision Making: An Introduction*. Thousand Oaks, London, New Delhi: Sage Publications, Inc., 1995.

[YL04] T. YU and K.-J. LIN. "Service Selection Algorithms for Web Services with End-to-end QoS Constraints". In: *Proceedings of the IEEE International Conference on E-Commerce Technology*. CEC '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 129–136. DOI: 10.1109/ICECT.2004.1319726.

[YMH+06] J. YU, T. P. MANH, J. HAN, Y. JIN, Y. HAN, and J. WANG. "Pattern Based Property Specification and Verification for Service Composition". In: *Proceedings of the 7th International Conference on Web Information Systems*. WISE '06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 156–168. DOI: 10.1007/11912873_18.

[YPH02] J. YANG, M. P. PAPAZOGLOU, and W.-J. V. d. HEUVEL. "Tackling the Challenges of Service Composition in E-Marketplaces". In: *Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02)*. RIDE '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 125–133. DOI: 10.1109/RIDE.2002.995106.

[YYA09] S. S. YAU, Y. YIN, and H. G. AN. "An Adaptive Tradeoff Model for Service Performance and Security in Service-Based Systems". In: *IEEE International Conference on Web Services (ICWS)*. 2009, pp. 287–294. DOI: 10.1109/ICWS.2009.141.

[YZB11] Z. YE, X. ZHOU, and A. BOUGUETTAYA. "Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing". In: *Proceedings of the 16th international conference on Database systems for advanced applications: Part II*. DASFAA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 321–334. DOI: 10.1007/978-3-642-20152-3_24.

[YZL07] T. Yu, Y. Zhang, and K.-J. Lin. "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints". In: *ACM Transactions on the Web* 1.1 (May 2007). DOI: 10.1145/1232722.1232728.

[ZBD+03] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. "Quality Driven Web Services Composition". In: *Proceedings of the 12th international conference on World Wide Web*. WWW '03. New York, NY, USA: ACM, 2003, pp. 411–421. DOI: 10.1145/775152.775211.

[ZBD+06] J. M. Zaha, A. Barros, M. Dumas, and A. ter Hofstede. "Let's Dance: A Language for Service Behavior Msodeling". In: *Proceedings of the 2006 Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE - Volume Part I*. ODBASE'06/OTM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 145–162. DOI: 10.1007/11914853_10.

[ZBN+04] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. "QoS-Aware Middleware for Web Services Composition". In: *IEEE Transactions on Software Engineering* 30.5 (May 2004), pp. 311–327. DOI: 10.1109/TSE.2004.11.

[ZDT00] E. Zitzler, K. Deb, and L. Thiele. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results". In: *Evolutionary Computation* 8.2 (June 2000), pp. 173–195. DOI: 10.1162/106365600568202.

[Zho07] D. Zhovtobryukh. "A Petri Net-based Approach for Automated Goal-Driven Web Service Composition". In: *Simulation* 83.1 (Jan. 2007), pp. 33–63. DOI: 10.1177/0037549707079226.

[ZK04] E. Zitzler and S. Künzli. "Indicator-Based Selection in Multiobjective Search". In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by X. e. a. Yao. Vol. 3242. Springer, 2004, pp. 832–842. DOI: 10.1007/978-3-540-30217-9_84.

[ZLT02] E. Zitzler, M. Laumanns, and L. Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization". In: *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*. Ed. by K. C. Giannakoglou et al. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[ZT99] E. ZITZLER and L. THIELE. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach". In: *IEEE Transactions on Evolutionary Computation* 3.4 (1999), pp. 257–271. DOI: `10.1109/4235.797969`.

[ZVB13] E. ZIAKA, D. VRAKAS, and N. BASSILIADES. "Web Service Composition Plans in OWL-S". In: *Agents and Artificial Intelligence*. Ed. by J. Filipe and A. Fred. Vol. 271. Communications in Computer and Information Science. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 240–254. DOI: `10.1007/978-3-642-29966-7_16`.

[ZZL10] Z. ZHENG, Y. ZHANG, and M. R. LYU. "Distributed QoS Evaluation for Real-World Web Services". In: *Proceedings of the 2010 IEEE International Conference on Web Services*. ICWS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 83–90. DOI: `10.1109/ICWS.2010.10`.

Index