UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Enabling framework for service-oriented collaborative networks

Sargolzaei, M.

Publication date
2018
Document Version
Final published version
License
Other

Link to publication

Citation for published version (APA):
Sargolzaei, M. (2018). *Enabling framework for service-oriented collaborative networks*.

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:26 Jul 2022

Enabling Framework for Service-oriented Collaborative Networks

Enabling Framework for Service-oriented Collaborative Networks

Mahdi Sargolzaei

Enabling Framework for Service-oriented Collaborative Networks

Mahdi Sargolzaei

Enabling Framework for Service-oriented Collaborative Networks

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op dinsdag 15 mei 2018, te 10.00 uur

door

Mahdi Sargolzaei

geboren te Mashhad, Iran.

Promotor:	Prof.dr. H. Afsarmanesh	Universiteit van Amsterdam
Promotor:	Prof.dr. F. Arbab	Universiteit Leiden
Co-promotor:	Dr. F. Santini	Universit di Perugia
Overige leden:	Prof. Dr. F.C.A. Groen	Universiteit van Amsterdam
	Prof. Dr. M. Worring	Universiteit van Amsterdam
	Prof. Dr. R. Meijer	Universiteit van Amsterdam
	Prof. Dr. P.W.P.J. Grefen	Technische Universiteit Eindhoven
	Dr. M.M. Dastani	Universiteit Utrecht
	Dr. L. Xu	Bournemouth University
	Dr. A. Belloum	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Copyright © 2018 by Mahdi Sargolzaei

Printing production: Off Page, Amsterdam

ISBN: 978-94-6182-890-3

to my little angel, Sarina

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

With the increasing adoption of the Global Village phenomenon, the fundamental aspects of economic systems have exponentially extended from the family, first to the small village and the region, and then growing into the country and nation, and finally to the entire world [20]. As such, over the last several centuries, many businesses have become truly global. Historically, the ancient Silk Road exemplifies one of the first global supply chains in the world [45]. During the recent centuries however, global supply chains have mainly focused on the basic needs for agricultural goods and raw materials. The Industrial Revolution especially, caused further extension of global trades, due to the increase in trade volume, as well as in the capabilities of the sources (regions and nations) in producing their products that finally resulted in the emergence of industrial networks. Over the last decades, the industrial networks and distributed manufacturing got considerably matured and now present themselves as potential solutions to facilitate the fast changing requirements in market demands, i.e. the provision of flexibility and agility [30].

In a nutshell, business globalization has now reached the point that mandates increasing the integration of societies and the synergy in their economies and industries. Consequently, a growing trend has also emerged in service industries, moving towards collaboration and cooperating through participation in networks. In a broad sense collaboration means working together and networking by a group of people and organizations that are closely connected. Although the notion of network is addressed in many fields, such as in communications, social sciences, computer science, biology, etc., the special relevance in the context of our research work is the area of Collaborative Network (CN), and mostly pertinent to organizations. A set of definitions addressing different kinds of collaborative networks are addressed in [32]. For example, the Collaborative Networked Organization (CNO) is defined as a structured form of organizations that follow a predefined set

of governance principles and rules, while the Ad-hoc Collaborations are defined as those spontaneously formed, without any precise structure and predefined targets, and that are not business oriented. There are also two specific forms of CNOs closely applied in our research, namely: the long-term strategic networks and the goal-oriented networks. One instance of the goal-oriented collaborative network especially is called the Virtual Organization (VO), which is aimed at achieving its members' common business goals. The definition of a VO as adopted in our research follows [31]:

"*Virtual Organization (VO) is a dynamic and temporary form of collaborative networks, comprising a number of independent organizations that wish to share their resources and skills to achieve its common mission/goal.*"

One instance of the long-term strategic network especially is called the Virtual organizations Breeding Environment (VBE), which is established with the purpose of providing an environment and the conditions to support rapid configuration of the virtual organizations. The formal definition of a VBE as adopted in our research follows [1]:

"*VO Breeding environment (VBE) represents an association of organizations and their related supporting institutions, adhering to a base long term cooperation agreement, and adoption of common operating principles and infrastructures, with the main goal of increasing their preparedness towards rapid configuration of temporary alliances for collaboration in potential Virtual Organizations.*"

With rapid advances in ICT, nowadays collaborative networks are supported by a large set of diverse tools, including broadband mobile computing and cloud computing that further push early concepts of collaborative networks into much wider and completely new territories. In many areas of production and services, Small and Medium-sized Enterprises (SMEs) provide their business services online. SMEs are increasingly interested in working together and establishing collaborative networks , through joining their skills, resources, abilities, and knowledge. Usually a VO is established by SMEs to fulfill the following two purposes.

A first purpose for VO formation is to best target a specific emerged opportunity in the market or society, which either requires the combination of different capabilities and resources provided by collaborating organizations, or simply requiring the accumulation of their resources and/or capacities. The formation process of a VO typically starts in a VBE, where the candidate SMEs are usually selected by the VO broker and invited to jointly accept the responsibility of fulfilling the tasks needed to achieve the common goals of the VO.

A second purpose for VO formation is to support innovation. For instance, one or more SMEs together foresee the potential of investing into the development and

provision of a certain new service in the market, and one acts as the VO broker, to combine some of the abilities, resources, capacities, etc. from a number of participating SMEs, who can then together fulfill the development of the planned innovation.

Nevertheless, in either case, in order to act agile and to be able to compete in the market and society against the real large existing organizations, a minimum base platform for collaboration must pre-exist among the SMEs before the formation/operation of the VO. For this purpose, the Virtual organizations Breeding Environment serves as the base environment that provides this minimum collaboration base, within which the VOs can then be launched.

In order to collaborate effectively and to become time/cost efficient, SMEs in the VO must act together as a single larger entity, and thus sharing with each other their resources and capabilities, and co-working efficiently. One important set of resources to be shared among such SMEs in the VO consists of the pool of their online business services. These services must be shared as if all involved SMEs belong to one larger real organization. These services must also be integrable, to support the creation of value-added services within the VO.

Clearly, in order to share and compose software services that represent business services provided by different SMEs in a VO, they need to be formally and uniformly defined. In fact, such unification in their definition format must be properly addressed and pre-defined at the VBE level, in order to enhance their sharing and collaboration among SMEs, once the VO is established. In other words, when all VBE members formally and uniformly define their business services, once in a VO, all partners will be able to seamlessly share each others services and to integrate them for creating new value-added services and/or for innovation purposes in their sector, as if these SMEs all belong to one large organization. At present however, due to the lack of uniformity in full and formal definition of implemented business services, as provided on-line by SMEs, neither their effective discovery as existing software services, nor their effective composition toward creation of integrated services, can be supported. For instance today, creating a new value-added service out of the existing services within the VOs, to provide it as a new online service to the customer, is quite challenging in the needed efforts, and the time and cost spent, as also exemplified below [4].

As an example, consider a VBE in which a partner SME-1 plans to simply create a new integrated tourism package to include the reservation of flight-tickets, hotel-rooms, day-trips, and dinners at restaurants in a specific touristic region. In an ideal situation, all SMEs involved in the VBE would have already implemented and provided their individual business services as shared web services. As such, SME-1 could have discovered all needed services and identified the most-fit SMEs for sharing their services and working together to create this new package as an integrated service. SME-1 could then also act as the VO broker, start the formation of the new needed VO, and invite the other SMEs for it. But at present, due to the lack of such uniform and unambiguous formal definition for

the provided business services through the VBE, SME-1 can neither be properly supported with the discovery of such existing services, nor with facilitating their composition. In other words, currently even the discovery task of the most-fitting services can at best perform a search/match based only on the service's capability and interface, including the service names, operation names, and a few other additional information source related to the functionality of the requested service, e.g. in terms of its preconditions, assumptions, post-conditions, and effects [166].

However, in order to effectively co-work and co-develop in VOs, organizations need to both discover much more than this about the VBE's shared services and be assisted with their potential composition. As a first step, the shared business services in the VBE need to be concisely specified and accessible through a virtual common pool of uniformly defined services. As such, organizations need to accordingly register/publish their services in this virtual common pool, searchable through the directory. When achieved, SME-1 would be able to discover the needed web services for the tourism package such as flight-tickets, hotel-rooms, and day-trips. These services in fact need to be retrieved from the common pool based on their functional and non-functional properties. For example, SME-1 can investigate/discover services based on which operations they invoke and in which sequence, i.e. based on our so-called behavior of the service. Finally, SME-1 as a service integrator needs support to bundle its selected services in order to offer the tourism package as a new composite service.

While the above is needed and desired in service providing VOs, the following business service related challenges must be first tackled and resolved before they can be supported. The main challenges identified and addressed by our research include the following:
(1) There is no uniformity in business service definition among organizations, since organizations are and remain independent and autonomous.
(2) There is a lack of common ontology for definition of business services, even at the level of existing associations/clusters of organizations related to each industry sector.
(3) There is no unambiguous/concise specification for business services, specifically for service functionality, as it is vital to developing their equivalent software services.
(4) There is a lack of support for selecting most-fit business services against the user-defined desired criteria, in relation to the syntax, semantics, and function/behavior of the service.
(5) There is no support for business service integration/composition, to allow creating value-added services in the VOs.

This thesis addresses the above challenges as its main contribution. Our approach goes beyond the existing business services (BSs) standards and tools through proposing a competency model for business services within the VOs. The proposed BS competency model represents a number of characteristics that are

needed for the description of BSs as web services. As such, VO business services can be uniformly and concisely defined and published in order to support their sharing and reuse. The thesis also defines an approach and the needed system architecture supporting the semi-automation of its proposed solution to the last critical challenges.

1.2 Research Questions

Uniform definition for business services and specially the concise specification of their behavior are of great importance for co-working in VOs. Semi-automation of business service manipulation tasks e.g. the service composition tool, greatly enhances the effective co-development within service industry. Our proposed approach and system is developed to properly respond to the following fundamental research questions:

RQ1: How to support VO member organizations to co-work/co-develop through sharing their capabilities when defined as business services?

This research question primarily requires the design of a holistic framework to support service oriented VOs and is addressed in Chapter 2 of the thesis. In order to address service oriented collaborative networks, first we extend the traditional architecture of the SOA in support of the identified needs, challenges and new opportunities raised in collaborative networks area. We introduce a new variation to the SOA paradigm for this purpose, and thus propose a framework to support establishing service oriented computing in VOs. At the abstract level, our framework consists of three software modules, including: **Specification Module**, **Discovery Module**, and **Composition Module**, as are further explained in Chapter 2.

RQ2: How to effectively support service reusability in service-oriented VOs?

This research question is related to the specification and representation of business services that are mainly addressed in Chapter 3. It includes the following two sub-questions:

S1-RQ2: How to uniformly define business services from independent and autonomous organizations participating in the VO?

Our concise definition of business services is supported through the C3Q model, addressed in Chapter 3. This model addresses the **Capability** (i.e. addressing functional properties of services including their syntax, semantics, and behavior),

Cost, **Conspicuity**, and **Quality criteria** of the service, each defined through its shared common ontology. As such, all VO business services can be uniformly published in the common VBE directory, and efficiently shared and reused by all VBE members.

S2-RQ2: How to specify in an unambiguous/concise representation manner, the behavior/functionality of business services, as required for developing their equivalent software services?

This research question is also addressed in Chapter 3, through the formal representation introduced for service behavior. This representation can sufficiently support generating machine readable specification of business processes that run at each organization in the VO. As such, the defined **service behavior** can be reused unambiguously, for developing equivalent software services, and thus providing an executable definition for business services.

RQ3: How to enhance discovery/selection of most-fit business services in VOs, while supporting a number of user-specified criteria?

This research question is primarily related to provision of the required framework and the implementation of the required architecture and mechanisms for automated **discovery** of component services. The discovery task involves proper matchmaking between the characteristics of existing services against their desired user-defined **syntax**, **semantics**, **behavior** and **quality criteria**. This is mainly addressed in Chapter 4, which tackles the following two sub-questions:

S1-RQ3: How to enhance the search criteria, thus improving accuracy of service discovery results?

The power and efficiency of the **discovery module** here, similar to any other information retrieval method, is bound to the number of relevant accurate results that it generates. A relevant result in discovery cases, is a registered service that is potentially able to do what user requires, i.e. it is functionally matched to the user query. Many approaches have tackled this concern and even some of their related tools have achieved good results for service discovery based on searching the syntactic and semantic properties of web services. However, to the best of our knowledge, none of the provided approaches consider searching also based on the behavioral properties of services, which can play a very important role in search results. To deal with this deficiency, we introduce a tool for **behavior-based Discovery** of matched services, in which also the requested behavior specified within the user's query is taken into account.

S2-RQ3: How to ensure the quality aspects of retrieved/discovered services?

While the proposed search engine can discover services according to their functional properties, non-functional properties (especially quality parameters of services) also play an important role in users selection. In other words, when several web services offer similar capabilities, it is necessary to also consider the non-functional properties of services carefully as selection criteria. Thus, besides the functional requirements, our tool supports users to also consider the set of **QoS (quality of services)**requirements in their queries in order to assist with **service selection** in a **lexicographic order**.

RQ4: How to semi-automate integration/composition of component business services in VOs, to support the creation of value-added services and co-innovation?

This research question is addressed in Chapter 5, in which enhancing service integration and semi-automated composition of the existing published services in the VBE are addressed. In other words, reusability of business services existing in the virtual VBE service pool, and the creation of new value-added services in the VOs are supported by our introduced approach and implemented mechanisms. This topic is addressed through the following two sub-questions:

S1-RQ4: How to represent internal configuration (i.e. behavior) of the component services, as required to be concisely defined for the purpose of their automated invocation?

Stateful services, with more than one internal configuration, may permit the exchange of messages of different types in their different configurations. To approach this challenge, we propose the use of **web service behavior Specification (WSBS)** in terms of constraint automata, in order to show the desired configuration of the defined service, i.e. the sequence of its operations' invocations, and to call the operations according to it. Our proposed tool can then monitor the transitions of the WSBSs to follow the proper configurations of the service. It then packs both the data items and the synchronization points (of the orchestrator) into some appropriate message format, and then sends these messages over the network to the actual service for its proper invocation when needed.

S2-RQ4: How to model the complex interactions and communications among several component services as required to be defined for their proper integration into a composite service and how to semi-automate the process?

With the added challenge of executability aimed for this task, a software coordination language was needed to be adopted as the base. We studied many

coordination languages, and chose to adopt Reo [9] as the most suitable base language and environment to both model and handle the interaction protocols among the component services. We present a compositional construction of web services in this chapter, while using Reo and constraint automata. For each web service, our approach automatically generates a **proxy** that then manages the communication between this service and the **Reo connectors**. This proxy component is therefore in charge of managing the communication between the technology behind each service with the Reo environment.

Obtaining satisfactory answers to the above sub-questions have resulted in developing our tool for specification, discovery, and composition of business services and effectively support and promote collaboration, competition, and co-innovation among a network of organizations in the VOs.

1.3 Research Method

This section addresses the main steps and the methods followed in our research and addressed in the thesis.
(1) Establishing the Motivation. This phase aims at indicating the importance of developing a framework for service oriented collaborative networks. We show how the proposed model is applied in a network of organizations, enables the members to work as a network more effectively, and increase their collaboration and even innovation. We address the shortcomings of the current supporting standards and tools that target employing software services in collaborative networks. These aspects serve as the motivation for this dissertation research, and are introduced in Section 2.4.1. These problems and required new elements to solve them are addressed through our introduction of a variation of SOA paradigm, as presented in Section 2.4.2.
(2) Reviewing the Related Work. This step targets providing enough information for the approach pursued in this thesis, to specify, reuse, and integrate business services shared within the networked organizations. Our study of related work aims to provide solid understanding of the theoretical prospectives involved and exploration of challenges, opportunities, and practicalities related to service industries.
(3) Proposing Innovative Framework. This phase aims at introducing an abstract framework to support service oriented collaborative networks. Applying SOA in VOs, we present a variation of the traditional SOA paradigm, which is used to augment existing web services standards and tools, in order to overcome their drawbacks for our purposes. Chapter 2 addresses our introduced abstract framework, which is designed based on a new SOA paradigm, and used in subsequent chapters for the implementation of our supporting business service manipulation tools.
(4)) Developing Tools to Support the Framework. This step targets the

development of a set of needed tools to support the designed abstract framework for service oriented collaborative networks. A GUI is implemented to ease the behavioral specification of services, and which extends the WSDL standard. Moreover, a tool is developed for similarity-based discovery of services and in order to rank the service descriptions in our registry. Finally, a code generator for the orchestration of services in Reo [9] is implemented. The tool automatically generates a proxy for each service that manages the communication between this service and the orchestrator. These tools are implemented in Java, and their implementation details are respectively addressed in Chapters 3, 4 and 5.

(5) Evaluating the Proposed Approach. The goal of this phase is to evaluate the efficiency and correctness of our proposed approaches. In Chapter 6, we evaluate each approach (i.e. **XWSDL**, **BehSearch**, and **ProxCG**) by comparing them against several state of the art related work. Moreover, the correctness of our search results according to the information retrieval metrics, as well as its efficiency based on the scalability factors, are measured in this chapter.

(6) Validating with a Case Study. The validation phase aims at studying the effects of the proposed approaches and tools on several case studies. We apply some models and approaches depended in this dissertation to the **GloNet** project, which is a European founded research project aimed at supporting development of business service enhanced products in a collaborative network focused in the area of solar power plants. This part of the research is also presented in Chapter 6.

1.4 Thesis Structure

This thesis addresses mechanisms that are used to support the reusability and integration of business services in VOs. The structure of the thesis is as follows:

In **Chapter 2**, an abstract framework is specified to support collaboration within a network of organizations, through sharing their business services. In comparison with traditional collaborative networks, this style of CNs, i.e. service oriented collaborative networks, promotes and simplifies reusability and interconnection of shared assets especially in the information technology sector, i.e. regarding the handling of software services in a distributed manner. Our framework is designed based on a new proposed variation of Service Oriented Architecture (SOA) paradigm, which is applied in collaborative networks. This chapter further provides a short survey that enables effective categorization and comparison of different state of the art approaches addressed as related works.

Chapter 3 addresses and resolves several main obstacles and challenges existing for the definition and specification of business services, in order to provide a basis for discovery and composition of these services. This chapter briefly outlines some theoretical and technical assumptions made by works related to business service specification. A brief description of business services is proposed in this

chapter and then we sketch out our own position to provide a specification model for the VO business services called C3Q. A light extension of WSDL standard is also presented to describe different aspects of the business services, as they are considered in C3Q. Finally, a GUI is developed to assist users in specification of services.

In **Chapter 4**, we present a tool that is able to discover web services registered in a database and to rank them according to a similarity score that expresses the affinities between each service and a user-submitted query. To determine these affinities, both the behavior of the user's query and the behavior of the services are taken into account and compared. The name of service operations, their desired order of invocation (i.e. the service behavior) and their parameters, form the functional similarity score for the discovery. This information is expressed and formalized in the user's queries by soft constraints, to accommodate the user's needs in the best possible way. The behavior-based specification and discovery is proposed for the stateful services, and we argue that a proper formalization for the behavior of many services that are commonly thought to be stateless, in fact do require a stateful representation. As such, our method and our tools can accommodate discovery of these services better than the alternatives that consider them as stateless. The discovery process is modeled as a Constraint Optimization Problem. We also enhance our discovery tool, considering QoS metrics to further meet the non-functional needs specified by the users.

The focus of **Chapter 5** is on the composition of services in our framework, in which services distributed over a network of organizations can be composed according to the requirements defined by a service integrator. The composition of services needs additional efforts to impose coordination among the component services. We study and compare several alternative coordination languages and standards that can model the interaction protocols among component services in an integrated/composite service. We pick Reo [115] as the coordination language to use, and recall the most relevant aspects of Reo for the sake of our VO service composition, as the necessary background notions.

In **Chapter 6**, we first validate the proposed approaches and tools through applying our tools in the European R&D project GloNet [36] that involves real cases in the context of service enhanced products in solar power plants. We then evaluate our proposed approaches (including XWSDL, BehSearch, and ProxCG) by comparing them to several related work results. Furthermore, we also measure the correctness of our discovery tool by calculating its precision and recall. Finally, we present a peer-to-peer implementation of the tool to overcome the scalability issues.

The **Chapter 7** concludes the thesis and addresses how we have answered to the defined research questions. The assessment of our approaches and the developed tools, as well as several on going and future works are also addressed in this chapter.

1.5 Main Contributions

The main contributions of this thesis is in the field of handling business services specification, discovery, and composition within collaboration networks, as presented in Table 1.1.

Table 1.1: The main contributions of this disertation

Title	**Description**	**Place**
SOCN	A framework that addresses the needed modules to implement service oriented collaborative networks	Chapter 2
C3Q	A competency model for business services specification	Chapter 3
XWSDL	An extension of WSDL to describe web services, based on the C3Q model	Chapter 3
Fizzim+B	A GUI to ease behavioral specification of services, through extending the Fizzim tool	Chapter 3
BehSearch	A behavior-based services discovery	Chapter 4
ProxCG	A tool for automated generating of proxies, able to connect web services to Reo circuits as the orchestrators	Chapter 5

The material represented in Chapters 2 to 6 of this thesis have been published in a series of co-authored Journal articles and conference papers, as indicated below.

- Service Oriented Collaborative Network Architecture [142].
  - Partially presented in **Chapter 2**.
  - Published in *PRO-VE Conference Proc., 18th Conference on Collaboration in a Data-Rich World*, Springer, Berlin, Heidelberg, 2017.
  - **Mahdi Sargolzaei**: Addressing a variation of the service oriented architecture paradigm for collaborative network. Designing a new framework to implement service oriented collaborative networks.
  - Hamideh Afsarmanesh: Guidance and technical advice.
- Semi-automated software service integration in virtual organisations [4].
  - Partially presented in **Chapter 2** and **Chapter 5**.

  - Published in *Journal of Enterprise Information Systems, Volume 9, Issue 5-6: Service-based interoperability and collaboration for enterprise*, Taylor & Francis, UK, 2015.
  - **Mahdi Sargolzaei**: Proposing a meta-date to define business services unambiguously in VOs. Discussing state of the art in the area of service oriented computing. Developing a framework semi-automated software service integration in VOs.
  - Mahdieh Shadi: Proposing approach for trust evaluation.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- A framework for automated service composition in collaborative networks [3].
  - Partially presented in **Chapter 2**.
  - Published in *PRO-VE Conference Proc., 14th Conference on Collaborative Networks in the Internet of Services*, Springer, Berlin, Heidelberg, 2012.
  - **Mahdi Sargolzaei**: Designing a conceptual framework to facilitates automated service composition in VOs.
  - Mahdieh Shadi: Proposing approach for trust evaluation.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- C3Q: A Specification Model for web services within Virtual Organizations [141].
  - Partially Presented in **Chapter 3**.
  - Published in *PRO-VE Conference Proc., 18th Conference on Collaboration in a Data-Rich World*, Springer, Berlin, Heidelberg, 2017.
  - **Mahdi Sargolzaei**: Designing a new competency model for business services in VOs. Extending WSDL standard based on C3Q.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- A Tool for behaviour-Based Discovery of Approximately Matching of web services [144].
  - Partially presented in **Chapter 4**.
  - Published in *SEFM Conference Proc., 11th International Conference on Software Engineering and Formal Methods*, Part of the Lecture Notes in Computer Science book series (LNCS, volume 8137), Springer, Berlin, Heidelberg, 2013.

  - **Mahdi Sargolzaei**: Designing an architecture for approximate behaviour-matching of web services. Implementation of the tool.
  - Francesco Santini: Providing several of the formal definitions.
  - Farhad Arbab: Guidance and technical advice.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- Automatic Code Generation for the Orchestration of web services with Reo [83].
  - Partially presented in **Chapter 5**.
  - Published in *ESOCC Conference Proc., European Conference on Service-Oriented and Cloud Computing*, Part of the Lecture Notes in Computer Science book series (LNCS, volume 7592), Springer, Berlin, Heidelberg, 2012.
  - **Mahdi Sargolzaei**: Designing and implementing the service side of the Proxies.
  - Sung-Shik T. Q. Jongmans: Designing and implementing the circuit side of the proxies.
  - Francesco Santini: Providing several of the formal definitions.
  - Farhad Arbab: Guidance and technical advice.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- Orchestrating web services using Reo: from circuits and behaviours to automatically generated code [84].
  - Partially presented in **Chapter 5**.
  - Published in *Journal of Service Oriented Computing and Applications, Volume 8, Issue 4: service-oriented and cloud computing*, Springer, Berlin, Heidelberg, 2014.
  - **Mahdi Sargolzaei**: Designing and implementing the service side of the Proxies. Working on the case studies in order to validate the tool.
  - Sung-Shik T. Q. Jongmans: Designing and implementing the circuit side of the proxies.
  - Francesco Santini: Providing several of the formal definitions.
  - Farhad Arbab: Guidance and technical advice.
  - Hamideh Afsarmanesh: Guidance and technical advice.

- A Competition Space for Complex Product Specification [152].
  - Partially presented in **Chapter 6**.

- Published in *PRO-VE Conference Proc., 16th Conference on Collaborative Systems for Smart Networked Environments*, 2014.
- **Mahdi Sargolzaei**: Designing and implementing the Service specification tool(SST), and service discovery module of product/Service Discovery and Recommendation (PSDR).
- Mohammad Shafahi: Designing and implementing the product specification (PST) and Product/Service Discovery and Recommendation (PSDR).
- Hamideh Afsarmanesh: Guidance and technical advice.

Chapter 2

Service Oriented Collaborative Networks - SOCN

The research results presented in this chapter are partially published in the following papers:

- Sargolzaei, M. and Afsarmanesh, H., 2017. Service Oriented Collaborative Network Architecture. In Collaboration in a Data-Rich World. Springer Berlin Heidelberg.
- Afsarmanesh, H., Sargolzaei, M. and Shadi, M., 2015. Semi-automated software service integration in virtual organisations. Enterprise Information Systems, 9(5-6), pp.528-555.
- Afsarmanesh, H., Sargolzaei, M. and Shadi, M., 2012. A framework for automated service composition in collaborative networks. In Collaborative Networks on the Internet of Services (pp. 63-73). Springer Berlin Heidelberg.

2.1 Introduction

In today's economy, cooperation among organizations tend to change from traditional static supply chains to dynamic organization networks [158]. In general, networking organizations and co-development among business partners brings forth lower cost, higher quality service/product, larger service/product portfolio, faster delivery, and more agility. However, the pace at which these changes in trend need to occur has resulted in high demands in supporting collaboration through networks of organizations, i.e., establishing collaborative networks (CNs). In another simultaneous emerging trend, organizations constantly search for new ways to improve their business processes and to enrich collaboration among their potentially distributed workers [81]. In our research work, the above

two challenges are addressed, and the promising paradigm of Service Oriented Architecture (SOA) is investigated and applied to the enhancement of organizations collaborative networks. Besides proposing the use of SOA in effective support of CNs, we needed to extend the generic model of SOA as the reference framework. An implementation architecture is also elaborated based on our analysis of the requirements from the reference framework.

This chapter specifically aims at introducing our proposed new framework to support collaboration within a network of organizations, through their shared business services (BSs). Applying this framework facilitates Service Oriented Collaborative Networks (SOCN) by promoting and simplifying reusability and interconnection of shared software services, in a distributed manner. Furthermore, a set of sub-systems is designed within an implementation architecture that supports all SOCN's required functionality.

This chapter starts by providing a short introduction to the Service Oriented Architecture in Section 2.2. In Section 2.3 the role of SOA in today's organizations is discussed. An extended model of SOA paradigm is addressed in Section 2.4 to identify the basic requirements for supporting SOCN. A reference framework and architecture to facilitate assisting the implementation of our proposed model is presented thereafter, identifying and briefly describing its main components. We then conclude the chapter in Section 2.5.

2.2 Service Oriented Architecture

Recently, Service Oriented Architecture (SOA) has become well-known while somewhat imprecisely defined. For example, an organizational methodology to design systems, an IT infrastructure in business, and a structure for increasing the efficiency of IT, are some of the definitions provided for SOA from three different points of view. In fact, this problem with defining SOA reminds the author of a poem by Rumi titled: "The blind men and the elephant" [146] about the men examining an elephant in the dark, where each man depicts the elephant different than the others, according to his own individual experiences. Similar to these men, people from different contexts have correctly identified many capabilities of SOA, but they almost failed to communicate this concept as a whole.

In our research, as a preliminary definition, we consider SOA as a loosely-coupled architecture designed to meet the business requirements of the organizations. It means that the dependency among services is minimized, while they only require being aware of each other. Being open, extensible, federated, and composable are the characteristics we adopt for SOA, as formally defined by Erl in [56], promoting service-orientation in enterprises. From his point of view, services in SOA are autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable, which are implemented as web services.

Organization for the Advancement of Structured Information Standards (OA-

SIS) also provides a holistic definition of SOA that we consider in our research as follows:
"Paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" [106].

Indeed, Service Oriented Architecture is a programming paradigm that uses "services" as the constructs of distributed business applications to support reusability and agility. Services in this architecture are autonomous and platform - independent computational entities that can represent the steps of a business process and communicate with each other [126]. These services can be described, published, discovered, and dynamically integrated in order to develop massively distributed, interoperable, and evolvable applications. Each service can perform some functions that are able to execute either simple requests or complex business processes through a peer-to-peer relationships between service clients and providers.

The subject of Service Oriented Architecture and Computing covers many concepts, standards and technologies that find their origins in a wide variety of disciplines, including distributed computing systems, computer networking, computer architectures, cloud computing, software engineering, programming languages, database systems, security, artificial intelligence and knowledge representation. An explanation of these related concepts is beyond the scope of this thesis.

2.2.1 Applying SOA

For a long time, increasing the level of abstraction in programming has been a main motivation in software engineering. Therefore, we have witnessed evolutions in programming approaches from sequential programming, respectively to procedural programming, object-oriented programming, component-based programming, and finally service oriented programming. Procedural programming has been defined as a subtype of imperative programming that moves some statements into procedures. During the 80s, object-oriented programming (OOP) was introduced as an extension of procedural programming that increases the level of abstraction and encapsulation of the programs by defining collections of individual units called objects that form the programs. Then, in the 90s, the component-based development models come to the programming paradigm. Finally, components in the programming model are enhanced to the services. There is no clear dividing line between component oriented programming and service oriented programming. In fact, services are the enhancement of components, where the individual services can be considered as single components. Both SOA and component based architecture support abstraction and loose-coupling. The main difference between SOA and component based architecture is related to the

connection between services, and also the ability to offer single services for third parties [130]. In principle, SOA provides the services for third parties on the Internet, therefore software developers are able to use foreign, external "components" in the form of web services. Figure 2.1 shows the timeline of increasing abstraction level in programming paradigm from procedural methodology to SOA.

Service oriented programming holds the promise of moving beyond the simple exchange of information and remote invocation of methods on objects, to the concept of encapsulating data and application to services and passing messages between them. An important economic benefit of this shift in programming paradigm is that it improves the effectiveness of software development activities and enables enterprises to bring their new applications to the market more rapidly and cost-effectively than ever before, by developing composite services from the existing component applications [108] and [126].

2.2.2 Web services

Currently, web services are the most promising technology that implements the concept of SOA, and provides the basis for the development and execution of business processes that are distributed over the Internet. A web service is defined as a self-contained, modular, loosely coupled, reusable software application that can be described, published, discovered, and invoked over the world wide web [135], [130] and [42]. They benefit from several Internet-based standards, such as web services Description Language (WSDL) for service description of services, and Simple Object Access Protocol (SOAP) to exchange messages and communicate independently from the adopted platform [42]. The key-role in these standards and protocols is their XML-based structures that ensure independence from implementation and platform.
A web service consists of three main roles: Service Provider, Service Client, and Service Repository.

- **Service Provider** implements the web service and makes it available on the Internet. The service provider should define an abstract service description using the WSDL, and then register it in a registry or repository of services.

- **Service Repository** is a logically centralized directory of services that allows a web service to be registered by the service provider and discovered by a service client. It returns a Uniform Resource Identifier (URI) as a result of the search to service requester/client that points to the service itself, and client can then use this information to bind to the web service and invoke it.

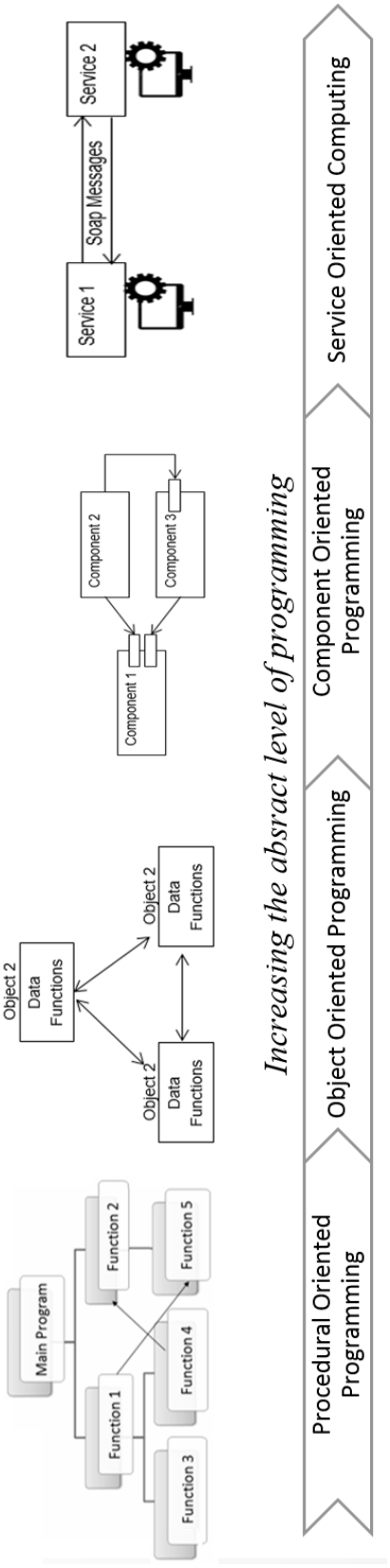


Figure 2.1: The history line of programming.

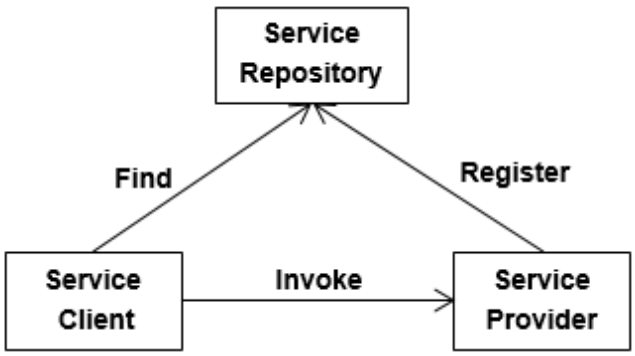


Figure 2.2: SOA paradigm for web services.

- **Service Client** is any consumer of the web service that uses the repository to find his/her desired service. The service client has to use and access the WSDL document to select and invoke the web service.

Figure 2.2 shows the interactions among Service Provider, Service Client and Service Repository, which is the principle of the usage of web services. Besides standard web services, RESTful [94] is another implementation of a service, which is the acronym of Representational State Transfer. There is a unique URI for each RESTful service that is basically a representation of some objects. Standard web services use a XML-based message protocol (SOAP) for communication between the service provider and client, while RESTful services use JSON or XML (without any specific protocol) to send and receive data after calling the URI path.

2.3 Migrating to SOA-based organizations

As a very promising methodology, SOA can support business processes of organizations. Applying the service oriented architecture paradigm and technologies in organizations leads to reduction of complexity and costs in reusing business supported functionality and operations, and increase in flexibility and efficiency [126]. These advantages allow the organizations to adapt more easily to the needed changes, and it is therefore expected that the SOA paradigm improves the efficiency of organizations more than other previous approaches and technologies. It provides a logical approach to designing component-based applications that can serve either the end-users or the other applications in a network. As such, SOA can empower the business and organizational environments by offering independent, reusable automated business processes that can be materialized through business services.

Figure 2.3 shows a trivial example of applying a *SOA-based* infrastructure across organizations' architectures. Enterprise-1 consists of a service provider that

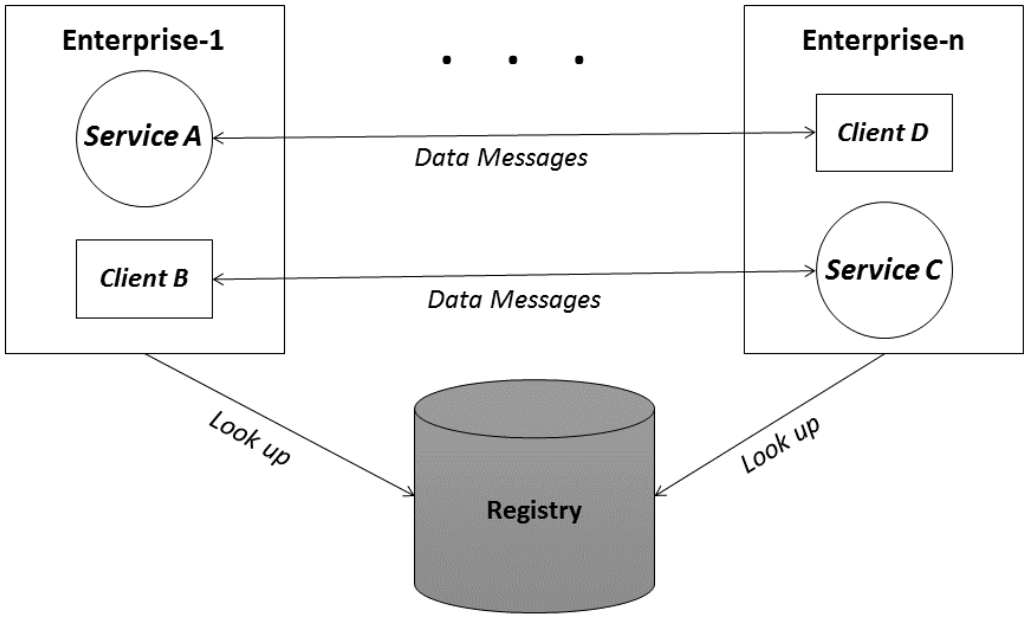


Figure 2.3: A sample of SOA setup between organizations.

offers *ServiceA*, and a service client labeled as *Client B*, similarly, Enterprise-n consists of a service provider that offers *Service C*, and service *client D*. Different organizations (e.g., *Enterprise-1* and *Enterprise-n*) publish their services in a service registry, and then using the registry, clients at enterprises can look up service details in order to select to execute them. In the context of web services, service details is specified in WSDL and the data messages exchanged between the enterprises are in the form of SOAP. This simple example shows how enterprises interpretations can be obtained through SOA. These interpretations may occur between different organizations or even within one large organization. In fact, SOA offers new and flexible ways to develop tools for supporting the activities both inter-organizations and intra-organization. Thus, two types of usages, the Intra-Organizational and Inter-Organizational platforms, can utilize such service oriented computing (SOC) tools.

Considering the example of a large enterprise. Each single department can share its services through a common repository, and offer them for other departments to access and use for different purposes, and even use them for a different purpose than what the business service was originally built for. So, this usage of SOC tools could be seen as Intra-Organizational application of SOA. The other type of use is for Inter-Organizational platform, as in the networks established among different organizations as stakeholders in a VO. Each VO partner can then announce and register its services within the collaboration space, i.e., a directory of shared services, in order to make them identifiable and accessible by other VO partners. Figure 2.4 shows these two kinds of SOA-based organizations.

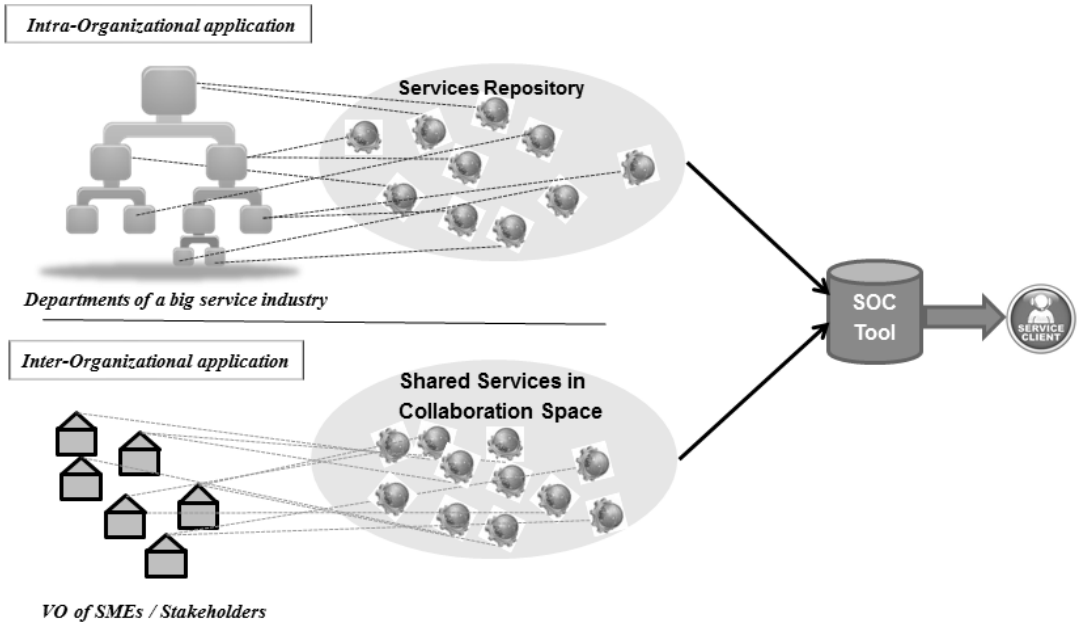


Figure 2.4: Two kinds of SOA-based organizational applications.

2.4 Towards Service Oriented Collaborative Networks - SOCN

In a large number of application areas, such as tourism, insurance, and e-commerce, an increasing number of SMEs (Small and Medium Enterprises) are increasingly formalizing the definition of their business services. This is primarily due to the fact that further to the individual SMEs interested in providing their business services on-line, in many areas of production and services, SMEs are increasingly interested in working together and establishing collaborative networks (CNs), through joining their skills, resources, and knowledge. VO is one form of CN, which is usually established to fulfill the following two purposes. One purpose of VO formation is to best target a specific opportunity emerging in the market or society, which requires either the combination of different capabilities and resources, provided by a number of different organizations, or simply the accumulation of their resource capacities, or both. The candidate SMEs are then usually selected by the VO broker and invited to accept the joint responsibility of fulfilling the needed tasks to achieve the common goals of the VO. The second purpose of VO formation is to support innovation. For instance, one or more SMEs together foresee the potential of investing into the development of new services, and act as the VO broker, targeting the merge of abilities, resources, capacities, etc. from a number of SMEs who can then together fulfill the development of the planned innovation.

Nevertheless, in either case, in order to act agile and be able to compete in the

market and society against the real large existing organizations, a minimum base platform for collaboration among SMEs must pre-exist before the formation/operation of the VO. For this purpose, Virtual organizations Breeding Environment (VBE) is usually established, providing this minimum collaboration base, within which the VOs can then be launched.

In order to collaborate effectively and to become time/cost efficient, SMEs in the VO must together act as a single larger entity, and thus sharing all their resources and capabilities. An important set of resources to be shared among the SMEs consists of their business services that are provided on-line, which must be sharable as if they all belong to a larger real organization, and must be integrable for creation of value-added services in the VO. But such unification in the definition format must be addressed at the VBE level, in order to enhance the sharing and collaboration among SMEs once the VO is established. In other words, when the VBE members formally and uniformly define their business services, once in a VO, all partners will be able to share these services and to integrate them for creating new value-added services, and/or for innovation purposes in their sector. At present however, due to the lack of uniformity in the full and formal definition of the implemented business services, which are provided on-line by SMEs, effective discovery of the existing software services and their composition toward creation of an integrated service are not supported. For instance, today creating a new value-added service out of the existing services in the VOs, and providing it as a new online service to the customer is quite challenging. In other words, currently, the discovery of most-fitting services can at best search/match based on the service names or perhaps some semantics related to the requested service, while our approach goes beyond that.

In this thesis, we aim to address this specific challenge and define an approach and system architecture to support the semi-automation of this challenging task that presents the contribution of this research. For this purpose, a new framework is proposed in this chapter, based on which a tool is developed to support service integration in VOs. The approach covers four aspects, including service modeling/provision, service registering, service discovery, and service composition/integration. Except for the modules planned as building blocks of the Service Provision, all other modules present in the architecture are implemented and running. Furthermore, suitable mechanisms are introduced for discovery and selection of the most-fit services, matching users' requested criteria.

We believe that applying the SOA paradigm to collaborative networks ensures a high level of abstraction for data and operations in the form of business services. Higher level of abstraction makes integration with functional capabilities (i.e., services) easier.

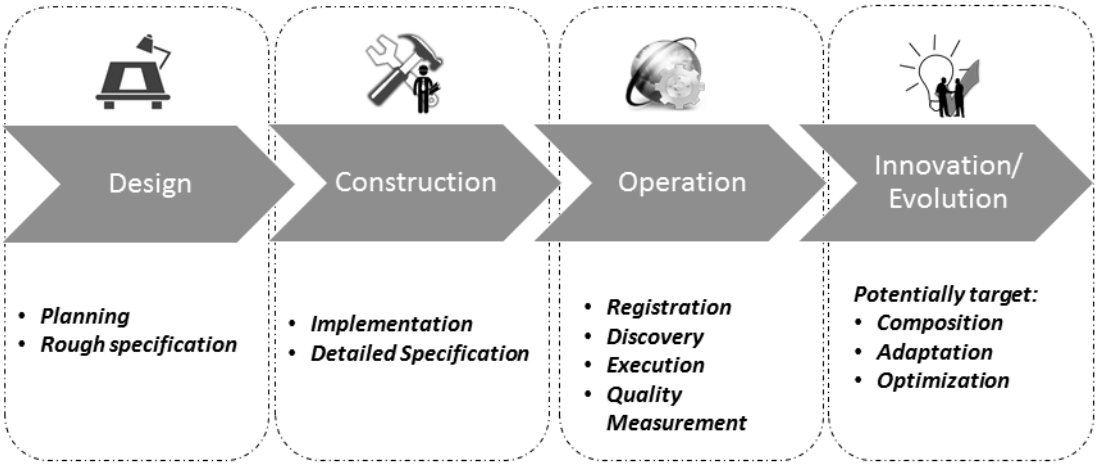


Figure 2.5: Service life cycle's phases.

2.4.1 Needs and Challenges

Business services are naturally dynamic; therefore, the supporting tools for their development and deployment are needed for different stages of the Service Life Cycle (SLC). At the macroscopic level, the life cycle of business services consists of four phases, i.e., Design, Construction, Operation, and Innovation, as illustrated in Figure 2.5. The need for each phase of the SLC is briefly described below.

- ***1st SLC phase- Design:*** This stage deals with strategic planning and rough design specification of business services. In this stage, we need to identify the required services (manual task or software services), their goals and functionalities.

- ***2nd SLC phase- Construction:*** This stage deals with the logistics, construction, and procurement of business services. Therefore, a number of functionalities (operations of a service) need to be implemented in this stage to support the configuration and establishment of the needed business services, for the purpose of service implementation. Moreover, this stage encompasses a fully detailed specification of business services, such as the interface of services (WSDL documents).

- ***3rd SLC phase- Operation:*** This stage is considered as the long operation phase of existing business services. Therefore, the operation phase encompasses a large number of functionalities related to the operation, management, deployment, announcement, and delivery of the services. The utilization of offered services, such as service registry, discovery and execution continue heavily in this stage. Moreover, in this stage of SLC, the quality of services are measured in order to make service level agreements (SLAs).

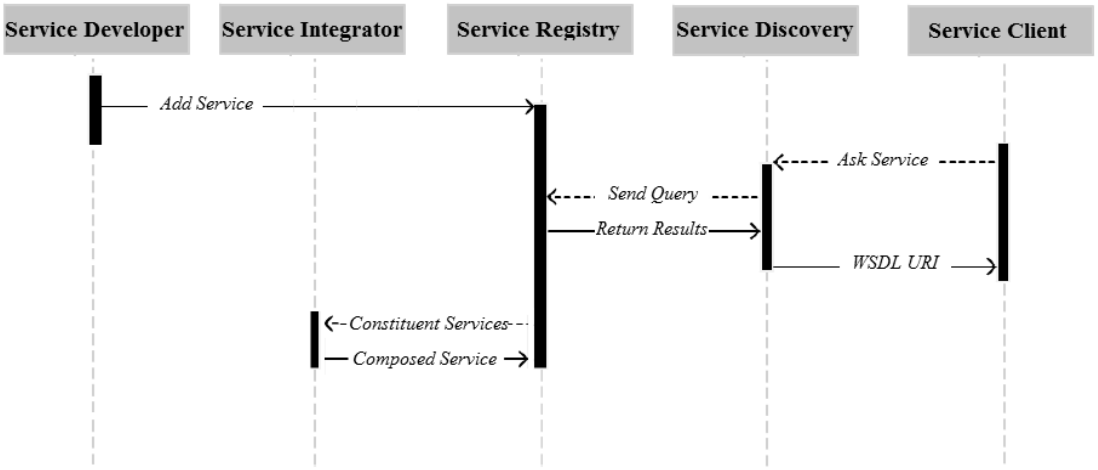


Figure 2.6: The UML sequence diagram for SLC.

- ***4th SLC phase- innovation:*** Finally, the innovation or design of new business services would be necessary for solving some emerging problems, or service enhancement/adaptation. In this stage of SLC, service designers can design new value added services through adaptation of the service interfaces, or compose some existing services. Moreover, it is possible to identify exactly the same services (functionality) and replace them with the optimized alternative service (non-functionality).

Figure 2.6 shows the UML sequence diagram for the main operations involved with business services in SLC. At first, a *Service Developer* must add its provided services to the *Service Registry*. Then, a *Service Client* can ask *Service Discovery* for his/her desired business services, and consequently, receive a WSDL URI as the response. The *Service Discovery* should send a query and receive corresponding results from the *Service Registry* to provide response for the *Service Client*. Moreover, for the purpose of providing composed business services, a *Service Integrator* can retrieve two or more business services from the *Service Registry* as the "Constituent Services", and then integrate them as a composite or composed business service, and finally publish it in the *Service Registry*.

Considering the discussion above, we have identified a set of functional and non-functional requirements to establish a framework for business service interoperation in CNs. Note that SMEs (VO members) in the VBE are fully independent and autonomous; therefore, the lack of uniformity in full and formal definition of implemented software is quite challenging. Moreover, functionality and interactions within each component service (we have called it behavior) are not addressed in the former works, which raise some challenges in semi-automated service discovery and composition.

The non-functional requirements mostly address the specifications of needed

meta-data for business services, as follows:

- Unified formalism of syntax, semantics and quality criteria of services.
- Formalization of service behavior, which tries to model the externally observable behavior of business services.
- Other non-functional requirements for software systems, e.g., security, trustworthiness, and scalability.

Besides the non-functional requirements, there are also some functional requirements for business services as described below, which need to be supported:

- Service specification/registration tool to store and index the specified meta-data for services.
- Effective service discovery to search among registered services, based on their specifications.
- Supporting bundled/composite services to make a bundle of atomic business services as an integrated composed business service.

We address these functional and non-functional requirements in the design of our proposed reference framework for service oriented collaborative networks (see next section), except some non-functional requirements, such as security and authorization that are out of the scope of this thesis.

2.4.2 Proposed Architecture

Figure 2.7 shows the traditional view of service oriented architecture, which consists of three major tasks: Service Provision, Service Registry, and Service Consumption.

In order to customize the traditional architecture of the SOA for our purpose, we should first add another sub-task to this architecture, namely the "Service Design & Implementation" sub-task beside the "Service Provision". Figure 2.8 shows this variation of the SOA Architecture. As such, in fact we have extended the traditional view of SOA by separating design/implementation from the service provisioning task, to also emphasize providing the common VO business service meta-data and elements that can serve as service selection criteria. The meta-data mainly specifies the capability of a business service. The capability consists of syntax, semantics and behavior of services, which are described in Section 3.3. Moreover, the VBE that facilitates a collaborative environment for business services, can then also monitor and adjust the non-functional values, which are claimed for such services by their service providers.

Figure 2.9 shows our second variation to the SOA that focuses on service composition for SOCN. Similar to the atomic services (see Figure 2.8), provision of

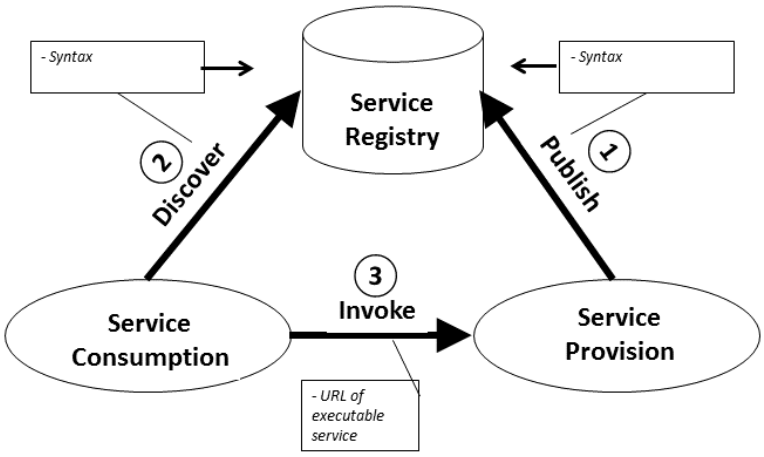


Figure 2.7: Traditional view of Service oriented architecture.

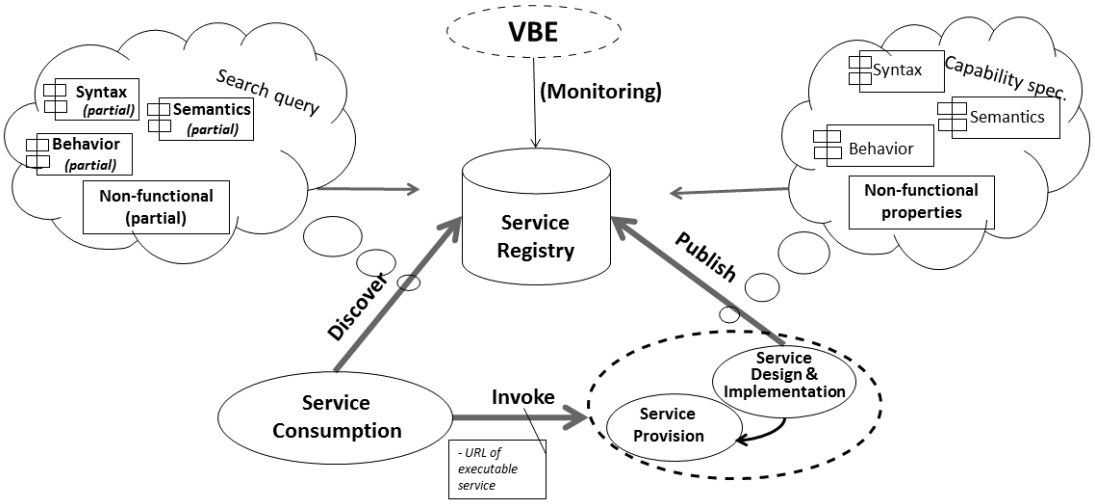


Figure 2.8: The first variation of SOA Needed for service design in SOCN.

a composite service is also separated from the composite service design and implementation. To design composite services, first the "Service Integrator" should discover (from the registry) the constituent services that he/she plans to involve in the composite service. The details of the discovery task will be explained later in Chapter 4. After that, one proxy should be automatically generated for each selected constituent service, as it is needed to support the end users in execution of the services. In fact, the generated proxies are required to support the automated invocation and the data exchange among services. Moreover, the integrated service designer/implementer needs to also interconnect (define the coordination of) the selected services, and only then he/she can define a full specification of the

integrated services to be provided as a new (composite) business service. Chapter 5 addresses the needed steps that should be done by an Integrated service designer / implementer to design a new composite service. Finally, the provided composite services should be published in the "Service Registry". As such, these services can also be discovered through the SOA triangle, like the atomic services.

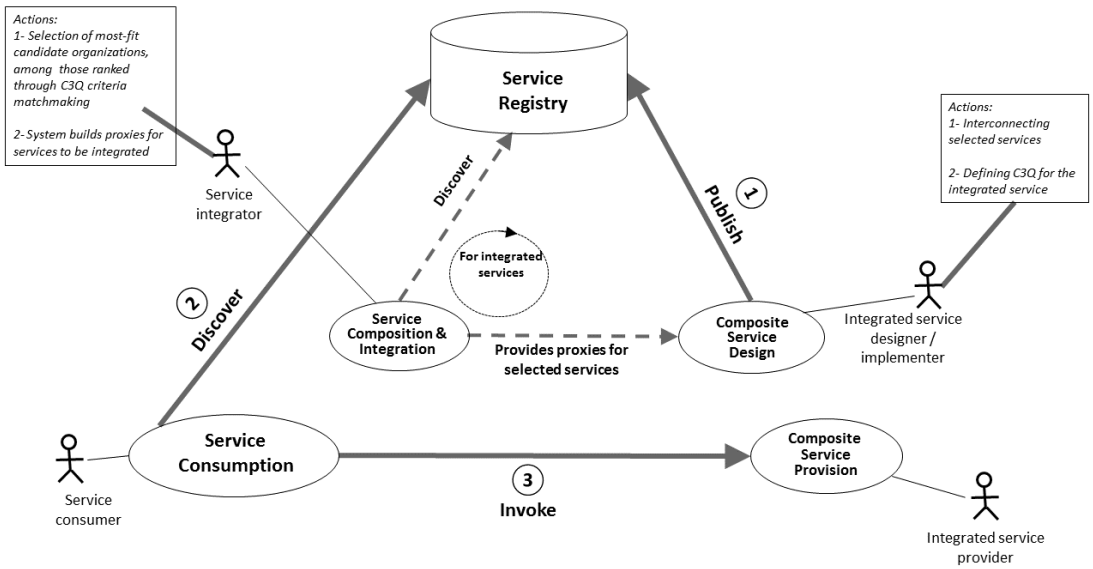


Figure 2.9: The second variation of the SOA addressing service composition in SOCN.

2.4.3 Implementation Architecture

Each Business Service (BS) shared within the VO is represented by a set of Business Processes (BPs), while each BP involves the invocation of one granular software service. Therefore, designed BSs may be specified either as atomic services or composite services, whereas each composite service is in turn composed of several other atomic or composite services as its constituent. To support business service composition in VOs, such software services need to be both discoverable and integrable, as it is considered in the designed reference framework.

In a VO, there are usually two kinds of business services provided by its organizations: manual tasks and software services. In our research and proposed framework, we only deal with software services, which correspond to their defined BPs. If desired by the environment, we can of course also define only a simple software service for each of the manual tasks to include two basic operations for them: Start and Stop, but nothing more than that.

Figure 2.10 illustrates an implementation architecture capturing the needed elements for establishing service oriented computing in VOs. This architecture

is designed in order to identify the significant sub-components and relationships among them, for the development of the extended SOA model (as represented in Figure 2.9). Note that the introduced implementation architecture is not directly tied to any standards, tools, or other concrete implementation technologies. This architecture seeks to provide common semantics that can then be used unambiguously across and between different implementation options, however in this PhD work, we have developed a proof of concept (POC) for our approach which presents particular architectures, standards, technologies and other implementation details to realize this implementation architecture. This architecture for our

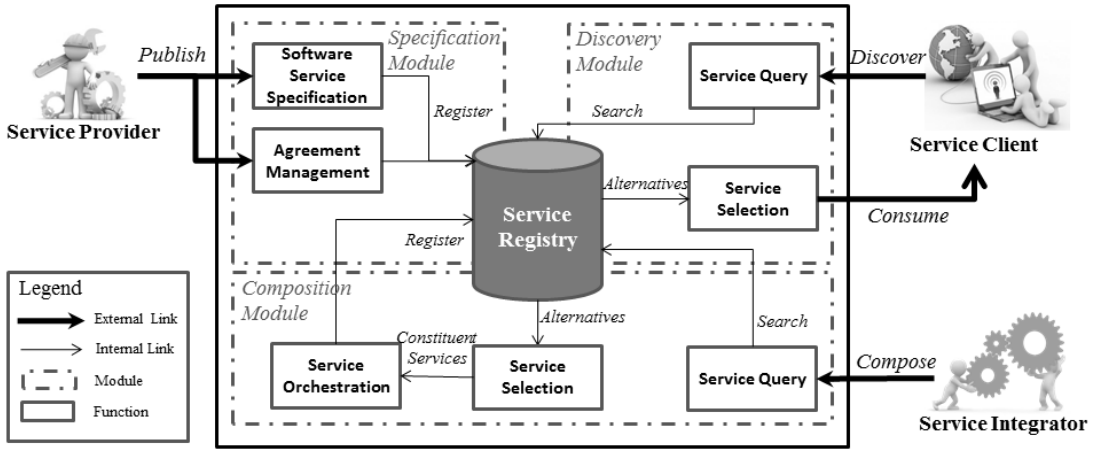


Figure 2.10: Implementation architecture for service oriented collaborative networks.

developed POC is conceptually composed of three software modules, including: Specification Module, Discovery Module, and Composition Module, as further briefly explained below.

- Specification Module addresses the software services that are offered by different members/stakeholders of the VO in the role of service providers. The shared services in the VO are published in a service registry or directory, such as the UDDI [42], complying with specific Operational Level Agreement (OLA) [70] defined at the VBE level. This OLA is then agreed among the VO stakeholders, to describe the responsibilities of each VO member/stakeholder toward provision of both their own atomic services, as well as when they are involved in the specified composite services. OLAs are also supported by Service Level Agreements (SLAs) defined among web services [105]. At VOs, SLA reflects an agreement between the provider and the clients of a service to create assurance on the service level at the binding time. As such, the expected performance behavior of a deployed software

service is defined by the service level. For example, response time, supported throughput, and service availability are performance metrics, which can be considered in SLA. In fact, SLAs provide some of the needed information for developing a new service competency model in order to assist the user in service selection. The approach adopted for service quality assessment in our framework borrows ideas from [150], which monitors the behavior of VO partners in order to identify their level of trustworthiness. According to this approach, all agreements in OLA and SLA are considered as promises exchanged among the involved partners in the VO. Then, by applying VO Supervisory Assisting Tool (VOSAT) [150], at any point in time, the trust level of a VO partner would be reflected on its claims about different characteristics of its own provided services, as well as on its assesment and feedback related to other providers' services.

The function labeled as "Agreement Management" manages all tasks related to the service agreements, and keeps the results in the Service Registry.

Another function of this module is named "Software Service Specification", which presents the content for triple (syntax, semantics, behavior) meta-data as the means to accurately formalize every atomic software service. Every composite business service (in turn constituting a set of BPs), is then represented by a set of contents for this meta-data, where each triple provides the concrete formalization of one atomic service. Our introduced meta-data captures the three characteristics of each service, namely its syntax, semantics, and behavior in order to properly support machine-to-machine service interoperation in VOs. The introduced meta-data in turn facilitates service-oriented computing (i.e., service discovery and service composition), which use these specifications, required by them to perform their tasks, in an automated way. These are described in more detail in the next chapters. All of the specifications will be registered in the Service Registry

- Discovery Module of the framework provides mechanisms for service discovery and selection of most-fit service among the existing shared services in the VO. Automated and successful application of search services of this module requires the functional meta-data (i.e., syntax, semantics, and behavior of services) provided through the Specification Module. The *Search* function returns a list of alternative services from the *Service Registry* that can be matched with the service query (based on the functional properties of services). After that step, the *Service Selection* function chooses one of the provided service alternatives that is optimal for all non-functional properties specified and requested for the service.

- Composition Module of the implementation architecture involves the functions introduced to support service composition. This supports service de-

signers by offering new value-added services, through composition of existing shared services in the VO. Eficient service composition through this module requires not only considering the rich meta-data captured in the Service Registry, but also coordination of the required interaction protocols among the constituent services that form a composed service. The information about the constituent services, as well as what is needed for their orchestration model, should be recorded in the Service Registry.

The requirements and implementation details of this framework are further discussed in the next chapters.

2.5 Conclusion

Emerging developments under the umbrella of Future Internet and particularly on web services, highlight SOA-based paradigms and approaches to support service oriented collaborative networks. Software services, e.g., the web services, and the SOA paradigm provide rapid, cost-effective and standard-based means to improve service interoperability and collaboration in CNs. Although the research area of collaborative networks is active, the higher-level abstraction of the activities that simplify collaboration among organization members in software service oriented CNs and specially using SOA is still lacking. The goal of this chapter is to address a comprehensive framework in order to support a semi-automated service discovery and compositions in VOs. With this in mind, we propose a reference framework and an implementation architecture that allows us to efficiently support service-oriented collaborative networks. The implementation of the proposed framework consists of three main software modules that realize the functional and non-functional requirements of SOCN, as addressed in the next chapters.

Chapter 3

Specification of Business Services

Parts of the material in this chapter is published in the following papers:

- Sargolzaei, M. and Afsarmanesh, H., 2017. C3Q: A Specification Model for web services within Virtual Organizations. In Collaboration in a Data-Rich World. Springer Berlin Heidelberg.

- Afsarmanesh, H., Sargolzaei, M., and Shadi, M., 2015. Semi-automated software service integration in virtual organisations. Enterprise Information Systems, 9(5-6), pp.528-555.

3.1 Introduction

Fast pace in development in the area of services prompts exploration of the role of Service Oriented Architecture (SOA) in assisting organizations to deal with service interoperability and flexibility demands. Using SOA and its available standards enable organizations to better connect their activities and operations. To properly support VOs, as a first step, organization services should be concisely specified, such that they are recognizable, discoverable, comparable and integrable. Currently, web services are the most promising technology that implements the concept of SOA, and provides the basis for the development and execution of business processes that are distributed over the Internet. A web service is defined as a self-contained, modular, loosely coupled, reusable software application that can be described, published, discovered, and invoked over the World Wide Web [56]. In the last decades, the Web Service Definition Language (WSDL) [41] has emerged as the most prominent standard for the specification of web services. This standard, however, does not provide the proper concise specification basis for a service client to get a full understanding of "What the service exactly does" and "How the service exactly performs". This lack of information about services usually results in the mismatch between the providers

objective and consumers demands, when especially considering the functionality of corresponding service. In spite of several proposed additional standards, a comprehensive view on which aspects of a service is required to be concisely specified is still lacking [158].

As mentioned in Chapter 2, in order to effectively share the business services (BSs) and to facilitate their reusability and integration in VOs, it is necessary to define and register the BSs in a common VO directory. Despite the simple appearance of the above requirements, several complexities and challenges arise that need to be precisely addressed, as mentioned below:

- There is no uniformity in service definitions, since VO partner organizations are and remain independent and autonomous.
- There is a lack of common ontology for services that need to be shared in the VO.
- There is no comprehensive and concise functional specification for services, as required for their potential integration into value added services and to deploy shared software services.
- There is lack of unambiguous machine interpretable (e.g., an XML-based standard) representation of services including all aspects of BSs, as needed to generate the required codes for their executions.

This chapter aims to define a model to address and resolve these obstacles and challenges, and to provide a basis for discovery and composition of services in VOs, as further discussed in Chapters 4 and 5. Our proposed service specification model can be effectively applied for specifying software services (i.e. business services of organizations materialized by software systems) and it can also minimally support specifying the start and termination of manual tasks (i.e. business services executed by human).

This chapter is organized as follows. Section 2 briefly outlines some theoretical and technical aspects of the related works in order to serve as the base for our service specification. Section 3 is devoted to a brief description of the business service concept. In Section 4, we sketch out our proposed model for VO business services, called C3Q. Section 5 represents our extension of WSDL documents to address all aspects of C3Q. Moreover, a GUI is implemented to assist users with behavioral description of their services, which is demonstrated in Section 6. In Section 7, our designed model for registration of services is represented, and finally we conclude the chapter in Section 3.7.

3.2 Related work

Web services have nowadays turned into a prominent research point in the field of Service-Oriented Architecture [74], and has been widely accepted by the service

industry. One key point for the success of web services technology is the employment of XML-based standards, such as SOAP and WSDL both for communicating and self-describing [48]. The Web Services Description Language (WSDL) is the most adopted standard for web service description, while it is limited to self-describing of the structure of the messages and operations, but not for supporting the concept or capability of the service. This limitation of WSDL which is known as the "lack of semantics" [53] in describing the service capability, has consequently required human intervention in order to interpret and assume the semantics of the message content, and the capability of the web service, as it is vital to ensure valid and befitting use of the service. Apart from lack of semantics, another limitation of WSDL is not addressing the configuration of stateful web services, or the so-called behavior of services. A single web service that consists of several operations can be considered as a stateful web service, where the state of a specific instance of the service can be kept between its invocations. The behavior specification of a service in fact, indicates the valid sequence of its operations' invocations, which is absent in WSDL. Specification of behavior plays a vital role in service composition and also improves service discovery as discussed in the subsequent chapters. Thus, the lack of semantics and behavioral specification are a major drawbacks of WSDL, and consequently become barriers to achieving automatic or semi-automatic service discovery, composition, and execution.

Numerous standards and languages have been proposed to describe semantics of web services, such as OWL-S [136], WSMF [58], and WSMO [100]. Nevertheless, the typical tendency of researchers is to build a semantic layer either on the top of WSDL or to be integrated into WSDL, to semantically describe the functionalists of web services. Some research efforts are spent on extending WSDL with semantics annotations, such as WSDL-S [6] and SAWSDL [95]. WSDL-S can annotate the information provided in WSDL using different semantic languages, such as RDF and OWL. The elements of WSDL-S that are added to the WSDL standard documents are the modelReference, category, precondition and effect. SAWSDL is also defined as an extension of WSDL to describe the semantics of its elements through providing the mechanisms to bind ontology concepts to the semantic annotations of WSDL.

Although several researchers have tackled "the lack of semantics" problem and also some developed tools (e.g., [133]) have achieved good results and succeeded with specification of syntactic and semantic properties of web services, they hardly address the behavioral signature of a service, describing a service's sequence of operations, which the user actually needs to properly use and interface with. This is partly due to the limitations of today's standard specifications, which do not cover such aspects. Representation of the behavior of stateful services is very important during the discovery and composition of services, to provide users with an additional means to refine their search and to automate its composition in diverse environments. So far a few formalisms have been proposed that are able to model the behavior of a service. For example, session types as a formalism

for structuring interactions and reasoning over communicating processes can be applied as a model to describe the behavior of services. A session is defined as a logical unit of information exchanged among participants, which specifies the topic of conversation as well as the sequence of the communicated messages [46]. Session types, which can be assigned to end-point processes, describe the user view of an interaction. In [47], the authors have specified component behavior as session types showing that session types can also describe the behavioral signature of services. Several works such as [73] and [28] have also proposed to encode the behavior of services in a BPEL (Business Process Execution Language) specification. They present some shortcoming in terms of performances, supporting complex behavior, and representing an unambiguous machine interpretable (e.g., an XML-based standard) description of services' behaviors. In our work, we use constraint automata (CA) [18] as our base formalism for specification of behavior, because the CA models are supported by our related tool sets for service orchestration, can readily be extended to support soft constraints, and we believe automata models are more understandable and readable for engineers. As a consequence, we propose to improve the WSDL documents through supporting a constraint automata specification as an extension. This light extension of WSDL allows us to have an unambiguous machine interpretable description of web services' behaviors.

Besides the functional description of services, it is essential to also capture non-functional properties or quality of service (QoS), of business services in order to meet the performance requirements of clients, such as service availability, and even to respond to providers' requests, e.g., about cost. In other words, describing and registering the non-functional service properties are necessary in order to satisfy both the service clients and the suppliers. One such specification can be found in [114], where the specification of the i^{th} web service is represented as a tuple (Fi, Qi, Ci). Fi is a textual description of the service functionality, Qi specifies the QoS values, and Ci represents the costs associated with the web service i. Several research works in this area are instead in favor of extending the WSDL capabilities rather than introducing additional languages on top of it. The works done in [128] and [43] are two instances that capture QoS specifications with extensions of WSDL files. A lightweight WSDL extension called Q-WSDL (QoS-enabled WSDL) is introduced in [43], which specifies the QoS characteristics of web services.

3.3 C3Q Model of VO Business Services

For the sake of developing an architecture to support service oriented VOs, we first define a holistic service model, on top of which this architecture can be founded. Here, first a new meta-data based on this mode is introduced to formalize the description of business processes. This model targets the comprehensive capture

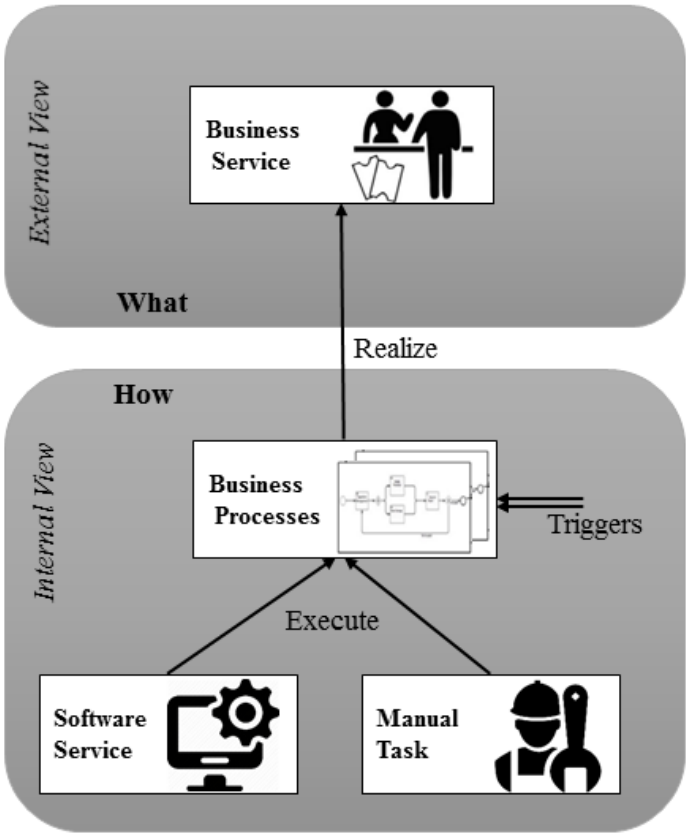


Figure 3.1: Views of business services.

of all characteristics of BSs, through an unambiguous formal description. From the service analysis point of view, all services intended to be shared within the VO are required to be unambiguously defined, according to C3Q, by their providing enterprises.

In VOs, there are usually two kinds of business services, including the manual services and the software services. Figure 3.1 shows that each business service can be realized by one or more business processes (BPs). In fact, sometimes the BS might be carried out through alternative kinds of business processes, based on different triggering events. As such, a business service might be seen as an abstract construct that encapsulates the external or client's view. This "construct" describes "what" added values would be delivered by the service to the client, as well as their delivery conditions. Furthermore, the internal view of the business services illustrates "how" the BS is performed through the business processes/sub-processes, and with which corresponding triggering events (see Figure 3.1). The actions involved in the business service execution might be materialized either automatically, through some software functions (the so-called software services), manually through some human-based tasks (the so-called manual tasks), or through a combination of these two. In a nutshell, each BS may in-

volve a number of BPs, while each BP may involve either the invocation of more granular software services or performance of some manual tasks. In our proposed framework, we only deal with software services, which correspond to their defined one or more BPs. For manual task, we can only define a simple software service that includes two of their basic operations: Start and Stop. Through this specification only, manual tasks can also be specified and treated in the framework as software services, otherwise these are outside the scope of this thesis. Considering that a defined business service may be composed of several software services as its components, the business service designer shall also present a complete flow of interaction protocols, e.g., through a BPMN diagram.

We propose a concise representation model for business services, our so-called C3Q model, namely addressing their Capability, Costs, Conspicuities, and Quality criteria. As such, VO business services can be uniformly defined by their providers and published in the common pool of VO services in order to support their effective sharing and reuse. Figure 3.2 shows our service profile that we define as an extended instance of the "business service competency" consisting of *capability*, *cost*, *conspicuity*, and *quality criteria* as defined in [3]. For our concern, Capability is the most important part of the service profile, which represents the functional properties of business services, and we have extended it with *syntax*, *semantics*, and *behavior*. An example of the syntactic aspects of a BS description include its input and output, while the semantic aspects of the BS description include its textual description, and the purpose-classification, under which the BS falls. The behavioral aspects of a BS description address the concise definition of its functionality, based on which it can be unambiguously implemented by a software developer. Services' *syntax*, *semantics*, and *behavior* are each defined in detail in the next subsections. The other elements of C3Q model, including the costs, conspicuities and quality criteria of services, represent the other important required descriptions related to non-functional properties of BSs. These three are also further defined in the next subsections. Since two different BSs may offer similar functionality but have distinct non-functional characteristics, it is necessary to consider both the functional and non-functional properties as the BS competency, in order to fully satisfy the demands of a service client, especially during the service selection phase [104], [113]. There is no widely accepted standard for definition of different component of the C3Q service profiles except for representing the syntactical properties of the web services, which is through WSDL. Since all web services, as the most popular business service implementation, already contain their WSDL documents, we extend WSDL to support the C3Q service profile. In the next subsections, we study different elements of C3Q model in detail. Moreover, we introduce several new tags that must be added to standard WSDL, in order to form our proposed extension of WSDL, namely XWSDL. We can apply a number of different notational options for representing each C3Q aspect in XWSDL. We have adopted a specific notation for formalizing each of these aspects, as later addressed in this section. Furthermore, in Section

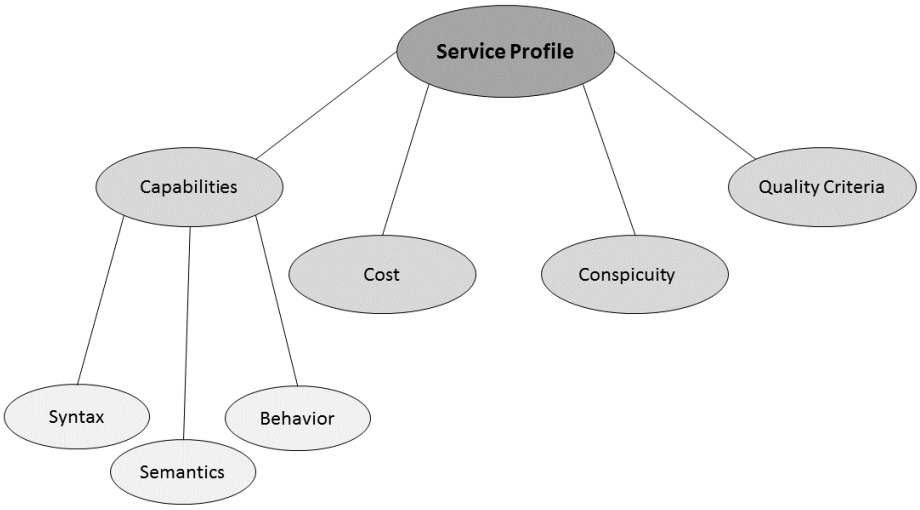


Figure 3.2: The C3Q services profile.

3.4, we will discuss more about the XWSDL documents and introduce the data-model for our proposed extensions to achieve a complete definition of XWSDL.

3.3.1 Syntax

Typically, the syntactic properties of a service are represented by XML-based standards and languages, such as through Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP) [42]. A WSDL description is an XML document that contains the following information about a specific web service:

- **What** the service does, which textually describes the service's operations, as well as the input and output parameters that define the operation's messages.
- **How** the service is accessed, which describes the data structures, binding implementation and protocols needed for sending messages through the web to reach the service location.
- **Where** the service is located, i.e. the hosting address that executes the service implementation.

3.3.2 Semantics

The conceptual properties of software services, here referred to as semantics, are typically defined with an ontology, i.e. an explicit specification of a conceptualization of knowledge related to services. The service ontology definition, also in the VO context, encompasses a group of vocabularies that specify the semantic attributes of the services (e.g., goals and category) and their inter-relationships, which together present a meaningful concept about the service [33]. In fact, the semantic description of business services would enrich the information about services to the level that cannot be specified by their mere syntactic description. Purpose-classification of the BS (e.g., goals and context) are good examples of the semantic aspects of the BS specification, which aim to categorize services in order to improve service discovery and matchmaking.

The proposed service semantics within our C3Q-based service description consists of a set of conceptual information that is more understandable for both the human and the machine using third-party ontologies. For example, "goal" as a semantic attribute can describe the business logic of the service (e.g., the goal of Monitoring). An ontology for this semantic attribute describes the "things" that exist in the domain (i.e., goal) including the concepts, relations, etc., in a consensual and formal way [107]. The ontology thus provides the stable baseline for shared understanding of the domain that can be communicated between service providers, clients, and inter-organizational service integrators. It is also possible to consider an existing element of the WSDL document, e.g. "operations", as a semantic attribute, by annotating that element and linking it to a domain ontology.

To support semantic discovery, we must link each attribute to a particular reference domain ontology. Such ontologies encompass a set of well-founded structured data and their semantic inter-relationships, which can then be used to improve the matchmaking and discovery of services. In this research, we do not deal with challenges of ontology construction. Rather, we assume the existence of a pre-defined domain-specific ontology or simply a taxonomy for the specific domain related to each semantic attribute. We capture the three elements described below to specify each service semantic attribute.

- **name** that represents the title of the attribute, e.g., goal or context.
- **taxonomyUri** that refers to the link to a related domain ontology or taxonomy for the attribute.
- **value** that represents the value of that attribute for the service.

An example of service semantic specification in our model, including the two semantic attributes of context and goal, are represented in Figure 3.3.

```
<xwsdl:semantics>
    <attribute>
        <name> context </name>
        <taxonomyUri>
        http://example.org/onto\#context
        </taxonomyUri>
        <value> Purchase </value>
    </attribute>
    <attribute>
        <name> goal </name>
        <taxonomyUri>
        http://example.org/onto\#goal
        </taxonomyUri>
        <value> Monitoring </value>
    </attribute>
</xwsdl:semantics>
```

Figure 3.3: Example of the WSDL extension for semantic description.

3.3.3 Behavior

Beyond the semantic description for the operations that a service can provide, and the syntax description of how they are to be invoked, a specification of the proper order in which those operations can be invoked is also a prerequisite for correct implementation and use of a service. Behavioral specification of a service refers to the specification of all admissible invocation orders of the operations of that service. The discovery of suitable services that can match a query must also consider the behavioral specification of candidate matches. Furthermore, what operations can be performed at a given point in time by a client of a service may depend on the history of the previous operations that have already been performed (usually, by the same client) on that service. Therefore, a specification of the behavior of a service is in general, "stateful". However, these states are not typically maintained within the service itself. Specifically, the so-called "stateless" services do not maintain a state between requests, which means the result of each service invocation request is completely independent from the previous requests. RESTful [94] is an implementation of such services. We have proposed the term *exostate* to denote the states of a running service or system that are maintained outside of the implementation of that service, and similarly we have used the term *endostate* to capture the service's internal configuration states. Trading endostates for exostates has important architectural advantages. For instance, because a REST service has only a single endostate, it never needs to reset itself to recover from a potential previous communication failure that disrupts a client's session. However, most of such services are not truly stateless, in the sense that to use them properly, a client must still follow a permissible sequence of invocation for their operations, encoded in their exostates. The exostates of a REST service

are represented by the values of a set of context parameters that are passed back and forth between the service and each of its clients.

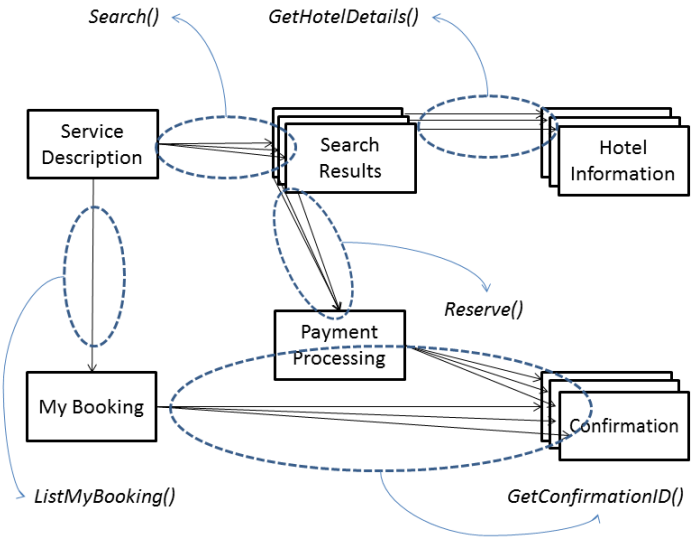


Figure 3.4: Operations of a restful example: the Hotel booking service [94].

Consider a hotel booking example, such as the restful service of defined in [94], illustrated in Figure 3.4. In our terminology, as a restful service, the implementation of this service has a single endostate. However, this service cannot be used properly unless for instance *getHotelDetails* operation is invoked only after a *search* operation. Any proper use of this service requires remembering whether or not a search has indeed been performed yet, and perhaps also remembering the results of such a search, etc. The REST architecture requires such information to be kept outside of the service implementation itself, on the client/user side (perhaps kept as cookies), and passed back and forth between the client and the service, during the service invocation. From the perspective of a client however, the stateless service of [94] depicted in Figure 3.4 cannot be used without considering an specification of its stateful behavior. In our approach, we represent a formal and concise specification of the behavior of services in terms of Constraint Automata (CA) [18]. A CA is essentially a labeled transition system (LTS) that resembles classical finite state automaton in the sense that they consist of finite sets of states and transitions. When a CA is applied for modeling the behavioral specification of a service, its states represent the configuration of the service, and every transition describes how the configuration of this service changes from one state to another by execution of one of the service's operations. Figure 3.5 shows the CA which specifies the behavior of the example service represented in Figure 3.4.

The above example illustrates that many more systems and services than commonly acknowledged are truly stateful, in that the specification of their behavior

involves more than a single state, regardless of whether such states are implemented as endostates or exostates. Searching for an appropriate service, as well as its manual or (semi-)automated adaptation, composition, or invocation must take its desired behavior into account. In our research, we use the term stateful comprehensively to refer to any service specification whose behavior requires more than a single state.

To the best of our knowledge, previous works on matching and retrieval of services do not consider the impact of exostates on the suitability of the behavior of services in their search. This fact makes our behavior-based discovery tool represented in Chapter 4 novel, while not directly comparable with search and retrieval tools, such as the work presented in [133]. The service behavior specification also assists us in developing a tool to support automatic code generation for the orchestration of component web services, as represented later in Chapter 5.

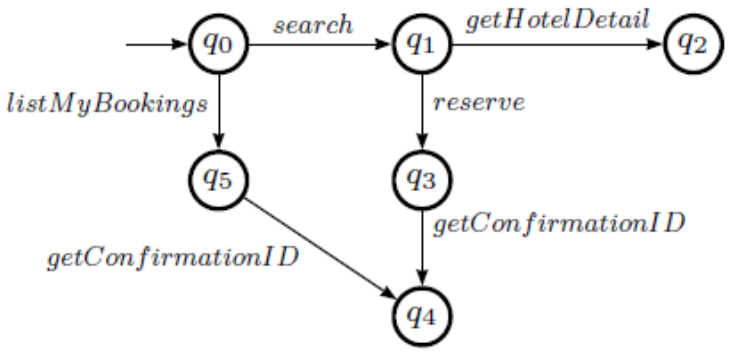


Figure 3.5: The behavior specification of the Hotel booking service in terms of Constraint Automata.

Although WSDL provides required information to establish a connection with a web service, this specification lacks the behavior description of the services [4]. We propose XWSDL to also incorporate behavioral information to extend a WSDL document. Note that our approach retains the original structure of the WSDL documents and merely enhances them by adding a few new tags in the XML-based files. Figure 3.6 shows an example of the WSDL extension with the behavior description of the example represented in Figure 3.5.

3.3.4 Quality Criteria of Services

The model for quality of service (QoS) consists of a number of properties, each related to an aspect of QoS.

Large number of reported research have focused on supporting QoS for business services. Although so many QoS solutions are proposed, service developers

```
<xwsdl:behaviour>
    <transition>
        <name> Search </name>
        <currentState> q0 </currentState>
        <nextState>  q1 </nextState>
    </transition>
    <transition>
        <name> getHotelDetails </name>
        <currentState> q1 </currentState>
        <nextState>  q2 </nextState>
    </transition>
    <transition>
        <name> reserve </name>
        <currentState> q1 </currentState>
        <nextState>  q3 </nextState>
    </transition>
    <transition>
        <name> getConfirmationId </name>
        <currentState> q0 </currentState>
        <nextState>  q5 </nextState>
    </transition>
    <transition>
        <name> getConfirmationId </name>
        <currentState> q3 </currentState>
        <nextState>  q4 </nextState>
    </transition>
</xwsdl:behaviour>
```

Figure 3.6: Example in XWSDL (WSDL extension) for the behavior description.

and clients still are not able to handle QoS-related concerns easily. The reason is that a universal QoS specification standard is still absent.

Gu et all [69] define a QoS specification language for web applications, which mainly includes three levels: user-level QoS specifications, application-level-QoS specifications, and resource-user-level QoS specifications. The user-level QoS specification consists of the overall descriptions about the application (e.g., name and provider) and associated qualitative user-level QoS criteria (e.g., low, average, high, excellent).

Web service selection process is an important aspect of service-oriented computing, also when considering the increasing growth of Internet. However, by considering the Quality of Service (QoS), an optimal service selection can be guaranteed. As such, service providers have to enhance not only the functionalities of web services, but also their QoS, such as reliability, response time, availability, etc.

QoS documents quality criteria such as performance, reliability, availability, response time, etc. Web service quality however cannot be measured by only one quality criterion, or even by several fixed and pre-defined criteria. Indeed, the quality criteria of each business service can be varied based on the purpose of the service evaluation or the domain area of the user service. As an example, resolution is a very relevant quality criterion for image-based services. Therefore, in our framework we need to provide an approach for QoS specification that supports defining quality criteria dynamically.

We consider that a QoS specification has the following attributes, in order to assure an expressive formal description of the quality of services.

- **Criterion** which represents a quantifiable aspect of a service like availability.
- **Unit** that is used as a standard for counting or measuring the corresponding quality criterion, e.g., hours per day, hours per week, etc. for availability.
- **Range** (Min and Max) which depicts the highest and lowest possible values for the quality criteria. The range is needed when we want to compare the same quality criteria with different units.
- **Value** that represents the amount of the corresponding quality criterion.

An example of QoS specification in XWSDL is represented in Figure 3.7.

```
<xwsdl:QualityCriteria>
    <Criterion>
        <Name> availability </Name>
        <Unit> hours </Unit>
        <Value min="0", max="24"> 24 </Value>
    </Criterion>
    <Criterion>
        <Name> reliability </Name>
        <Unit> Percentage </Unit>
        <Value min="0", max="100"> 90 </Value>
    </Criterion>
</xwsdl:QualityCriteria>
```

Figure 3.7: Example in XWSDL (WSDL extension) for quality criteria for services.

The $< QualityCriteria >$ tag is a container tag, which contains at least one $< Criterion >$ tag. The $< Criterion >$ tag is also a container tag indicating the required attributes of the corresponding criterion including “Name” “Unit” and

"Value". The $< Value >$ tag also provides the minimum and maximum values of the criterion, i.e., the "Range" attribute.

3.3.5 Cost

Cost is a key economic attribute that affects selection and usage of business services. Thus, we introduce cost as an additional QoS parameter for specification of business services. A cost specification consists of three attributes:

- **Initial price** which represents the value of the cost, e.g., 5.
- **Unit** that defines the unit of the cost, e.g., Dollar or Euro.
- **Price plan** which is used to model the method of cost estimating, e.g., per invocation, per transmitted byte charges, etc.

in Figure 3.8, an example of cost description is provided in our proposed XWSDL specification.

```
<xwsdl:Cost>
    <InitialPrice> 10   </InitialPrice>
    <Unit> Dollar </Unit>
    <PricePlan> Per invocation  </PricePlan>
</xwsdl:Cost>
```

Figure 3.8: Example in XWSDL (WSDL extension) for the cost.

The $< Cost >$ tag is a container tag involving three other sub-tags in order to specify the cost of the services. The $< InitialPrice >$ tag indicates the initial charging price when the user invokes the business service. The unit of the price is revealed within the $< Unit >$ tag. Finally, the $< Priceplan >$ tag represents the method that the service provider would like to adopt for cost estimation.

3.3.6 Conspicuity

Conspicuity for a business service is the quality or state of indicating proof of good performance, from the prospective of the service client. It represents the means to identify the validity of information related to a provided service, as claimed by its provider. A business service's conspicuity is measured both through studying the behavior of the corresponding service provider and by capturing the past service consumers' feedbacks.

In our approach for VO support, we have adopted the VO Supervisory Assessment Tool (VOSAT) [149] to assess service conspicuity. The approach adopted for VOSAT monitors the behavior of VO members for identifying their level of

trustworthiness [151]. Applying this approach, all agreements in Operational Level Agreement (OLA) and Service Level Agreement (SLA) are considered as "promises" exchanged among the involved partners in the VO. In the proposed framework, the trustworthiness of each VO partner is reflected, as calculated by VOSAT during the VO operation phase, considering the claims made by each partner, as explained in [151] and [121]. As indicated in [148] different introduced states for promises include: conditional, unconditional, kept, not kept, withdrawn, released, and invalidated, which address different stages within the entire life-cycle of every promise. The life-cycle of every promise is then formalized and monitored, and the trust level of VO partners are assessed through a set of pre-defined causal-relationships among different promise states, and the trustworthiness of the VO members. Therefore, at any point in time, the trust level of a VO member would be reflected on its claims about different characteristics of its provided services, as well as on its feedback about other VO member services. The trust level of each partner is calculated in reference to its own performance in the VO by VOSAT during the VO operation phase.

In our framework, the calculated trust level of each VO member is used as the conspicuity specification for its provided services in the C3Q service competency model defined in this research.

An example of conspicuity description focused on trust level in our proposed specification is represented in Figure 3.9.

```
<xwsdl: Conspicuity>
    <name> trustLevel </name>
    <value min="0", max="1"> 0.6 </ value >
</xwsdl: Conspicuity>
```

Figure 3.9: Example in XWSDL (WSDL extension) for the conspicuity.

The technical details of measuring the organizations' trustworthiness is out of the scope of our research. The authors of [149] show how such trust measurements can be calculated to serve as an input for conspicuity specification of our business services.

3.4 XWSDL

As mentioned before, there is no widely accepted standard for any of the parts of the C3Q services profile, except for representing the syntax for specification of the properties that we define in the web services through WSDL. Furthermore, since WSDL is widely adopted by the web services community, and most web services already have WSDL documents, we have extended the WSDL to support the other properties of the C3Q service profile. Figure 3.10 shows the new tags

added to the standard WSDL that form our proposed extension of the WSDL, namely the XWSDL.

The XWSDL extension follows the defined rules for extending WSDL [41], to

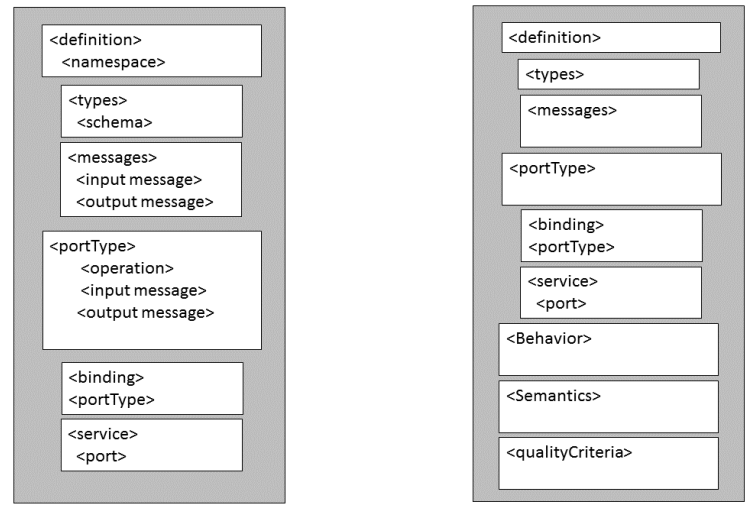


Figure 3.10: Standard WSDL vs XWSDL.

guarantee that any service consumer unaware of the extensions can still parse, validate and use the extended version of WSDL files, i.e., an XWSDL documents. Only, a new namespace "XWSDL" should be used to identify the tags part of the extension. Our approach retains the original structure of WSDL documents and enhances it with new tags in the XML-based files. In order to extend WSDL, we need to define a schema of the elements of XWSDL as its name space. Figure 3.11 shows our schema designed for XWSDL to define all C3Q service profile's properties in the form of an xml-based elements schema.

We used XML-Liquid 2.0 to translate the schema to XML documents, which consist of XSD tags in order to define the elements of the schema. This XML document is used later as the *namespace* for defining XWSDL documents. Figure 8.1 in Annex I shows a part of the XSD tags of this schema.

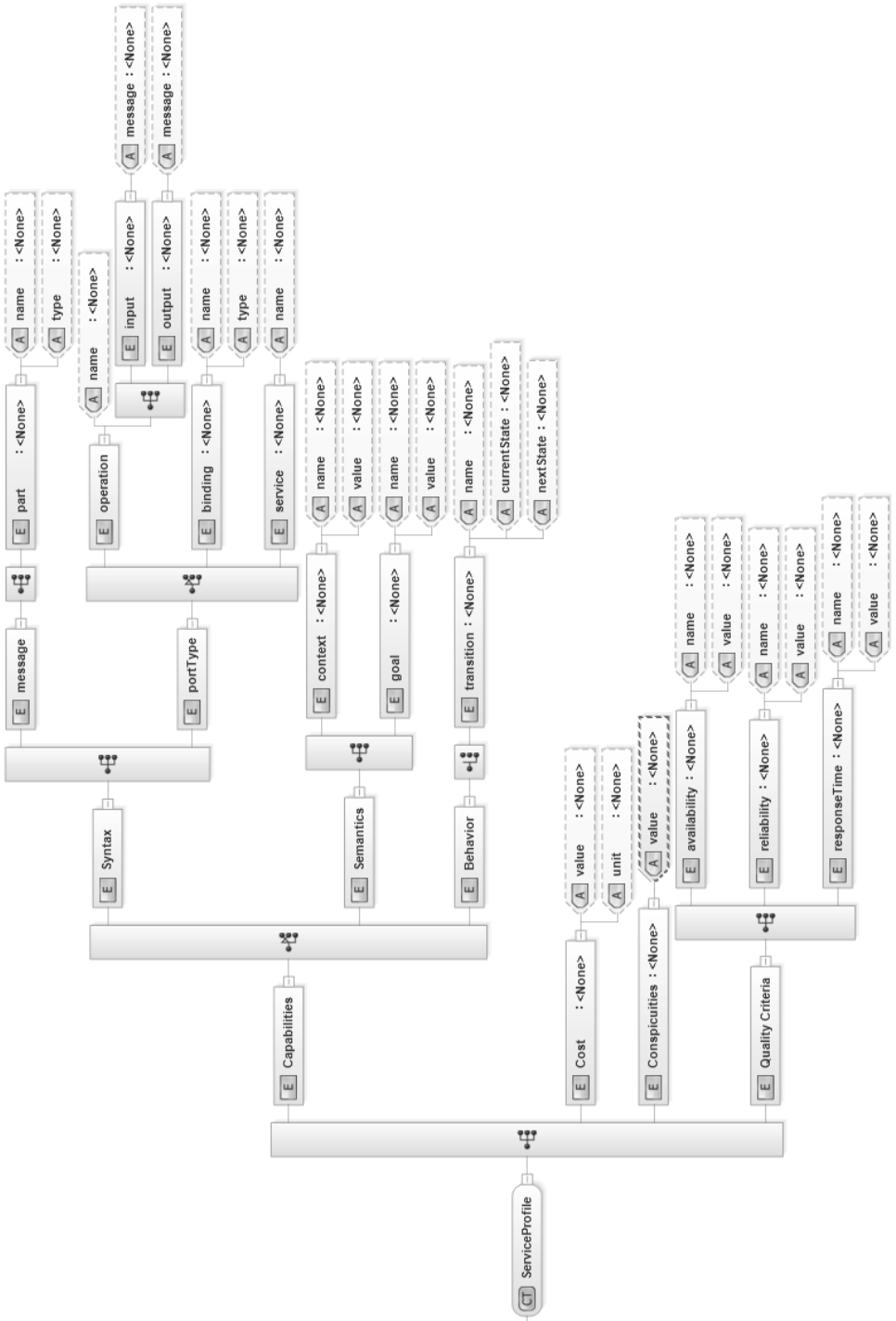


Figure 3.11: The schema of XWSDL.

Our XWSDL definition also adopts "Model-Driven Architecture" (MDA) [112] as the guideline. The Object Management Group (OMG) has proposed MDA as a vision for software development that relies on linking object models together to build complete systems [112]. Model Driven Architecture focuses on providing meta-model, which is simply a model of a modeling language. This kind of meta-models is defined by the use of the Meta Object Facility (MOF[1]), which is the OMG's standard to specify meta-models aimed at describing another model. Nowadays, employing MDA in web service standards has received significant attention, also to assist the automated generation and extension of web service models [62], [43]. Therefore, we apply MDA to our definition of XWSDL through designing a meta-model for it in order to appropriately enrich our web service descriptions based on the C3Q.

3.4.1 XWSDL Meta-model

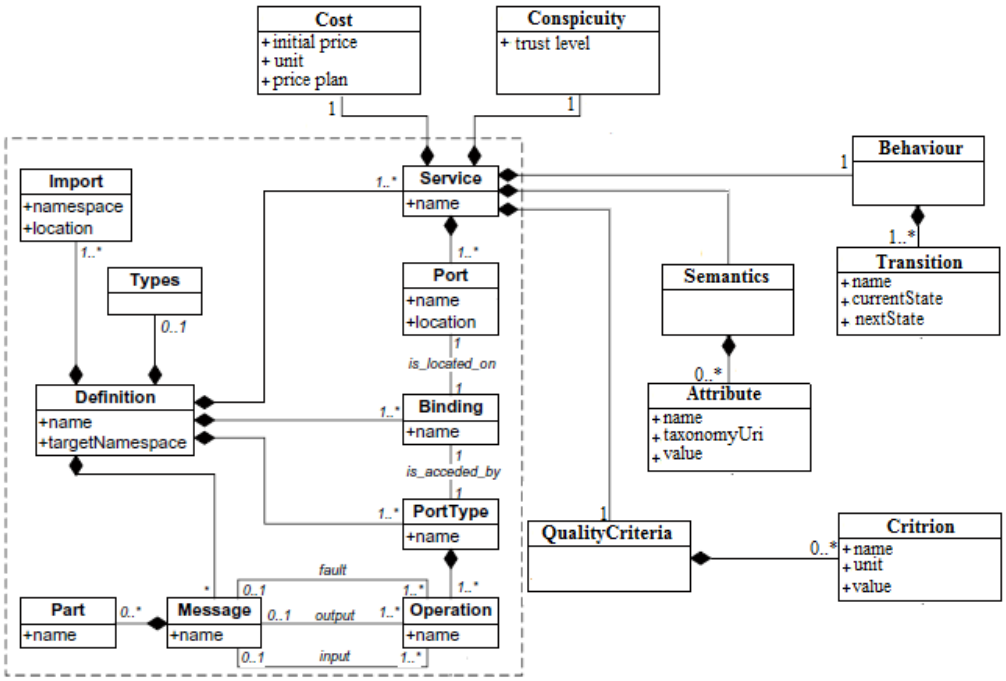


Figure 3.12: The XWSDL meta-model.

Figure 3.12 introduces the XWSDL meta-model, as an extension to WSDL meta-model, from which the XWSDL XML Schema is derived. In fact, the XML schema defines the XWSDL language, and in the same respect, the meta-model introduced in Figure 3.12 is used to define the schema. Representing an XML-based language in terms of a meta-model allows to enhance its comprehensibility

[1]http://www.omg.org/mof/.

and to facilitate its extension [43]. The basic WSDL meta-model is represented in the portion of Figure 3.12 bounded by a dashed line rectangle. Note that some classes related to specific documenting and extensibility features of XML are removed from this basic WSDL meta-model, for brevity and readability. The other classes and associations outside the dashed lined area in Figure 3.12 indicate our extension of the WSDL meta-model to include the description of C3Q for a web service. In other words, all classes and associations represented in Figure 3.12, both inside and outside the dashed lined area, form the XWSDL meta-model. Obviously, the multiplicities of 0..1 or 0..* indicate optional associations, while associations with multiplicities of 1 or 1..* show required associations. This means, for example, that *transition* is required for the *Behavior* class, while *Attribute* is optional for *Semantics* (see Figure 3.12).

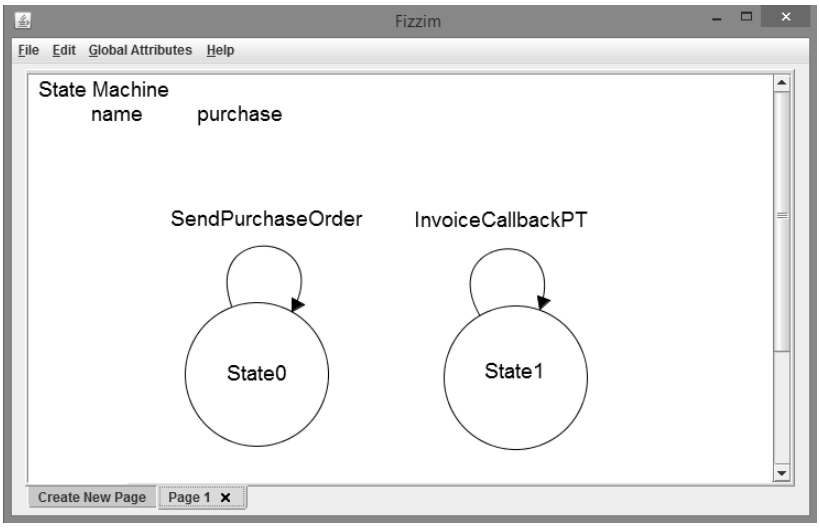


Figure 3.13: The preliminary behavioral-specification of "Purchase" service.

Since XWSDL is a lightweight extension of standard WSDL, the existing WSDL documents can easily be enriched without altering their original content. The introduced meta-model of XWSDL assists service providers in order to transform their WSDL documents to the proposed XWSDL descriptions.

3.5 Graphical User Interface

As mentioned earlier, a WSDL document is our selected standard to represent the required syntactical elements of the meta-data for web services. As such, service providers can edit/load and then parse WSDL documents of their web services, and finally enrich a document according to suggested XWSDL schema. All of these activities should be done within the "Specification Module of the proposed implementation architecture presented in Figure 2.10 of Chapter 2.

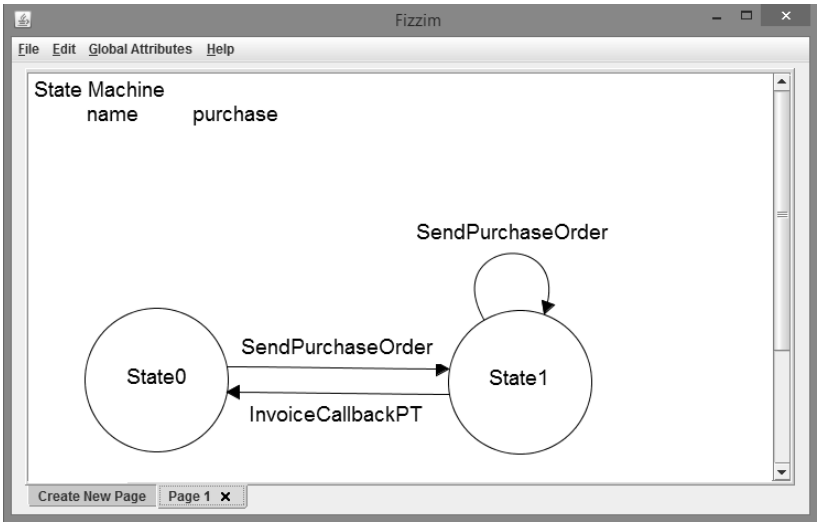


Figure 3.14: The revised version of the behavioral description of Figure 3.13.

We have implemented in Java a GUI to ease the users task of behavioral specification of services and to allow their visualization. For this, we have extended *Fizzim*, which is an open-source graphical finite state machine (FSM) design tool [171]. Fizzim is developed in Java, with its back-end in Perl that is used for its portability and ease of modification. Our extension of the Fizzim tool supports opening of a WSDL document of a service as input, and then automatically draws a preliminary design of the service's behavioral specification. This preliminary specification assumes each operation of the service as a self-loop transition on a single state, which means the execution of each operation is independent of the other operations. Figure 3.13 shows a screen-shot example of a preliminary behavioral-specification, obtained by loading a *purchase.wsdl* document. The WSDL document describes that there are two operations for this service, namely "sendPurchaseOrder" and "invoiceCallbackPT", so two self-loop transitions for them are drawn automatically in the preliminary behavioral-specification of this service. Then, the user can exploit our developed tool to interactively adjust the constraint automaton to reflect the relation between these operations as states and transitions that specify the concise behavior of the represented service. For example, Figure 3.14 is a revised version of the description specified in Figure 3.13, as specified by the user. Finally, the designed graphical description of the service behavior is exported as an XWSDL document for this service. We call our developed GUI Fizzim+B, for behavior.

3.6 Registration of business Services

AS we mentioned in Figure 2.10 a specified business service, including both atomic and composite services, should be registered in a service registry in the VO.

Figure 3.15 shows that all information specified for services would be captured in this Service Registry. As such, the VO Service Registry supports two kinds

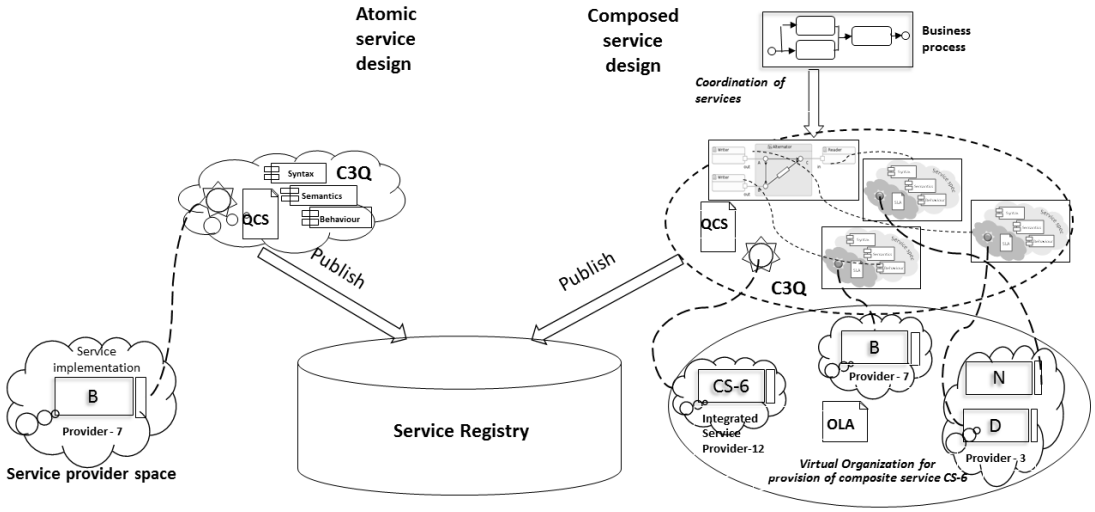


Figure 3.15: Registration of Atomic & Composite Service.

of registration corresponding to the atomic services and composed/composite services. Each atomic service has one functional service specification including its syntax, semantics and behavior, while each of component of a composite service has a different functional specification. These functional specifications together with the other parts of C3Q, as they are described by the service provider, would be captured in this registry in order to publish the service. Furthermore, the coordination pattern of the component services in every composite service could also be captured and registered in the service registry for reusability of that composite service.

3.7 Conclusion

In this chapter, we presented an extension and improvements to the current web service description approaches and standards, in order to support more efficient service discovery and composition in VOs. First, we presented a data model, C3Q, to represent the various information needed for the description of BSs as web services. C3Q is considered as the competency model for reusability within the VOs.

Then, we introduced a light extension of WSDL that we call XWSDL, to incorporate the C3Q model in the specification of web services. To the best of our knowledge, XWSDL is the first model that provides a comprehensive description

of capabilities over web services, and that highlights the important role of service behavior in the realization of the semi-automated service oriented computing. Since XWSDL is a lightweight extension of standard WSDL, the existing WSDL documents can easily be enriched with it, without altering their original content. The meta-model of XWSDL is also presented here that assists transforming WSDL documents to the proposed XWSDL descriptions, through our Fizzim+B tool. In XWSDL, the power and flexibility of the C3Q model have been combined with the simplicity and convenience of standard WSDL, thus reaching the right balance between flexibility and expressiveness for VO services.

Finally, we developed a GUI, which assists service designers to describe and visualize the behavioral specification of the web services correctly and easily.

Chapter 4

QoS-aware behavior-based Services Discovery

Parts of the material in this chapter is published in the following papers:

- Sargolzaei, M., Santini, F., Arbab, F. and Afsarmanesh, H., 2017. A Tool for QoS-aware Behaviour-based Discovery of Approximately Matching web services. Submitted to the International Journal on Software and Systems Modeling (SoSyM). Springer Berlin Heidelberg.
- Sargolzaei, M., Santini, F., Arbab, F. and Afsarmanesh, H., 2013. A tool for behaviour-based discovery of approximately matching web services. In International Conference on Software Engineering and Formal Methods (pp. 152-166). Springer Berlin Heidelberg.

In this chapter, we present a tool that is able to discover stateful web services in a registry, ranked according to a similarity score that expresses the affinities between each service and the user-submitted query. To determine these affinities, we also take the service behavior into account, both of the user's query, and of the web services. In fact, a web service behavior, addressing the names of its service operations, their order of invocation, and their parameters, may differ from those required by the user's query, but still it may be identified as similar so long as its operations collectively represent similar behavior. We use soft constraints to formalize the requirements that a user expresses in her query, the solutions to which accommodate her needs in the best possible way. We argue that a proper formalization of the behavior of many services that are commonly thought of as stateless, in fact requires a stateful representation (see Section 3.3.3). As such, our method and our tool can accommodate discovery of these services better than alternatives that consider them as stateless. Our tool uses a procedure to assess an approximate operational-similarity score among a set of services requested by their Soft Constraint Automata, which we use as the formal model of service behavior. As such, the discovery is modeled as a Constraint Optimization

Problem (COP) [138]. Finally, we enhance our tool by also considering a set of QoS metrics to further meet the user's needs.

4.1 Introduction

Web services (WSs) [8] constitute a typical example in the *Service Oriented Computing* (SOC) paradigm. WS discovery is the process of finding a suitable WS for a given task. To enable a consumer to use a service, its provider usually augments a WS endpoint with an interface description, using the *web service Description Language* (*WSDL*[1]). In a loosely-coupled environment such as SOC, automatic discovery becomes even more complex, since users' decisions must be supported by taking into account a similarity score that describes the affinity between a user's requested service (the query) and the specifications of a number of actual services available in the environment.

Although several researchers have tackled this problem and some search tools (e.g., [133]) have achieved good results, very few (see Section 4.2) consider the behavioral signature of a service that describes the sequence of operations in which a user is actually interested. This is partly due to the unavoidable limitations of today's standard specifications, e.g., WSDL, which do not encompass such aspects. Despite this fact, the behavior of stateful services represents a very important issue to be considered during service discovery, to provide users with an additional means to refine the search in such a diverse environment.

The impact of considering stateful behavior in search of services is indeed broader than it may seem at first. As we argue in Section 3.3.3, our notion of stateful services in fact covers a much wider class of services, and includes many of those that are commonly considered as stateless.

We first describe a formal framework (originally introduced in [15]) that, during the search procedure, considers both a description of the requested (stateful) service behavior, and a global similarity score between services and the queries. This underlying framework consists of *Soft Constraint Automata* (*SCA*), where semiring-based soft constraints (see Section 4.3) enhance classical (not soft) CA [18] with a parametric and computational framework that can be used to express the optimal desired similarity between a query and a service. In our work, we use CA as our base formalism for the specification of WS behavior. Since CA models are supported by our related tool in [61], it can readily be extended to also support soft constraints. Overall, we observe that automata models are more understandable and readable for engineers than other models. However, we do not unavoidably depend on CA, and in principle we can use other formalisms, and then internally convert their behavior to CA for our discovery purposes.

The main contribution of the work reported in this chapter is the design and implementation of such a framework using an approximate operational-similarity

[1] web services Description Language: `https://www.w3.org/TR/wsdl`.

evaluation method, applied to two SCAs. We implement this inexact comparison between a query and a service, as a *Constraint Optimization Problem* (*COP*), by using JaCoP libraries[2]. We are eventually able to rank all search results according to their similarity with a proposed query. In this way, we can benefit from off-the-shelf techniques with roots in Artificial Intelligence (AI), in order to tackle the complexity of search over large databases. To evaluate a similarity score we use different metrics to measure the syntactical distance between operations and between parameter names (see Section 4.6), e.g., between "*getWeather*" and "*g_weather*". These values are then automatically cast into soft constraints as semiring values (see Section 4.3), to enable parametric composition and optimization during the process of discovery. Thus, a user is enabled to eventually choose a service that better adheres to his needs than the other ones in a registry.

Exploitation of the service behavior during a search process applying a formal framework, represents the main feature of our tool. SCA represent the formal model that we use to represent behaviors: the different states of an SCA represent different states of a stateful service and/or query. Relying on SCA allows us to have a framework that comes along with a set of sound operators for composition of queries [15].

Furthermore, we describe a QoS-based service recommendation mechanism besides our discovery tool, to assist users in service selection. The reason for this feature is that non-functional properties such as QoS parameters play an important role in user's selection. Due to already having scores that represent functional similarity, using further QoS metrics in the same framework comes at a reduced cost. We propose a lexicographic composition of soft constraints to capture the trade-off of preferences in selecting the best fitting WS. We evaluate the results of our proposed tool in Chapter 6.

The rest of this chapter is structured as follows. In Section 4.2, we report on the related work. In Section 4.3 we summarise the background on semiring-based soft constraints [24], as well as the background on SCA [15]. Section 4.4 shows some examples of how to use SCA to represent the behavior of services and the similarity between their operation and parameter names. In Section 4.5 we describe the architecture of a tool that implements the search introduced in Section 4.4. In Section 4.6 we focus on how we measure the similarity between two different behavioral signatures. In Section 4.7 we explain our QoS-based service recommendation method that assists users with service selection. Finally, in Section 4.8 we draw final conclusions and explain our future work.

4.2 Related Work

Compared to the work reported in the literature, the solutions in our approach are more general, compact, and comprehensive, since it can encompasses any

[2]Java Constraint Programming solver (JaCoP): `http://www.jacop.eu`.

semiring-like metrics, and the whole framework is expressively modeled and solved using Constraint Programming [137]. Moreover, elaborating on a formal framework allows us to easily check properties of services/queries (e.g., to model-check or bi/simulate them [15]), or to use join and hide operators for their composition and abstraction [15]. A first step towards this work has been developed in [16]. The remaining of this section addresses and briefly analyses the most relevant state of the art research. The modeling and verification of long-running transaction involving composed WSs proposed in [90] and [117] are compatible with our framework. Sharing the same underlying formal model also allows us to apply model-based testing techniques such as in [91], and compliance verification and analysis as addressed in [89]. Nevertheless, most related literature report on more ad-hoc engineered, and specific solutions, instead of formal solutions, which consequently are less amenable to formal reasoning.

In [153] the authors propose a new behavior model for services using automata and logic formalisms. Roughly, the model associates messages with activities, and adopts the IOPR model (i.e., Input, Output, Precondition, Result) of *OWL-S*[3] to describe activities. While the authors use an automaton structure to model service behavior, similarity-based search is not mentioned in [153].

In [172] the authors present an approach that supports service discovery based on structural and behavioral service models, as well as quality constraints and contextual information. However, the behaviors are matched through a sub-graph isomorphism algorithm, thus vertices cannot be merged or deleted when needed, as in the case of a sub-graph epimorphism.

In [67] the problem of behavioral matching is translated to a graph matching problem, and existing algorithms are only adapted for this purpose.

The model presented in [160] relies on a simple and extensible keyword-based query language and enables efficient retrieval of approximate results, including approximate service compositions. Further on, since representing all possible compositions can result in an exponentially-sized index, the authors investigate clustering methods to provide a scalable mechanism for service indexing.

In [22] the authors propose a crisp translation from interface description of services to classical crisp *Constraint Satisfaction Problems* (*CSPs*). The work in [22] does not consider service behavior however, and it does not support a quantitative reasoning on similarity/preference involving different services.

In [169] a semiring-based framework is used to model and compose QoS features of WSs. However, no notion of similarity relationship is provided in [169].

In [51], the authors propose a novel clustering algorithm that groups the names of parameters of service operations into semantically meaningful concepts. These concepts are then leveraged to determine similarity of inputs (or outputs) of service operations.

In [124] the authors propose a framework of fuzzy query languages for fuzzy

[3]OWL-S: Semantic Markup for Services, 2004: `www.w3.org/Submission/OWL-S/`.

ontologies, and present query answering algorithms for these query languages over the fuzzy *DL-Lite* ontologies.

In [72] the authors propose a metric to measure the similarity of semantic services annotated with an *OWL ontology*. They calculate similarity by defining the intrinsic information value of a service description based on the "inferencibility" of each of *OWL Lite* constructs.

The authors in [133] show a method of service retrieval called *URBE* (*UDDI Registry By Example*). The retrieval is based on the evaluation of similarity between the interfaces of WSs. The algorithm used in *URBE* combines the analysis of the structure of a WS and the terms used inside it.

Baresi et al. [19] introduce DREAM as an innovative infrastructure for the distributed publication and discovery of WSs. DREAM provides partial solutions for users requests through a set of matchmakers such as WSDL-based Matchmaker and XPath-based matchmaker. The ability to adding new matchmakers gives more flexibility to DREAM, but considering the experimental results, they should improve the precision and recall without affecting the flexibility. Moreover, this version of DREAM could not manage the behavioral and also non-functional aspects of services.

In order to consider QoS metrics as additional criteria to select services from a set of functionally equivalent candidates, we can simply specify QoS properties as meta-attributes. However, the semantics of such schemes is too weak to allow reasoning about QoS properties. In [116], the authors extend constraint automata (CA) with Q-algebras to define *Quantitative Constraint Automata* (*QCA*) to specify the QoS properties of services for an optimized service selection and composition. Also in [116], they introduce *Quasi-Classical Temporal Logic* (*QCTL*) as a logic for reasoning about both behavioral and QoS aspects of services modeled by QCA. Because QCA and our work share constraint automata as their base model, we can extend our soft constraint automata model by adopting and further extending the QCA, which will enable us to use QCTL logic for a richer form of reasoning about the properties of services, when needed.

Preference modeling is an important issue in various fields such as economics, mathematics, informatics, and even psychology. We would deal with preferences when we have to make choices on behalf of users [27].

In [63] the authors discuss soft constraint techniques and using the lexicographic order for preferences. This work is pivotal for the use of soft constraints when dealing with problems with a number of criteria that must be satisfied with a fixed order of importance.

Managing tradeoffs of the QoS preferences has been addressed in [77] using a lexicographic-based specification language for expressing the QoS preferences.

This is also addressed in [109] based on a model of lexicographic semi-order. Although lexicographic semi-order seems to work better in our context by considering the threshold for each criterion, unfortunately it is not associative, and therefore could not be applied.

To the best of our knowledge, none of the related work presented in this section comes with tools for direct behavioral discovery and approximate matching of stateful web services. Our approach provides an achievement in which the VO members can find such services more accurately within SOCN. Moreover, with our tool, a user is able to compose all the constraints representing his or her required functional and non-functional requirements in a given lexicographic order, i.e., different criteria have different precedences. While most of the QoS-aware related work compute QoS ranking as an extra criterion for search that may impact the results by giving higher ranking to completely irrelevant services that have high QoS values, we avoid this problem by composing both functional and non-functional requirements as semirings in our approach. This advantage is granted by the formal definition of our proposed lexicographic ordering on soft constraints.

4.3 Soft Constraint Automata

Semiring-based Soft Constraints. A *c-semiring* [24] (simply semiring in the sequel) is a tuple $S = \langle A, +, \times, 0, 1\rangle$, where A is a possibly infinite set with two special elements $0, 1 \in A$ (respectively the bottom and top elements of A) and with two operations $+$ and $\times$ that satisfy certain properties over A: $+$ is commutative, associative, idempotent, closed, with 0 as its unit element and 1 as its absorbing element; $\times$ is closed, associative, commutative, distributes over $+$, 1 is its unit element, and 0 is its absorbing element. The $+$ operation defines a partial order $\leqslant_S$ over A such that $a \leqslant_S b$ iff $a + b = b$; we write $a \leqslant_S b$ if b represents a value *better* than a. Moreover, $+$ and $\times$ are monotone on $\leqslant_S$, 0 is the min of the partial order and 1 its max, $\langle A, \leqslant_S\rangle$ is a complete lattice and $+$ is its *least upper bound* operator (i.e., $a + b = lub(a, b)$) [24].

Some practical instantiations of the generic semiring structure are the *boolean* $\langle\{false, true\}, \vee, \wedge, false, true\rangle$, *fuzzy* $\langle$[0..1],max, min, 0, 1$\rangle$, *probabilistic* $\langle$[0..1], max, $\hat{\times}, 0, 1\rangle$ and *weighted* $\langle\mathbb{R}^+ \cup \{+\infty\}, min, \hat{+}, \infty, 0\rangle$ (where $\hat{\times}$ and $\hat{+}$ respectively represent the arithmetic multiplication and addition).

A *soft constraint* [24] may be seen as a constraint where each instantiation of its variables has an associated preference. An example of two constraints defined over the *weighted* semiring is given in Figure 4.1b. Given $S = \langle A, +, \times, 0, 1\rangle$ and an ordered finite set of variables V over a domain D, a soft constraint is a function that, given an assignment $\eta : V \rightarrow D$ of the variables, returns a value of the semiring, i.e., $c : (V \rightarrow D) \rightarrow A$. Let $C = \{c \mid c : D^{|I \subseteq V|} \rightarrow A\}$ be the set of all possible constraints that can be built starting from S, D and V: any function in C depends on the assignment of only a (possibly empty) finite subset I of V, called the *support*, or *scope*, of the constraint. For instance, a binary constraint $c_{x,y}$ (i.e., $\{x, y\} = I \subseteq V$) is defined on the support $supp(c) = \{x, y\}$. Note that $c\eta[v = d]$ means $c\eta'$ where η' is η modified with the assignment $v = d$. Note also that $c\eta$ is the application of a constraint function $c : (V \rightarrow D) \rightarrow A$ to a function

$\eta : V \to D$; what we obtain is, thus, a semiring value $c\eta = a$. The constraint function $\bar{a}$ always returns the value $a \in A$ for all assignments of domain values, e.g., the $\bar{0}$ and $\bar{1}$ functions always return 0 and 1 respectively.

Given the set C, the combination function $\otimes : C \times C \to C$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$ [24]; $supp(c_1 \otimes c_2) = supp(c_1) \cup supp(c_2)$. Likewise, the combination function $\oplus : C \oplus C \to C$ is defined as $(c_1 \oplus c_2)\eta = c_1\eta + c_2\eta$ [24]; $supp(c_1 \oplus c_2) = supp(c_1) \cup supp(c_2)$. Informally, $\otimes/\oplus$ builds a new constraint that associates with each tuple of domain values for such variables a semiring element that is obtained by multiplying/summing the elements associated by the original constraints to the appropriate sub-tuples.

The search engine of the tool we present in Section 4.5 relies on the solution of *Soft Constraint Satisfaction Problems* (*SCSPs*) [24], which can be considered as COPs. An SCSP is defined as a quadruple $P = \langle S, V, D, C \rangle$, where S is the adopted semiring, V the set of variables with domain D, and C is the constraint set. $Sol(P) = \bigotimes C$ collects all solutions of P, each associated with a similarity value $s \in S$. Soft constraints are also used to define SCA.

Soft Constraint Automata. Constraint Automata were introduced in [18] as a formalism to describe the behavior and possible data flow in coordination models (e.g., Reo [18]); they can be considered as acceptors of *Timed Data Streams* (*TDS*) [14, 18]. We now recall the definition of *TDS* from [14], while extending it using the softness notions described in Section 4.3: we name this result as *Timed Weighted Data Streams* (*TWDS*), which correspond to the languages recognised by *SCA*.[4] For convenience, we consider only infinite behavior and infinite streams that correspond to infinite "runs" of our soft automata, omitting final states, including deadlocks.

4.3.1. Definition. [Timed Weighted Data Streams] Let *Data* be a data set, and for any set X, let X^ω denote the set of infinite sequences over X; given a semiring $S = \langle A, +, \times, 0, 1 \rangle$, a Timed Weighted Data Stream (*TWDS*) is a triplet:

$$\langle \lambda, l, a \rangle \in Data^\omega \times \mathbb{R}_+^\omega \times A^\omega \text{ such that, } \forall k \geqslant 0 : l(k) < l(k+1) \text{ and } \lim_{k \to +\infty} l(k) = +\infty$$

Thus, a *TWDS* triplet $\langle \lambda, l, a \rangle$ consists of a data stream $\lambda \in Data^w$, a time stream $l \in \mathbb{R}_+^\omega$, and a preference stream $a \in A^\omega$; k is a natural number that is used to enumerate the elements of each stream. The time stream l indicates, for each data item $\lambda(k)$, the moment $l(k)$ at which it is exchanged (i.e., being input or output), while the $a(k)$ is a preference score related to $\lambda(k)$.

In [15] we paved the way to the definition of *Soft Constraint Automata* (SCA), which represent the theoretical foundation behind our tool. Use a finite set N of names, e.g., $N = \{n_1, \ldots, n_p\}$, where n_i ($i \in 1..p$) is the i-th input/output port. The transitions of SCA are labeled with pairs consisting of a non-empty

[4] TWDSs do not imply time constraints, and thus our (soft) CA are not "timed" [18].

subset $N \subseteq N$ and a soft (instead of crisp as in [18]) data-constraint c. Soft data-constraints can be viewed as an association of data assignments with a preference for that assignment. Formally,

4.3.2. Definition. [Soft Data-Constraints] A soft data-constraint over a set of port names $\mathcal{N}$ and data values $Data$, is an expression produced by the following grammar, with $\mathbf{c}$ as its distinguished symbol:

$$\mathbf{c} ::= \mathbf{f} \mid \mathbf{f} \oplus \mathbf{f}$$
$$\mathbf{f} ::= \bar{\mathbf{0}} \mid \bar{\mathbf{1}} \mid \mathbf{c} \otimes \mathbf{c} \mid (\mathbf{c}) \mid c$$

where for $N \subseteq \mathcal{N}$, c represents a function $c : (\{d_n \mid n \in N\} \to Data) \to A$ over a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, the set of variables $\{d_n \mid n \in N\}$ constitute the support of the constraint, and $\{d_n \mid n \in N\} \to Data$ is a function that associates with every variable d_n in this support, a data item $v \in Data$ that passes through the port $n \in N$.

Informally, a soft data-constraint is a function that returns a preference value $a \in A$ given an assignment for the variables $\{d_n \mid n \in N\}$ in its support. In the sequel, we write $SDC(N, Data)$, for a non-empty subset N of N, to denote the set of soft data-constraints. We will use SDC as an abbreviation for $SDC(\mathcal{N}, Data)$. Note that in Definition 4.3.2 we assume a global data domain $Data$ for all names, but, alternatively, we can assign a data domain $Data_n$ for every variable d_n.

We state that an assignment η for the variables $\{d_n \mid n \in N\}$ satisfies c with a preference of $a \in A$, if $c\eta = a$.

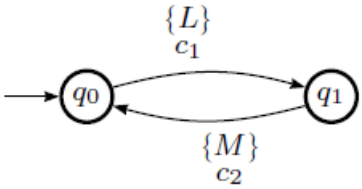
In Definition 4.3.3 we define SCA. Note that by using the *boolean* semiring, thus within the same semiring-based framework, we can exactly model the "crisp" data-constraints presented in the original definition of CA [18]. Therefore, CA are subsumed by Definition 4.3.3. Note also that weighted automata, with weights taken from a proper semiring, have already been defined in the literature [52]; in SCA, weights are determined by a constraint function instead.

4.3.3. Definition. [Soft Constraint Automata] A Soft Constraint Automaton over a domain $Data$, is a tuple $T_S = (Q, N, \longrightarrow, Q_0, S)$ where *i)* S is a semiring $\langle A, +, \times, 0, 1 \rangle$, *ii)* Q is a finite set of states, *iii)* N is a finite set of names, *iv)* $\longrightarrow$ is a finite subset of $Q \times 2^N \times SDC \times Q$, called the transition relation of T_S, and *v)* $Q_0 \subseteq Q$ is the set of initial states. We write $q \xrightarrow{N,c} p$ instead of $(q, N, c, p) \in \longrightarrow$. We call N the name-set and c the guard of the transition. For every transition $q \xrightarrow{N,c} p$ we require that i) $N \neq \varnothing$, and ii) $c \in SDC(N, Data)$ (see Definition 4.3.2). T_S is called finite iff $Q, \longrightarrow$ and the underlying data-domain $Data$ are finite.

The intuitive meaning of an SCA T_S as an operational model for service queries is similar to the interpretation of labeled transition systems as formal models

for reactive systems. The states represent the configurations of a service. The transitions represent the possible one-step behavior, where the meaning of $q \xrightarrow{N,c} p$ is that, in configuration q, the ports in $n \in N$ have the possibility of performing I/O operations that satisfy the soft guard c and that leads from configuration q to p, while the ports in $N \backslash N$ do not perform any I/O operation. Each assignment of variables $\{d_n \mid n \in N\}$ represents the data associated with ports in N, i.e., the data exchanged by the I/O operations through ports in N.

In Figure 4.1a we show an example of a (deterministic) SCA. In Figure 4.1b we define the *weighted* constraints c_1 and c_2 that describe the preference (e.g., a monetary cost) for the two transitions in Figure 4.1a, e.g., $c_1(d_L = 2) = 5$.



$c_1 : (\{d_L\} \to \mathbb{N}) \to \mathbb{R}^+$ s.t. $c_1(d_L) = d_L + 3$

$c_2 : (\{d_M\} \to \mathbb{N}) \to \mathbb{R}^+$ s.t. $c_2(d_M) = d_M + 5$

(a) A Soft Constraint Automaton. (b) c_1 and c_2 in Figure 4.1a.

Figure 4.1: An example of a SCA, as well as its associated weighted constraints.

In [15] we have also softened the synchronization constraints associated with port names in N over the transitions. This allows for different service operations to be considered somehow similar for the purposes of a user's query. Note that a similar service can be used, e.g., when the "preferred" one is down due to a fault, or when it offers bad performance, e.g., due to the high number of requests. Definition 4.3.4 formalises the notion of soft synchronization-constraint.

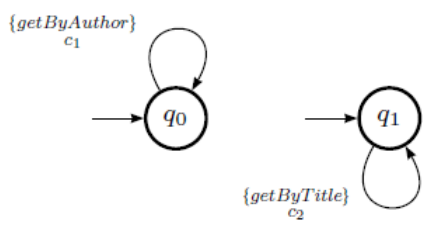
4.3.4. Definition. [Soft synchronization-constraint] A soft synchronization- constraint is a function $c : (V \to N) \to A$ defined over a semiring $S = \langle A, +, \times, 0, 1 \rangle$, where V is a finite set of variables for each I/O port, and N is the set of I/O port names of the SCA.

4.4 Representing the behavior of Services with SCA

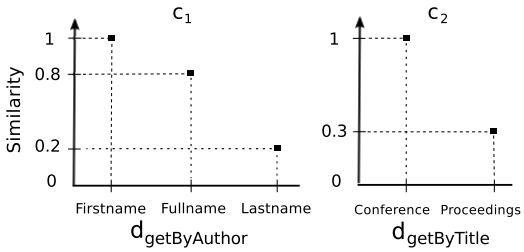
In this section we show how the formal framework presented in Section 4.3 (i.e., SCA) can be used to consider a similarity score between a user's query and the service descriptions in a registry, in order to find the best possible matches for the user.

We begin by considering how parameters of operations can be associated with a score value that describes the similarity between a user's request and an actual service description in a registry. We suppose to have two different queries:

the first, `getByAuthor(Firstname)`, which is used to search for conference papers using the `Firstname` (i.e., the parameter name) of one of its authors; the name of our invoked service operation is, then `getByAuthor`. The second query, `getByTitle(Conference)`, searches for conference papers, using the title of the `Conference` wherein the paper has been published; the name of our invoked operation is `getByTitle`. These two queries are represented as the SCA (see Section 4.3) q_0 and q_1, in Figure 4.2a. Soft constraints c_1 and c_2 in Figure 4.2b, define a similarity score between the parameter name used in a query and all parameter names in the registry (for the same operation name, i.e., either `getByAuthor` or `getByTitle`). These similarity scores can be modeled with the *fuzzy* semiring $\langle [0..1], \max, \min, 0, 1 \rangle$ wherein the aim is to maximise the similarity ($+ \equiv \max$) between a request and a service returned as a matching result. Constraint c_1 in Figure 4.2b states that similarity is full if a `getByAuthor` operation in the registry takes `Firstname` as parameter (since 1 is the top preference of the *fuzzy* semiring), less perfect, that is 0.8, if it takes `Fullname` (since usually, `Fullname` includes `Firstname`), or even less perfect, that is 0.2, if it takes `Lastname` only. Similar considerations apply to the operation name `getByTitle` (see Figure 4.2a) and c_2 in Figure 4.2b. The similarity scores are automatically extracted as it is explained in Section 4.5.



(a) Two soft Constraint Automata representing two different queries.



(b) The definitions of c_1 and c_2 in Figure 4.2a.

Figure 4.2: Two example queries represented by soft Constraint Automata.

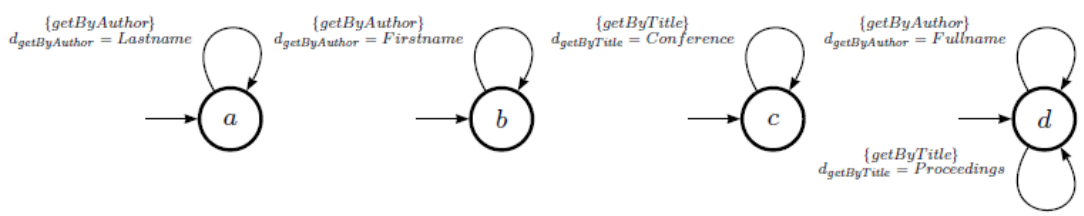


Figure 4.3: A set of registered services for the queries in Figure 4.2a; d performs both kinds of search (by author and by title).

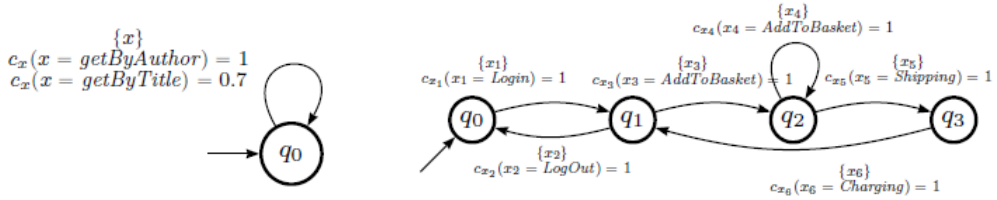
Suppose now that our registry contains the four services represented in Figure 4.3. All these services are stateless, i.e., their SCAs have a single state each. For instance, here service a has only one invokable operation whose name is `getByAuthor`, which takes `Lastname` as parameter. Service d has two distinct operations, `getByAuthor` and `getByTitle`.

According to the similarity scores expressed by c_1 and c_2 in Figure 4.2b, queries q_0 and q_1 in Figure 4.2a return different result values for each of these four operations/services, depending on the instantiation of variables $d_{getByAuthor}$ and $d_{getByTitle}$. Considering q_0, services a, b, and d result respective preferences of 0.2, 1, and 0.8. If query q_1 is used instead, the possible results are services c and d, with respective preferences of 1 and 0.3. When more than one service is returned as the result of a search, the end user has the freedom to choose the best one according to his preferences, for instance: for the first query q_0, the user can opt for service b, which corresponds to a preference of 1 (i.e., the top preference), while for query q_1 the user can opt for c (top preference as well).

We now move from the parameter names to operation names, and show that by using soft synchronization constraints (see Definition 4.3.4), we can also compute a similarity score among them. For example, suppose that a user queries q_0 in Figure 4.2a. The possible results are services a, b and d in the registry of Figure 4.3, since service c has an operation named `getByTitle`, different from `getByAuthor`. However, these two services are also somewhat similar, since they both return a paper even if the search is based either on the author or on the conference. As a result, a user may be satisfied also by retrieving (and then, using) service c. This can be accomplished with the query in Figure 4.4a, where $c_x(x = getByAuthor) = 1$, and $c_x(x = getByTitle) = 0.7$ are its constraints. Note that we no longer deal with constraints on parameter names, but on operation names. Then, we can also look for services that have similar operations, not only similar parameters in operations.

However, our main goal is to compute a similarity score considering also the behavior of queries and services. For instance consider a query that may require stateful services such as the query in Figure 4.4b, a user may need to find an on-line purchase service satisfying the following requirements: *i) shipping activity must occur before charging activity,* ii*) to purchase a product, the requester first needs to log into the system and finally log out of the system, and* iii*) filling the electronic basket of a user may consist of a succession of "add-to-basket" actions.* In Section 4.6 we will focus on this aspect.

Constraints on parameters (their data-types as well) and operation names can be straightforwardly mixed together to represent a search problem where both are taken into account simultaneously for optimization. The tool in Section 4.5 exploits this kind of search: the similarity functions represented by constraints are computed through the composition of different syntactic similarity metrics.



(a) A similarity-based query for the *Author/Title* example.

(b) A similarity-based query for the on-line purchase service.

Figure 4.4: Two examples of stateless/stateful queries.

4.5 Tool Description

We have developed a tool that is able to discover stateful web services in the registry of SOCN, ranked according to a score expressing the similarity between each service and a user-submitted query, considering the behavior specification of both the users query and the SOCN services. We call our behaviorally-based web service discovery tool BehSearch. Conceptually, BehSearch proceeds in four successive steps:

- *i*) Generate a *web service Behavior Specification* (*WSBS*) for each registered WS (a WSBS is basically a CA).
- *ii*) Process preference-oriented queries (basically represented as SCA).
- *iii*) Compute an operational similarity-score between a query and our services as an SCSP (see Section 4.3).
- *iv*) Solve this problem.

Step *i* is needed because no standard language or tool exists to specify the behavior of stateful WSs. Therefore, we define our suggested WSBS as a behavioral specification for WSs, extending WSDL with extra necessary annotations. In **step *ii***, we obtain the query from user and we process it to find the similarities between the request and the actual services in the registry. In **step *iii***, we set up an SCSP (see Section 4.3), where soft-constraint functions are assembled by using the similarity scores derived in **step *ii***; at the same time, we define those constraints that compare the two behavioral signatures (query/service), and measure their similarity. Finally, we find the best solutions for this SCSP, and we return them to the user. These steps are implemented by a set of software modules, whose global architecture is illustrated in Figure 4.5. These modules are also one by one defined below.

WSDL Parser. For our implementation, we rely on a repository of WSDL documents that are captured in a registry, i.e., the *WSDL Registry* (see Figure 4.5). WSDL is an XML-based standard for syntactical representation of

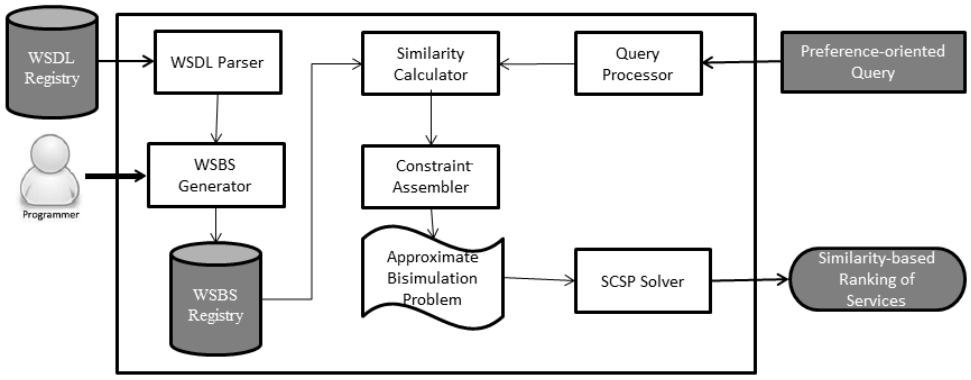


Figure 4.5: The architecture of the tool.

WSs, which currently serves as the most suitable for our purpose. We parse these XML-based documents to extract the names and interfaces of service operations using the *Axis2* technology.[5]

WSBS Generator. While a WSDL document specifies the syntax and the technical details of a service interface, it lacks the information needed to convey its behavioral aspects, as described in Section 3.3.3. In fact, a WSDL document only reveals the operation names and the names and data types of their arguments, and it does not indicate the permissible operation sequences of a service. If we know that a WS is stateless, then all of its operations are permissible in any desired order. For a stateful service however, we need to know which of its operations is or is not allowed in each of its states. As described in Section 3.3.3, we formalized the behavior of a web service (i.e., our so-called WSBS) in terms of CA [18]. The generated CA are captured as XML documents, where the <*transitions*> tags identify the structure of each automaton. In Figure 4.5, all WSBSs are stored in a *WSBS Registry*.

We can automatically extract a single-state automaton from the operations defined in a WSDL document describing a stateless WS: we use this support-tool to extract the automata for the real-world WSs used in our following experiment. For stateful WSs, we have developed an interactive tool that through a GUI allows a programmer (see Figure 4.5) to visually create the automaton states describing the behavior of a service, and tag its transitions with the operations defined in its WSDL document.

Query Processor. For search purposes, a user specifies a desired service by means of a text file, and feeds it to this module. An example of our query format is represented in Figure 4.8. This query format allows to specify all desired transitions among states, including operation names, and the names and data types of their arguments. It enables search for multiple similar services separated by the "or" operators. The tool ranks all the results in the same list (see Table 4.1).

[5] `http://axis.apache.org/axis2/java/core/`.

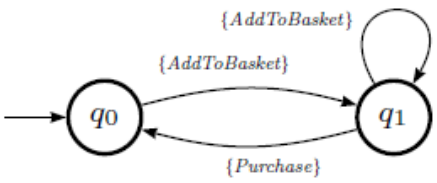


Figure 4.6: An example of WSBS.

```
q0 AddToBasket q1;
q1 AddToBasket q1;
q1 Purchase q0.
```

Figure 4.7: Text file representing the WSBS in Figure 4.6.

Finally, the tool assigns to each service description a preference score complying with what is prescribed by the user. A user may use a score in his/her query, e.g., fuzzy preferences in [0..1] to weigh all the results, as represented in Figure 4.8. Each query is represented as an SCA [15] (see Section 4.3), since preferences can be represented by soft constraints. This textual representation resembles a list of WSBSs, each of them associated with a preference score. See Table 4.1, Figure 4.8 and Figure 4.7 for a comparison.

Similarity Calculator. As Figure 4.5 shows, this module requires two inputs: the WSBSs and the processed query. It then returns three different kinds of similarity scores, which reflect the similarities between one service and one query, including: *i*) operations names, *ii*) names of input-parameters of operations, and *iii*) data types of input-parameters. We use a number of different string similarity-metrics (also known as *string distance functions*) as the functions to measure the similarity between two text-strings. We have specifically chosen and implemented three of the most widely known metrics that fit best to our purpose, the *Levenshtein Distance*, the *Matching Coefficient*, and the *QGrams Distance* [40]. Each of these metrics operates receives two input strings and returns a score estimating their similarity. Since each function returns a value in [0..1], we average these three resulted scores and merge them into a single value in [0..1].

These similarity scores are subsequently used by the *Constraint Assembler* module in Figure 4.5, in order to define the similarity functions that are translated into soft constraints, as it was explained in Section 4.4. The representation of the search problem in terms of constraints is completely constructed by the *Constraint Assembler* module, while the *Similarity Calculator* only provides it with similarity scores.

Constraint Assembler. This module produces a model of the discovery problem, in the form of operational similarity-evaluation addressed in Section 4.6, as an SCSP (see Section 4.3). To do so, it represents all preference and similarity requirements as soft constraints. In order to assemble these constraints, we use *JaCoP*, a Java library that provides a finite-domain constraint programming paradigm. We have further needed to introduce ad-hoc extensions to the crisp constraints supported by JaCoP in order to equip them with weights, and then we have exploited the possibility to minimise/maximise a given cost function to solve SCSPs. Specifically, we have expressed the WSs discovery problem as a fuzzy opti-

```
q0 Weather(City:string) q0, [1.0] or q0 Weather(Zipcode:string) q0,
[0.8]
```

Figure 4.8: A single-state query asking for the weather conditions over a `City`, or a `Zipcode`. Different user preference scores are represented within square brackets.

mization problem, by implementing the *fuzzy* semiring, i.e., $\langle [0..1], \max, \min, 0, 1 \rangle$ (see Section 4.3).

For instance, *SumWeight* is a JaCoP constraint that computes a weighted sum as the following pseudo-code: $w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 = sum$, where sum represents the global syntactic similarity between two operation in terms of the similarity between their operation names (x_1), their argument names (x_2), and their argument types (x_3). These scores are provided by the *Similarity Calculator*. Moreover, we can tune the weights w_1, w_2, and w_3 to give more or less importance to the three different parameters. In the experiments in Section 4.5, as an example we use equal weights. In Section 4.6, we discuss how to compute how similar are two behavioral signatures (query/service), and how we construct our general constraint-based model. More detailed information related this module is addressed in Section 4.6.

SCSP solver. Finally, receiving the specification of the model of the problem in terms of its variables and constraints, a search for a solution of the assembled SCSP can be started. This represents the final step (see Figure 4.5). The result can be generalized as a ranking of services in the considered registry, where at the top positions the services that are more similar to a user's request are positioned.

Experimental Results on a Stateless Scenario. In this section we show the precision results generated by our developed tool through a scenario involving stateless real WSs. Figure 4.8 shows a single-state query that searches for WSs that return the "weather" forecast for a location indicated by: either the name of a "city" (with a user's preference of 1), or its "zip-code" (preference of 0.8).

We have developed a WS crawler to find WSDL documents, and check their validations. With an open search for WSs on the Internet, we retrieved more than 2000 different WSDL documents, but only about 1000 of their corresponding WSs are valid, i.e., they work yet. The validated WSDL documents form our *WSDL Registry* in Figure 4.5.

Table 4.1 shows only the top-ten ranked experiment results, where the other WSs obtained a similarity score less than 0.3. From left to right the columns respectively show the position of WS in the final ranking, the obtained fuzzy score, the WS name, and lastly the matched service operation.

Table 4.1: The ranking of the top-ten matched WSs, based on the query represented in Figure 4.8.

Rank	Score	Name of WS	Interface of the operation
1	0.69	globalweather	$GetWeather(CityName : string)$
2	0.54	usweather	$GetWeatherReport(CityName : string)$
3	0.5	Weather	$Get\_Weather(ZIP : string)$
4	0.48	WeatherWS	$getWeather(theCityCode : string, theUserID : string)$
5	0.44	usweather	$GetWeatherReport(ZipCode : string)$
6	0.42	WeatherForecast	$GetWeatherByZipCode(ZipCode : string)$
7	0.4	WeatherForecast	$GetWeatherByPlaceName(PlaceName : string)$
8	0.36	weatherservice	$GetLiveCompactWeatherByStationID(stationid : string, un : UnitType, ACode : string)$
9	0.34	weatherinfo	$WeatherInfoByPstcode(PostCode : string)$
10	0.30	weather-area	$GetWeatherArea(id : string)$

4.6 On Comparing behavior Signatures

In this section we zoom inside the *Constraint Assembler* component that we introduced in Section 4.5. We describe how we calculate approximate behavior of a posed query against that of a service, since reaching a perfect match is quite unusual and uncommon.

The basic idea is to compute an operational similarity-score between two automata, respectively representing a query and a WS in a registry. The notion of operational similarity relation is obtained by relaxing the equality of output traces. This means instead of requiring them to be identical, we require that they remain "close". Metrics represented as semirings, in our case, essentially quantify how well a system is approximated by another, based on the distance between their observed behaviors. In this way, we are able to consider different transition-labels by estimating a similarity score between their operation interfaces, for different numbers of states. To compute such operational similarity with constraints, our approach roots in constraint-based graph matching techniques [140]; thus, we are able to "compress" or "dilate" one automaton structure into another. We take advantage of the notion of *sub-graph epimorphism*, corresponding to the application of node delete and merge operations to pass from one SCA to another, when checking operational similarity. The existence of an epimorphism from one graph to another is an NP-Complete problem [65].

In the following, we use the query example in Figure 4.9, and the service example in Figure 4.10 to describe our constraint-based model for the search. We first subdivide this description, considering how we match different elements of automata (transitions or states) and how we finally measure their overall similarity.

States. To represent our signature-match problem, for each of the query-automaton states (set Q) we define a variable that can be assigned to one or several states of a service (set S). For this purpose, we use *SetVar*, i.e., JaCoP

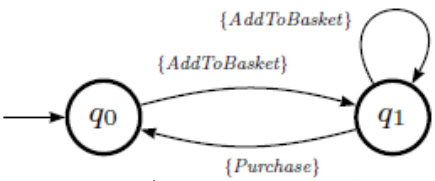


Figure 4.9: A query example.

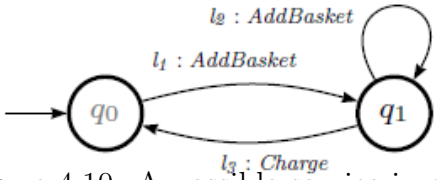


Figure 4.10: A possible service in a registry related to the query in Figure 4.9.

variables defined as ordered collections of integers. Considering our running example, one of the possible matches between these two signatures can be given by $\mathcal{M} \equiv q_0 = \{s_0, s_1, s_3\}, q_1 = \{s_2\}$. This matching is represented in Figure 4.9 and Figure 4.10 using grey and black labels for states. Clearly, the proposed modeling solution represents a relationship and not a function, since a query state can be associated with one or more service-states; on the other hand, different query states can be associated with the same service state, in case a query has more states than a service. Thus, to match the two automata we allow to "merge" together those states that are connected by a transition (e.g., s_0, s_1 and s_3 in Figure 4.10) into a single state (e.g., q_0) at the cost of incurring a certain penalty.

Transitions. In our running example, if we match the two behaviors as defined by $\mathcal{M}$, we consequently obtain a match for the transitions (and their labels) as well. Our model has a variable (*IntVar*, in JacoP) for each of the transitions in a query automaton; considering the example in Figure 4.9, we have three variables l_1, l_2, l_3. In Figure 4.9 and Figure 4.10 we label each transition with its identifier $(l_1, \ldots, l_3, m_1, \ldots, m_6)$, and a string that represents its related operation-name. Note that in this example, we ignore parameter names and types for the sake of brevity. Thus, the full match-characterisation is now $\mathcal{M} \equiv q_0 = \{s_0, s_1, s_3\}, q_1 = \{s_2\}, l_1 = m_2, l_2 = m_3, l_3 = m_5$. Note that, if a query has more transitions than a service, it may happen to be impossible to match all of them; for this reason, since we need to assign each of the variables in order to find a solution, we assign a mark *NM* (i.e., *Not Matched*) to unpaired transitions.

Automata Epimorphism. Algorithm 1 shows our approach to find subgraph epimorphism of the automata to match behavior specification of the query and services. The idea is that we can merge two or more neighboring states, i.e., states connected by transitions, to one single state. Every such merged state needs to adjust its transitions. In this way every in-coming transition to one of the combined states comes to the new state, namely the merged one. Similarly, outgoing transitions of both states become outgoing transitions of the combined state.

For example, we can find two automata epimorphisms for the service represented in Figure 4.10 with the corresponding query-automaton depicted in Figure 4.9. The state cardinality of the service-automaton is four ($|S| = 4$), while it is two for the query-automaton ($|Q| = 2$). Therefore there are two ways to

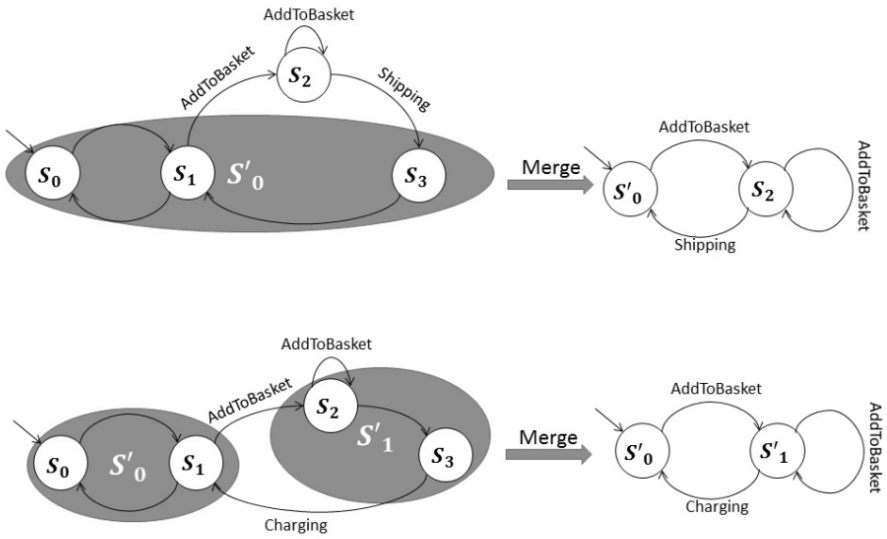


Figure 4.11: Two possible subgraph epimorphisms of the Figure 4.10.

merge: (1) merge three neighboring states into one state besides the other state, (2) merge two neighboring states into one state, and merge the remaining two neighboring states into another single state.

Figure 4.11 shows a second example in which two possible automata epimorphisms for this example. These two results can be obtained through Algorithm 1 at the following steps. Note as a reminder that $|Q| = 2$ and $|S| = 4$:

The first sub-graph epimorphism is the result of step $i = 2$, so three (i.e., $i+1$) neighboring states are merged into one state, i.e., state s_0 is merged with state s_1 and state s_3, and then becomes s_0', and the other state, i.e., state s_2 forms the other node of the new automaton. By ignoring the dissolved transitions (i.e., the internal transitions among the merged nodes), the transition matrix of the new merged automaton is:

$$T = \begin{bmatrix} Null & AddToBasket \\ Shipping & AddToBasket \end{bmatrix}$$

The second sub-graph epimorphism is the result of step $i = 1$, so two (i.e, $i + 1$) neighboring states are merged into one state (i.e., state s_0 is merged with state s_1, and then becomes s_0'), and the other two states form the other node of the new automaton: state s_2 is merged with state s_3, and then becomes s_1'. By ignoring the dissolved transitions (i.e., internal transitions among the merged nodes) the transition matrix of the new automaton (merged one) is:

$$T = \begin{bmatrix} Null & AddToBasket \\ Charging & AddToBasket \end{bmatrix}$$

In order to compare the behavior of the query and services, we replace the transition matrix T of each automaton by the adjacency matrix A, where $A[i,j] = 0$ if $T[i,j]$ is Null, and 1 otherwise. While the transition matrices of the two possible automata epimorphisms of the above example are different (at $T[1,1]$), their adjacency matrices both are the same with the adjacency matrix of the query and it is:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

It means that new automata epimorphisms are behaviorally matched with the query, but due to the incurring of the penalty factor (Q/S), their state similarity-scores would be 0.5. Note as a reminder that $|Q| = 2$ and $|S| = 4$. The transition similarity-scores for these two alternatives, which are derived from a comparison between matched labels, would be respectively 0.36 and 0.43. Therefore, the second epimorphism that results in a higher similarity score is considered during the discovery process for the requested service.

Operational-similarity of the Match. In this paragraph we show how to compute a global similarity score Γ for a match $\mathcal{M}$ (i.e., $\Gamma(\mathcal{M})$). We consider two different kinds of scores: *i*) a state similarity-score, $\sigma(\mathcal{M})$, which is derived from how much we need to (de)compress the behavior (in terms of number of states) to pass from one signature to another, and *ii*) a transition similarity-score, $\theta(\mathcal{M})$, which is derived from a comparison between matched labels. In a simple case, we can consider the mean value $\Gamma(\mathcal{M}) = (\sigma(\mathcal{M}) + \theta(\mathcal{M}))/2$, or we can apply more sophisticated weighted aggregation functions. A rather straightforward function is $\sigma(\mathcal{M}) = \min(|\mathsf{S}_{\mathcal{M}}|, |\mathsf{Q}_{\mathcal{M}}|)/\max(|\mathsf{S}_{\mathcal{M}}|, |\mathsf{Q}_{\mathcal{M}}|)$ and if we have $|\mathsf{S}_{\mathcal{M}}| = |\mathsf{Q}_{\mathcal{M}}|$, our match is perfect. But we can apply non-linear functions as well. The score $\theta(\mathcal{M})$ is computed by aggregating the individual syntactic similarity-scores ($ssim$), computed by the *Similarity Calculator* proposed in Section 4.5, obtained for each label match, and then averaging on the number of matched labels. In our example, $\theta(\mathcal{M}) = (ssim(label_{l_1}, label_{m_2}) + ssim(label_{l_2}, label_{m_4}) + ssim(label_{l_3}, label_{m_5}))/3$.

An Experiment with Stateful Services. As we discussed in Section 3.3.3, proper descriptions of the behavior of many implementations of stateless services are themeselves stateful. Therefore, we enrich our registered WSs with behavioral descriptions and use the query represented in Figure 4.12 against this registry. According to this stateful query, the ideal service matching the query first performs the Add-to-Basket operation one or more times, and then completes the required shipping processes, and finally charges the costumer for its purchase. Table 4.2 shows the results of this experiment: the transition similarity-score $\theta(\mathcal{M})$, the state similarity-score $\sigma(\mathcal{M})$, the global similarity-score $\Gamma(\mathcal{M})$, and the rank Rk of each service. These results match our expectations, since the behavior of S_6 is identical to the behavior of our query, and the behaviors of S_3, S_1, and S_2 are approximately close to the behavior of our query.

1. Let $\mid Q \mid$ be the cardinality of the query-automaton states, and $\mid S \mid$ be the cardinality of a service-automaton
2. If ($\mid Q \mid < \mid S \mid$) then

   %Find all possible sequences of merges for neighboring states in the service-automaton to reduce its state's cardinality to $\mid Q \mid$.

   2.1. For($i = \mid S \mid - \mid Q \mid;\ i \geqslant (|S| - |Q|)/2;\ i = i - 1$)
      2.1.1. Merge $i + 1$ neighboring states of the service to a single state.
      2.1.2. Merge $(\mid S \mid - \mid Q \mid) - i + 1$ neighboring states of the service to a single state.
      2.1.3. Adjust the transitions for the merged states.
      2.1.3. If (the new automaton is equal to the query-automaton) then consider it as a sub-graph epimorphism of the service-automaton.

3. If ($|Q| > |S|$) then
   Find all possible sub-graph epimorphism of the query-automaton to the service-automaton similar to 2.1

Algorithm 1: The sub-graph epimorphism algorithm for behavior specification matching.

```
q0 AddBasket() q1; q1 AddBasket() q1; q1 Shipping() q2; q2 Charge()
q0,[1.0]
```

Figure 4.12: A stateful query asking for a purchase online scenario including buying, shipping and charging.

4.7 QoS-aware Service Discovery

While the proposed search engine discovers services according to their functional properties, certainly non-functional properties, e.g., QoS parameters, also play an important role in the user's selection. Therefore, when many web services offer similar capabilities, it is necessary to also consider non-functional properties of services as selection criteria. While functional properties describe what a service can do, non-functional properties depict how well the service can satisfy its functional properties.

Besides the set of functional requirements (i.e., syntactical and behavioral matching of services) that we model as soft constraints, we can also encode any set of QoS requirements as soft constraints in order to assist users in service selection. In principle, we can compute QoS ranking as an extra criterion for search, but doing so may in turn impact the results of service discovery by giving higher ranking to some functionally-irrelevant services that have high QoS values. In order to avoid this potential problem, we use the lexicographic ordering, as

Table 4.2: The ranking of the top-ten matched WSs, based on the query represented in Figure 4.12.

Name	WSBS	θ	σ	Γ	Rk
S6	q0 AddToBasket(code:string) q1; q1 AddToBasket(code:string) q1; q1 Shipment(Address:String) q2 ; q2 Charge(AccountOnfo:string) q0	.88	1.0	.94	1
S3	q0 Login(IdInfo:String) q1; q1 AddToBasket(code:string) q2; q2 AddToBasket(code:string) q2; q2 Shipping(Address:String) q3 ; q3 Charging(AccountInfo:string) q1; q1 Logout() q0	.89	.75	.82	2
S1	q0 AddItem(code:string) q1; q1 AddItem(code:string) q1; q1 Shipping(Address:String) q2 ; q2 Charge(AccountInfo:string) q0	.56	1.0	.78	3
S2	q0 AddToBasket(code:string) q1; q1 AddToBasket(code:string) q1; q1 Shipment(Address:string) q0	.77	.67	.72	4
S7	q0 AddProduct(Prodcode:string) q1; q1 AddProduct(Prodcode:string) q1; q1 charge(AccountInfo:string) q0	.49	.67	.58	5
S4	q0 AddItemToBasket(code:string) q1; q1 AddItemToBasket(code:string) q1; q1 Charging(AccountInfo:string)	.44	.67	.56	6
S5	q0 Login(IdInfo:String) q1; q1 AddProduct(Prodcode:string) q2; q2 AddProduct(Prodcode:string) q2; q2 Shipment(Address:String) q3 ; q3 Payment(AccountInfo:string) q1; q1 Logout()	.32	.75	.54	7
TrackShipment	q0 ShippingInfo(Info:string) q0;	.67	.33	.50	8
ItemBasketService	q0 AddItemBasket(Itemcode:string) q1 ; q1 AddItemBasket(Itemcode:String) q1 ; q1 checkout(Itemcode:String) q0	.31	.67	.49	9
purchase	q0 sendPurchaseOrder(order:string) q1; q1 sendPurchaseOrder(order:string) q1; q1 shippingPT(Info:String) q0	.18	.67	.43	10

described below.

We compose all the constraints representing functional and non-functional requirements in the constraint assembler, but in a given lexicographic order, i.e., different criteria have different precedence. Assume that the user considers three QoS criteria for the service selection including: Availability, Reliability, and Response-Time. Then she also states a lexicographic order among these criteria: for example, the first preferred component is Reliability, the second one is Response-Time, and the last one is Availability. In order to combine functional requirements and QoS constraints into a single semiring for match-making of services, we define the lexicographic product of the semirings.

Our definition of lexicographic ordering derives from a lattice-based distance function we define in Definition 4.7.1. We use this function to exploit the structure of the complete lattice defined by $\langle A, + \rangle$. Below we explain why and how we perform this, first defining the needed lattice. A complete lattice is a partially ordered set in which all subsets have both a supremum ($\mathbf{1}$) and an infimum ($\mathbf{0}$).

4.7.1. Definition. [Lattice-based distance] The lattice-based distance is a function $dist : A \times A \to \mathbb{N}$ defined on a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, that returns the distance between two elements (absolute value) on the lattice defined by $\langle A, + \rangle$. In the following we provide two examples, depending if A is:

Finite and partially ordered. Given $a_1, a_2 \in A$, $dist(a_1, a_2)$ is the shortest path between a_1 and a_2 on the complete lattice defined by $\langle A, + \rangle$.

Infinite and totally ordered. Given $a_1, a_2 \in A$ and $a_1 <_S a_2$, $dist(a_1, a_2)$ is the distance obtained through a weak inverse of $\times$, i.e., $a_1 \div a_2$ [64].[6]

[6]Please refer to [64] for a formal definition of $\div$, which is outside the scope of this thesis.

Since, semirings can be defined over partial order sets, they can be represented as lattice graphs. The distance between two vertices in a lattice graph is the number of edges in a shortest path connecting them. To compute the *dist* function, we can use a distance matrix that is a square matrix containing the distances, taken pairwise, between the elements of a poset. For example, $dist(\{T\}, \{A, R, T\})$ in Figure 4.13 is 2.

We now define a lexicographic ordering which takes advantage of Definition 4.7.1. The goal is to use the distance of two elements from their least upper bound (obtained through +) as additional information to "stretch" the partial order into a "more" total order, to be considered in the lexicographic order.

4.7.2. Definition. [Lexicographic product of semirings] The lexicographic product of semirings $S_1 = \langle A_1, +_1, \times_1, \mathbf{0}_1, \mathbf{1}_1 \rangle$ and $S_2 = \langle A_2, +_2, \times_2, \mathbf{0}_2, \mathbf{1}_2 \rangle$, denoted as $S_1 \rhd S_2$, is the semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ where:

- The carrier of S is the Cartesian product of the carriers of S_1 and S_2, i.e., $A = A_1 \times A_2$.
- The selection operator + of S if given by:

$$\langle a_1, a_2 \rangle + \langle b_1, b_2 \rangle = \begin{cases} \langle a_1, a_2 \rangle & \text{if } dist(a_1, lub_1) < dist(b_1, lub_1) \\ \langle a_1, a_2 \rangle & \text{if } a_1 = b_1 \text{ and } dist(a_2, lub_2) < dist(b_2, lub_2) \\ undefined & otherwise \end{cases}$$

  where lub_1 is the least upper bound of a_1 and b_1 ($a_1 + b_1$), and lub_2 is the least upper bound of a_2 and b_2 ($a_2 + b_2$); $dist()$ is defined in Definition 4.7.1. We can even further refine the definition of our selection operator in Definition 4.7.2 by choosing the element that is more distant from their common greatest lower bound (*glb*) if two elements are equidistant from their *lub*. Note that two elements in A are always comparable with respect to *dist*, while they may not be comparable in $\langle A, + \rangle$.

- The composition operator $\times$ of S is given by:

$$\langle a_1, a_2 \rangle \times \langle b_1, b_2 \rangle = \langle a_1 \times_1 b_1, a_2 \times_2 b_2 \rangle$$

- $\mathbf{0} = \langle \mathbf{0}_1, \mathbf{0}_2 \rangle$ and $\mathbf{1} = \langle \mathbf{1}_1, \mathbf{1}_2 \rangle$.

The notion of distance that we have defined in the selection operator, is helpful in the case that two elements are not directly comparable. Assume several quality criteria of services are defined to rank WSs. These criteria would be partially ordered because all of their elements are not comparable. Figure 4.13 shows the lattice graph of these criteria including Availability (A), Reliability (R) and Throughput (T). For elements directly related by the partially order, for example

$\{A\} < \{A, R\}$, the choice is clear, so $\{A, R\}$ would be preferred in this comparison. However, we cannot compare some of these elements to each other, for instance $\{A\}$ and $\{R, T\}$. But using the *dist* function, $\{R, T\}$ is preferred, because the $dist(\{R, T\}, \{A, R, T\})$ is 1, which is smaller than $dist(\{A\}, \{A, R, T\})$ that is 2. In fact, we expect that $\{R, T\}$ would be better than $\{A\}$ because it is closer to the upper bound and satisfies more features.

We consider the lexicographic product operator on semirings to be right associative, i.e., $S_1 \rhd S_2 \rhd S_3 = S_1 \rhd (S_2 \rhd S_3)$. Accordingly, if A_1, A_2, and A_3 are the carriers of semirings S_1, S_2, and S_3, respectively, for simplicity, we skip ordering parentheses and denote the carrier of $S = S_1 \rhd S_2 \rhd S_3$ as $A = A_1 \times A_2 \times A_3$, instead of $A = A_1 \times (A_2 \times A_3)$, and denote the elements of A as $\langle a_1, a_2, a_3 \rangle$ instead of $\langle a_1, \langle a_2, a_3 \rangle \rangle$.

Let S_{Func}, S_{rel}, S_{ret}, and S_{avl} denote the semirings for the functional requirements and the QoS constraints for Reliability, Response-Time, and Availability, respectively. Then, $S = S_{Func} \rhd S_{rel} \rhd S_{ret} \rhd S_{avl}$ is the semiring that we use to find the best match for a query. The fact that S_{Func} appears as the leftmost operand in the lexicographic product that defines S ensures that we consider QoS properties of services in our ranking of candidates only if their functional similarity scores are the same.

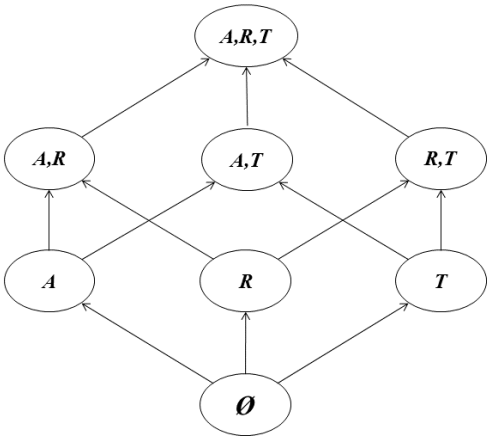


Figure 4.13: Lattice of subsets of $\{A, R, T\}$, partially ordered by "is subset of".

4.8 Conclusion

We have presented our novel tool for similarity-based discovery of web service that is able to rank the service descriptions in our registry, primarily focused on service behavior as well as QoS, in accordance with a similarity score matching each with the description of a service desired by a user. The formal framework behind the tool consists of SCA [15], which can represent different high-level stateful software services and queries. We can use SCA to formally reason on queries e.g., on operational similarity for SCA as we introduced in [15]. The tool is based on implementing approximate operational-similarity evaluation with constraints (see Section 4.6), allowing to quantitatively estimate the differences between two behaviors. Defining this problem as an SCSP makes it parametric with respect to the chosen similarity metric (i.e., a semiring), and allows using efficient AI techniques for solving it. To the best of our knowledge, our proposed web service search engine, BehSearch, is the only method that behaviorally matches services against queries according to the approximate operational-similarity evaluation. BehSearch also support non-functional requirements of services besides their functional constraints in a given lexicographic order.

The main intent for our approach has been to propose a formal framework and a tool with an approximate operational-similarity of behaviors at its heart, and not to directly compete against tools such as [133], that simply do not support behavior specification of services in their matching.

Chapter 5

Service Coordination and Composition

Parts of the material in this chapter is published in the following papers:

- Afsarmanesh, H., Sargolzaei, M., and Shadi, M., 2015. Semi-automated software service integration in virtual organisations. Enterprise Information Systems, 9(5-6), pp.528-555.
- Jongmans, S. S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. (2012, September). Automatic code generation for the orchestration of web services with Reo. In European Conference on Service-Oriented and Cloud Computing (pp. 1-16). Springer Berlin Heidelberg.
- Jongmans, S. S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. (2014). Orchestrating web services using Reo: from circuits and behaviours to automatically generated code. Service Oriented Computing and Applications, 8(4), 277-297.

A most important opportunity in collaborative networks that can benefit from the application of service-oriented architecture paradigm, is the composition of services. Although services composition has been heavily studied and discussed, several issues related to the coordination and automated execution of composed services still need to be addressed, especially when multiple service providers as VO members aim to collaborate with each other in order to create a new composite service.

In this chapter, we present our novel approach to compositional construction of web services that uses the Reo together with our constraint automata representation of web service behavior as the main "glue" ingredients. We apply Reo in our approach that is a graphical and exogenous coordination language based on channels. We propose a framework that, taking as input our proposed web service behavior specification representation constraint automata, the interface of services, and the description of their interaction in Reo, in an automated manner generates the necessary Java code to both compose and orchestrate a set

of services in practice. For each web service, we automatically generate a proxy that manages the communication between this service and the Reo circuit. These proxies in turn will execute their respective services.

5.1 Introduction

Services are encapsulated applications that are platform independent and employed to support rapid, low-cost, loosely-coupled compositions. In a *Service-Oriented Collaborative Network* (*SOCN*), services that are distributed over a network of organizations are desired to be composed according to the requirements defined by a service integrator. In principle, the composition of services requires additional efforts to impose coordination among the component services, e.g., using work flows, connectors, or glue code.

The holy grail of service oriented computing is to support proper and efficient reusability of software services and compose them in order to build new value added composite services, which are also seen as a kind of co-innovation in collaborative networks. Research on coordination models and languages plays a key role in this quest, as it facilitates flexible and formal ways to connect the component services and integrate them. In fact, the functional aspects are implemented and encapsulated in atomic services as the components, then the coordination process controls the way that those components shall communicate with each other to form the composite service. The coordination process in this context is particularly essential in order to avoid using ad-hoc composition approaches [98].

In SOA, coordination is partitioned into two fundamental paradigms that allow complex combinations of components: orchestration and choreography [129]. Orchestration is the most common approach that creates a central controller to implement the coordination among the involved components. Choreography is a collaborative endeavour, which does not rely on a central coordinator, rather on collaborative behavior of the services themselves. From the perspective of service composition also in collaborative networks, orchestration is a more flexible paradigm and has several advantages over choreography [86].

While traditional approaches use textual glue code to handle the coordination (e.g., [60], [111] and [173]), several coordination languages offer more visual approaches that can be realized using the so-called “channels” or “connectors”, in order to coordinate components in a composed system (e.g., [9], [102] and [147]). Connectors provide the glue code specifying the needed interaction protocol among the components or services, while describing the coordination concerns of a composed system independently from its implementation. In this way, they enable separating the coordination concerns from the computation concerns in a composite system, like the composed services [10]. We have therefore concluded that employing connector-based coordination models and languages in service-oriented systems, e.g., for our proposed SOCN, improves and simplifies describing

and managing the interaction protocols among the component services.

Although several coordination languages are proposed to orchestrate or to choreograph services, they mostly remain at the description level, without providing any kind of formal reasoning mechanisms or the practical tool to support the proposed notation for checking the compatibility of services [117]. Therefore, notwithstanding all the efforts spent on this topic, composition of services still is a challenging task. We propose to orchestrate composite services using Reo [9] which is a graphical and exogenous channel-based coordination language. Although there are already several rather theoretical studies on service composition using Reo (e.g., [115] and [116]), in our research we have extended those approaches through more practical perspectives.

In this chapter, we describe the design and development of a tool, which allows us to easily deploy Reo circuits for orchestrating real web services for our service composition approach. Our developed tool works as a specific wrapper, which: connects web services to Reo nodes, automatically translates the Reo circuits, and generates the executable orchestration of the Java code for the composed services. We also present here how to generate all the necessary Java code in an automated way, starting from the description of the orchestration (i.e. with a Reo circuit), the description of the web service interfaces (i.e. with WSDL files), and the behavioral specification of services (i.e. using constraint automata). For each web service, we generate a proxy application that acts as an intermediary relaying messages between the web service and the corresponding orchestrator circuit. In other words, this kind of proxies bridge the gap between the *web service world* and the *Reo world*. In our developed framework, all output code, which is necessary to manage the orchestration in practice, is generated automatically in a manner completely transparent to both client and service providers, whose programmers do not have to be concerned with this middleware at all. This work in turn also enhances Reo capabilities in service composition by implementation of service orchestration, where Reo already supports the modeling and even the verification of service composition [92]. Although the focus of our proposed tool and experiments is on web services, the same framework can be used to compose different kinds of service-oriented and component technologies, e.g., CORBA, WCF, and RPC, at the same time, through generating their different proxies and connecting them to the same Reo circuit.

The rest of this chapter is organized as follows: Section 5.2 describes the related work and further motivates this work with respect to the state of the art literature. The concept of coordination, as well as a tabular comparison between our choice of Reo and some other coordination languages is presented in Section 5.3. Section 5.4 briefly recalls the Reo coordination model as the necessary background notion, where we also present an example of a Reo connector for illustrative purposes. Sections 5.5 form the core of this chapter, since it discusses the details of the concepts behind, and the architecture of our Reo-based orchestration tool and shows how we have implemented it. In Section 5.6, we present

a case study of web service combination, whose code can be automatically generated with our tool, and in Section 5.7 we draw our final conclusions of this work.

5.2 Related work

Web service composition has received much interest for supporting collaboration among enterprise applications [168]. Many research projects have studied the composition of web services as the most promising choice to implement service oriented architecture and its strategic objectives. Some earlier approaches, such as SWORD project [134] give a simple composition mechanism for combining web services, without considering the complexity of coordination among the component services. SWORD uses an expert system to check whether a desired composite service can be realized using existing services or not. If so, it designs a plan for that service composition. However, SWORD does not benefit from the service-description standards (e.g., WSDL, SOAP, RDF, etc.), instead, it uses the Entity-Relationship (ER) model [161] to specify web services. In SWORD, each service is only represented by its inputs and outputs. In fact, it focuses more on the data integration, and no coordination issue has been addressed.

Several other research works on service composition rely on QoS concerns rather than an service interoperability and coordination, in order to optimize the quality of service composition. The eFlow project [37], WSQoSX [23], UPWSR [108] are some instances of this kind of research efforts. Due to the large number of existing works on service composition, we are not going to give an exhaustive survey, but we focus on several representative efforts that focus on coordination concerns in service composition.

Many languages have emerged and been proposed in academia and industry for coordination of composite web services, according to the choreography or orchestration principles. BPEL4WS [85], WS-CDL [68], BPML [164], YAWL [162], WSCI [17], $BPEL^{gold}$ [55], BPMN [44] and BPEL4Chor [44] are some examples of these languages. BPEL4WS (Business Process Execution Language for web services), or BPEL in short, is an XML-based language for web service composition through orchestration. There is also support in BPEL4WS for the specification of processes that involve operations provided by one or several web services. Furthermore, BPEL4WS draws upon concepts developed in the area of workflow management. When compared to languages supported by existing workflow systems and to related standards (e.g., WSCI), BPEL4WS is relatively more expressive. Since BPEL does not directly support choreography, BPEL4Chor has been proposed as a choreography-oriented version of BPEL4WS. $BPEL^{gold}$ is also proposed to support the representation of interaction models for collaborative business processes in BPEL [96]. Neither of these two standards, however, provide a full graphical notation, and thus they need to be combined with the

BPMN to fully support service interactions [97]. Additionally, a fundamental concept in software engineering, "separation of concerns", might be violated using combined BPMN and BPEL-based approaches for coordination of services [50]. In [50] the authors argue that a business expert is aware of the BPs in an organization, and he/she does not often know how to implement a service, while a programmer does not have enough knowledge about processes, he/she only implements services. It is therefore better to separate these two perspectives and concerns through different notations and standards for business process modeling and coordination. Following this strategy, the proposed approach similarly suggests using Reo as a graphical coordination language only for coordination concerns, as addressed in Section 5.4 of this chapter.

The web service Choreography Description Languages (WS-CDL) is a W3C candidate recommendation in the area of service coordination based on peer-to-peer collaborations of participants, which are realized through defining their observable behavior [154]. WS-CDL is an XML-based language that captures the interaction protocols of web services from a global perspective, i.e. all participating services are treated equally, unlike the orchestration-based language, such as BPEL and YAWL (Yet Another Workflow Language), where service interactions are described from the perspective of one single participant. Like WS-CDL, WSCI (web service Choreography Interface) focuses on the choreography of web services [17]. WSCI is an XML-based interface description language, which defines multi-party interaction scenarios through describing the flow of messages exchanged by a web service participating in choreographed interactions with other web services.

The Business Process Modeling Notation (BPMN) offers a rich set of graphical notations for control flow constructs and includes the notion of interacting processes where sequence flow (within an organization) and message flow (between organizations) are distinguished [44]. The works are done in [99], [103], and [123] indicate that BPMN, as the most prominent defacto standard to model business processes (BPs), does not target supporting the direct software development, and therefore lacks a formal declarative model defining precisely the logic behind the diagrams. Furthermore, there are several ambiguities inherent in BPMN diagrams, which also prohibit the direct development of equivalent web services for BPs [167], [26]. Thus, numerous formal proposals have been made for representing services using for example labeled transition systems, Petri nets, and Reo itself ([75], [25], [170], and [115]).

The composition of services pivots on coordination concerns. Coordination languages offer systematic middleware to support software composition in concurrent systems. In [10], a classification for coordination languages, is proposed from three perspectives: focus, locus, and modus of coordination. The locus of the coordination refers to where coordination activity takes place based on which the author classifies coordination models and languages as endogenous or exogenous. Endogenous languages provide primitives to support coordination within a computation module, while the provided primitives of exogenous languages support

coordination of entities from without [10]. In fact, exogenous languages separate coordination concerns from computation concerns, and the result of this separation makes composition of components easy to understand, and reusable in other applications. Talcott et al. [156] have considered a set of representative coordination features to study three most widely used exogenous coordination languages, i.e. Actors-Roles-Coordinators (ARC), Policy-based Russian Dolls (PBRD), and Reo. ARC divides principles of coordination into two groups including intra-role and inter-role coordination, and uses roles and coordinators, respectively as abstractions. The "coordinatees" in the ARC model are entities that interact with their environment via asynchronous message passing. In this model, coordination is achieved by time-space manipulations of messages, which are transparent to the coordinatees. In contrast to others, the RRD (Reflective Russian Dolls) uses a hierarchical structure to provide a general layered coordination model that focuses on more abstract specifications. PBRD is a restricted form of the RRD in which each layer (meta-object) is responsible for controlling the communication and the execution of objects in the layer below it. Reo is a graphical and exogenous coordination language based on channels, used for modeling the interaction protocols, referred to as exogenous interactions, among components [18]. Reo has been used in different applications especially in modeling of coordination in web service composition. Talcott et al. conclude that Reo is a mature language and it benefits from some formal semantics and tools for analysis, and finally Reo is closer to being a programming model.

Considering existing implementations, one can find service-oriented workflow research platforms, such as BliteC and Jolie, and even commercial offers such as IBM WebSphere, Bea WebLogic Integrator, Windows Workflow Foundation (WF), and Microsoft web services Support. Each of these systems provides a design tool and an execution engine for business processes in some workflow specification languages. For example, a part of Microsoft's BizTalk suite is the BizTalk Orchestration Engine. This engine implements XLANG, a predecessor of BPEL4WS. WF is a Microsoft technology that provides an API (Application Programming Interface), an in-process workflow engine, and a rehostable designer to implement long-running processes as workflows within .Net applications. BliteC [38] is a software tool that translates service orchestrations written in Blite, into readily executable Ws-BPEL programs. BliteC tries to solve some programming problems of WS-BPEL, i.e. lack of a formal semantics, and non-standardization of the deployment procedure. Jolie [118] is also a complementary approach for orchestration of services. It is a Java-based interpreter and engine, with a mathematical underlying model.

Comparing our work with the related work presented in this section, none of the XML-based languages in the proposed standards, e.g., BPEL4WS or WS-CDL, comes with tools for direct formal verification and model checking of programs or specifications in that language; therefore, verification of specifications in these languages requires a translation to a different level of abstraction, in con-

trast to other formal techniques, such as process algebra [25] and Petri nets [170]. Moreover, with Reo, a user is able to compose two orchestrators such that global synchronicity emerges from the synchronous behavior of the individual orchestrators [11] and [9]. This can be useful when different coordination protocols, designed for different services, need to be merged together in order to integrate all of them in the same single protocol. This advantage is granted by the formal definition of the join operator on two circuits. Furthermore, the Reo language allows direct declarative specifications of interaction, while with process algebra, one has to define a sequence of actions to achieve the same interaction. Besides the above features of the coordination languages, graphical modeling and visualization of data flow among the participating services make the comprehension of component interaction more intuitive [157]. A Reo IDE, called the Extensible Coordination Tools (ECT), comes with a strong animation framework for visualizing data-flows in Reo circuits. Moreover, thanks to the hide operation and consequently hierarchical composition in Reo, connectors in Reo circuits can be represented in an abstract level. Therefore, modeling the orchestration can be more comprehensive and better matched with its real configuration regarding the advantages of abstraction. Similarly, in WS-BPEL the complex processes can be encapsulated in components.

5.3 Coordination

Service oriented computations take advantage of the bundling of atomic services, as black-boxing of functional software artifacts, in order to yield new value added composite services as needed in collaborative networks. However, the availability of several services to work on a single composed service presents a new challenge to software technology, namely requiring coordination of the cooperation of a number of concurrent active services comprised into a composite service. The coordination paradigm offers a promising way to alleviate this problems and address some issues related to the development of composite services with complex interactions and communications among their constituent services.

In general, programming a distributed software system, like a composite service, can be divided into two orthogonal aspects of concerns and activities:

- An actual computation part, which comprises a number of processes involved in manipulating data and performed by a set of autonomous components or services.

- A coordination part, which is responsible for communication and cooperation between the realized processes by a kind of "glue code".

In fact, coordination can be used to distinguish the computational concerns from the communication concerns, not only allowing the separate development

but also the eventual amalgamation of these two major development phases [125]. In general, executable languages, such as Java can be used to coordinate the processes by putting code-your-own coordination logic in each process. But, this scheme is not suitable for service composition, because it mixes the computation code and the coordination code together, which raises the following two problems. First, this intermixing increases the complexity of programming. Second, if the interaction protocol is changed then the programs should be rewritten and therefore this approach is not a good candidate especially for dynamic service composition.

The concept of coordination is by no means limited to programming languages. There are many definitions for coordination, but one of the most widely accepted definitions in the area of software engineering is given by Carriero and Gelernter, where coordination is defined as the process of building programs by gluing active pieces together [66].

Services are essentially self-contained functional building blocks, which can be owned by third parties and are usually accessible only via published interfaces. Therefore, it is a natural choice to use an additional, external coordination mechanism to handle the interaction protocols among the constituent services from outside. The concept of "coordination from outside" the entities whose actions are coordinated is a notion that is called "exogenous coordination" [13]. Indeed, exogenous models of coordination offer the means to specify the demands of service interaction mechanism explicitly. Consequently, in endogenous coordination models, the primitives that control the coordination of a component with others can only settle inside of that component itself. It is increasingly becoming apparent that using endogenous coordination increases the complexity of the components, while exogenous coordination methods provide the means to specify explicitly the interaction between components or services in order to avoid ad hoc composition approaches [98]. While services implement only the functional aspects of an application, their coordination can be realized through so-called connectors, which describe and control the way the services communicate with each other. Indeed, connectors provide the needed glue code for specification of the interaction patterns in SOA-based systems.

Modeling of interaction protocols among components can be expressed as predefined connectors explicitly and independently from the connectors' implementation. Considering the paradigm of model-driven engineering, this alleviates the complexity of the applications and allows to use code generation for deriving the implementations. Furthermore, this level of abstraction for coordination models enables the use of formal verification and model checking. Therefore, we can conclude that employing connector-based coordination models and languages in service-oriented systems, e.g., our proposed SOCN, improves and simplifies describing and managing the interaction protocols among the component services.

In the literature, there are two main coordination paradigms to compose constituent services in order to make up a business process, including orchestration

or choreography languages [129]. In orchestration, the constituent services are under the control of a single endpoint central process. This process coordinates the execution of different operations on the services participating in the orchestra. An invoked service neither knows nor needs to know that it is involved in an orchestra and that it is playing a role in a business process definition. Choreography, in contrast, does not depend on a central coordinator. Each service that participates in a choreography has to know exactly when to become active and with whom to interoperate: it must be conscious of the business process, operations to execute, messages to exchange, as well as the timing of message exchanges. In fact, Choreography is typically defined as the interactions that occur between several services rather than a specific business process that a single party executes [154].

As Figure 5.1 shows, an orchestration defines an executable process involving message exchanges among different services, in which the sequences of message exchanges are controlled by an orchestrator. Choreography, on the other hand, proposes a protocol for peer-to-peer interactions among services, in order to guarantee the interoperability among these components. In other words, in orchestration the central control enforced by the conductor harmonizes the behavior of different actors in a distributed system. In choreography however, the distributed system behaves according to the rules individually complied by the actors to participate in a collaboration, without a centralized control [115]. WS-BPEL and WS-CDL are the most popular standards for orchestration and choreography, respectively. However, in real-world scenarios, corporate entities are sometimes unwilling to delegate control of their business processes to their integration partners. Therefore, this dissertation focuses on the orchestration paradigm, although Reo can be used to also describe choreographies [115].

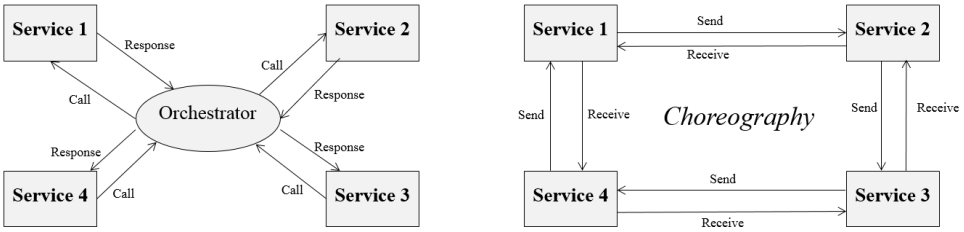


Figure 5.1: Service orchestration vs service choreography.

In order to justify the adoption of Reo as the base for our SOCN development, we have considered a large set of validation criteria. Sheng et al. [154] compare several coordination standards and languages for service composition defines the following six criteria. These are further extended by four more criteria supported by Reo, as mentioned later. We adopt/adapt these ten criteria in our SOCN development.

- Composability: The ability to assemble existing services into a new com-

posite service and model the interactions protocols among them.

- Role representation: The ability to demonstrate the role of a component service, which is needed to interact with the other components in the composite service.
- Complex structure support: The ability to model the complex structures, which indicates the execution logic and the ordering rule of operations executed during a composite service invocation.
- Adaptability: The ability to cope with business exceptions and process faults, which may occur within a composite service execution. It represents the ability to reverse the effect of some unsuccessful work in cases where exceptions occur.
- Compensability: The ability to neutralize the effect of unsuccessful actions during the execution of a composite service.
- Semantic support: The ability to support semantic representation of the services to improve automation and efficiency of service composition.

Four other criteria can be added to this comparison as described below.

- Visualization: The ability to visually represent dataflows among the component services.
- Formal verification: The ability to prove the correctness of coordination using the mathematical techniques and model checking.
- Abstraction support: The ability to encapsulate and represent the coordination elements in different abstract levels.
- Compositionally: The ability to compose individual coordinators (or coordination parts) in order to have a composed one, in which properties of program components, i.e. simpler connectors, are preserved when they are composed as more complex connectors.

Table 5.1 gives a detailed comparison in a tabular form between BPEL, WS-CDL, BPML, ebXML, OWL-S, and Reo, along the above ten major dimensions that characterize a coordination language. From this table, we can derive that Reo is the most-fit language that meets most of the needs for our purpose to compose the services. In fact, Reo supports all criteria that we have defined here for coordination. Reo as a domain specific coordination language provides full support for both “Composability” and ”Role representation” [13]. Moreover, the authors of [90] illustrate how Reo can enable “Adaptability” and “Compensability”. We can observe that only Reo and OWL-S provide semantic support for

Standards	WS-BPEL	WS-CDL	BPML	ebXML	Owl-S	Reo
Composability	√	√	√	√	√	√
Role representation	√	√		√		√
Complex structure	√	√	√	√	√	√
Adaptability	√	√	√	√		√
Compensability	√	√	√			√
Semantic support					√	√
Visualization						√
Formal verification						√
Compositionally						√
Abstraction	√					√

Table 5.1: Comparison of web services coordination languages.

services composition. Constraint automata, a semantic model for Reo, provide state-based representations of process workflows and enable their automation by means of translating constraint automata to Java code in ECT [89]. Furthermore, Reo benefits from a strong graphical tool and environment, i.e. ECT that enables "Visualization" and even animation of the circuits [88]. Separation of coordination from computation in Reo allows formal verification of interaction protocols [93], [12]. Such verifier and model-checker tools are also integrated in ECT. While one of the main advantages of Reo is that it supports compositional construction of connectors, i.e. "Compositionally" [120], it also support "Abstraction" (see Section 5.2). Finally, Reo potentially supports complex structures during a composite service invocation through following behavior specification of component services. Indeed, our proposed tool in this chapter, namely ProxCG, helps to fill the blind spot. ProxCG generates proxies for the involved services to automatically follow and execute the ordering rules of operations according to the web service behavior specification. Additionally, in [12] Arbab shows that a high-level protocol language, such as Reo, can have advantages with respect to performance as well.

Considering all results generated in this comparison, has convinced us to adopt Reo in our SOCN service composition approach for modeling of the interaction protocols among the component services, as well as for supporting the orchestration of their execution. Further on, as addressed in Section 5.5, we develop a new tool to extend Reo tools with the mechanisms needed for SOCN.

5.4 Reo in a Nutshell

In SOCN, we orchestrate our software services using the graphical language Reo [9]. Several (rather theoretical) studies such as [116], [115], [117] focus on service orchestration using Reo. Our tool is built upon the ideas presented in

these earlier works, and specially from a more practical perspective, we present tools that enable employing Reo for orchestrating real services, deployed and running on different servers. Reo has been explained in details in [9] and [10]. We briefly recall here the most relevant aspects of Reo for the sake of our service composition.

Reo is presented as a channel-based coordination language, facilitating compositional construction of circuits as communication mediums that coordinate interacting parties (i.e. software services in this work) each of them is built from a number of primitive simple connectors, so-called channels. The ability to compose connectors out of smaller channels is one of the strengths of Reo. It allows complex circuits to be expressed as a composition of simple channels. Every channel in Reo is able to impose a variety of relational constraints and behavioral policies including synchronization on the timing of dataflow, lossiness, buffering, and even the direction of dataflow.

A channel can communicate with others through its ports, which are called "ends". Each channel has exactly two ends, and each such end has exactly one of two types: source or sink. Channels accept data items through their source ends or offer data items through their sink ends. Note that, channels do not necessarily have both a source end and a sink end, i.e. they can have only two source ends or two sink ends. Table 5.2 shows six different channels and their behavioral description at the disposal of Reo users. These channels handle the dataflow between the components and, thus, impose interaction protocols among them such as synchronous vs. asynchronous communication, buffering, filtering, etc. More complex connectors called Reo circuits are built compositionally out of these primitive channels. Note that Reo supports an open-ended set of channels which can be used to construct circuits, enabling users to define their intended channels according to their specific requirements.

When channel ends are plugged together to build Reo circuits, a node is formed. In other words, a node is a fundamental concept of Reo representing a topological place where some channel ends coincide. Obviously, every channel end coincides on exactly one node. In such digraph view, each channel is presented as an edge of the Reo circuit. Each node may be of one out of three following types: source, sink and mixed. If all node's coinciding channel ends are source ends, the node is called "source node". Analogously, a node is called "sink node" if all its coinciding channels are sink ends. Finally, a node is known as "mixed node" if plugged to a combination of both source and sink ends. Figure 5.2 shows an example of the three kinds of nodes in Reo. We use the term "boundary nodes" to refer the source and sink nodes that form together an interface of a connector allowing interaction with its environment. This interface allows components or services to connect and communicate anonymously with each other by performing I/O operations (i.e. write operations on source nodes, and read operations on sink nodes) on the boundary nodes. Subsequently, each source node of the connector treats as a synchronous replicator, which atomically writes its data items to all

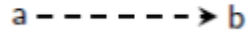
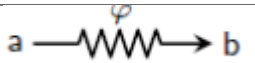
Channel	Name	behavior Description
	sync	Atomically fetches an item on its source end a and dispatches it on its sink end b.
	syncdrain	Atomically fetches (and loses) items on both of its source ends a and b.
	lossysync	Atomically fetches an item on its source end a and, non-deterministically, either dispatches it on its sink end b or loses it.
	filter(φ)	Atomically fetches an item on its source end a and dispatches it on its sink end b if this item satisfies the filter constraint φ; loses the item otherwise.
	fifo	Atomically fetches an item on its source end a and stores it in its buffer.
	fifo($\star$)	Atomically dispatches the item $\star$ on its sink end b and clears its buffer.

Table 5.2: Six primitive channels of Reo.

of its outgoing source ends. In a sink node, but one of the data items received from the sink ends is randomly selected, in order to be delivered to its connected component. Likewise the boundary nodes, mixed nodes do not interact with the environment, but instead, combine both behaviors by atomically receiving a data item from one of the connected sink ends and then coping it to its all connected source ends.

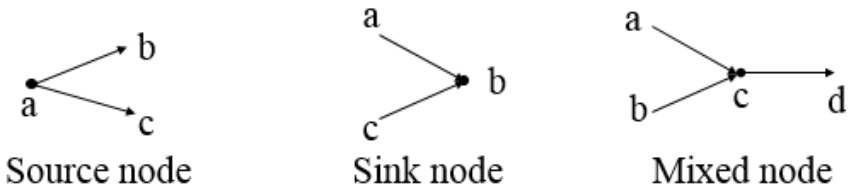


Figure 5.2: An example of Reo nodes.

In Figure 5.3, we present an example of Reo connectors that one can construct through composing the primitive channels defined in Table 5.2 using Reo nodes. Boundary nodes are represented as empty circles, and mixed nodes as filled circles in the circuits. Figure 5.3 shows a circuit, named *exclusive router*, composed of five Sync channels, two LossySync channels and one SyncDrain channel. This circuit imposes a specific routing path when parties can write and take data items to and from its boundary nodes (i.e. A, B and C): The exclusive router accepts data from its source node, i.e. node A and then flows the data to one of the sink nodes, i.e. B or C (but not both). In fact, this connector can consume data only if there is a write operation at the source node A, and there is at least one service attached to the sink nodes B or C, which performs a take/read operation. In the case that both B and C are able to accept data by a take operation, the decision of routing data to B or C is non-deterministically made by the mixed node I. It should be noticed that the node I can accept data only from one of its sink ends, and the latter's respective LossySync loses its data obtained from A, while the other LossySync passes its data. There are three possible dataflows for the exclusive router, which are shown by 2-coloring diagrams in Figure 5.4. Please note that the color of boxes in the figure, being white □, and black ■. Figure 5.4a shows how the data available on node A is forwarded to the node B but not to C, where the black boxes represent the data flow and empty boxes show the lost data. The case is shown in Figure 5.4b illustrates the routing the data from node A to the node C, i.e. the mirrored diagram w.r.t. Figure 5.4a. The last valid 2-coloring case of this Reo circuit represented in Figure 5.4c shows no dataflow.

In general, to model coordination among the services, a service integrator can derive the behavior of a circuit from the behavior of its channels and nodes. [84].

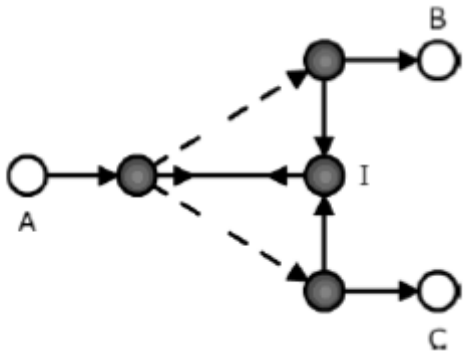
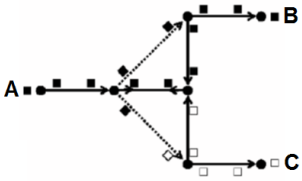
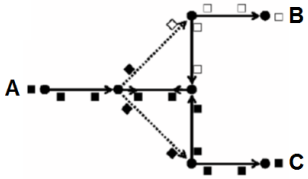


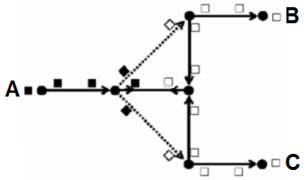
Figure 5.3: Exclusive router: an example of Reo circuits.



(a) Dataflow from A to B



(b) Dataflow from A to C



(c) No dataflow

Figure 5.4: 2-coloring examples for the exclusive router.

For brevity, we skip more detailed descriptions and examples, however, you can find more examples and details in [9] and [87].

5.4.1 Modeling Reo Circuits

There are various well-defined semantic models to formally describe the behavior of Reo connectors [82]. These models allow us to verify the correctness of Reo-based orchestration scenarios in composite services. For instance, Kokash et al. employ the mCRL2 toolkit, to verify the correctness of Reo circuits and ensure that the composed system behaves as intended [93]. They proposed an approach for mapping Reo connectors to the process algebra mCRL2 to have a formal verification for Reo circuits. In this work, the formal model of Reo channels, i.e. constraint automata, is presented to achieve two other purposes. First, we formally model the behavior of services in terms of constraint automata [18] (see Section 3.3.3). Second, we convert Reo circuits to constraint automata in order to automatically generate Java code for orchestrators, i.e. Reo circuits (see Section 5.5).

A Constraint Automaton (CA) is essentially a labeled transition system (LTS) to model Reo channels and circuits. In fact, constraint automata resemble classical finite state machines in the sense that they consist of finite sets of states

and transitions. When a CA is applied for modeling a Reo circuit, its states represent the internal configuration of the circuit, and its transitions describe the atomic coordination steps. The labels of CA represent two kinds of constraints: synchronization constraints and data constraints. A synchronization constraint represents a set of nodes which should be synchronized. Indeed, it specifies that through which nodes a data flow is simultaneously observed during a transition. Data constraint reveals which particular data items flow in a coordination step. Formally, we can define each transition of CA as a tuple of four elements: a source state, a synchronization constraint, a data constraint, and a target state. Figure 5.5 shows the constraint automata for some primitive Reo channels, which also represents the corresponding CA of the example shown in Figure 5.3 (exclusive router).

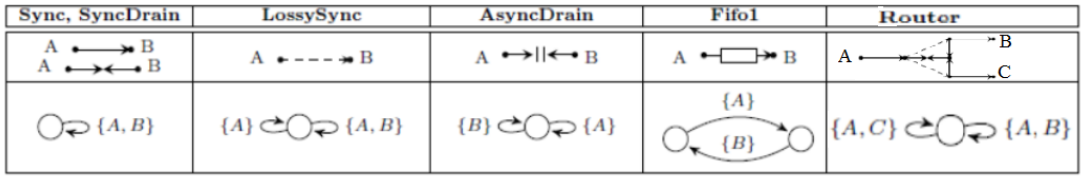


Figure 5.5: Constraint automata of common Reo channels and our example circuit.

5.4.2 Eclipse Coordination Tools

The Extensible Coordination Tools (ECT) [88] is a collection of open-source Eclipse plug-ins providing an strong visual programming environment as the default IDE for Reo. The ECT enables service integrators to design their intended Reo circuit diagrams using a drag-and-drop graphical editor. Furthermore, among other features, the designed Reo circuits can be animated in ECT in order to give more impressions about the dataflow in circuits. Several model-checkers are also integrated in ECT in order to verify the correctness properties of an interaction protocol using its CA. Most importantly, ECT can convert a Reo circuit to CA. Subsequently, ECT can also generate executable Java/C code from a CA as a single sequential thread. We deploy these two last features to automatically go from a circuit diagram (i.e. service orchestrator) to an executable code. However, the challenges related to web service orchestration with Reo are addressed in the next section.

5.5 Orchestrating SOCN web services with Reo

Conceptually, orchestrating web services (WSs) using Reo proceeds in three steps:

- Step (i): Design a Reo circuit as an orchestrator.
- Step (ii): Deploy and run this circuit.
- Step (iii): Connect the constituent web services to the circuit to build a composite service.

As mentioned earlier, the ECT perfectly supports the step (i): it allows to design Reo circuits using a drag-and-drop interface. Step (ii) can also be supported by a Reo-based tool called Circuit Code Generator (abbreviated "CircCG"), which is a Java code generator for Reo circuits implemented on top of the ECT [83]. This tool allows Reo users to deploy and run circuit diagrams as Java code, after having drawn and debugged them using the ECT.Suppose that a user of Reo has drawn a circuit diagram using ECT and wishes to generate Java code for it. Internally, ECT stores circuit diagrams as Xml documents, which serve as input to CircCG. Subsequently, CircCG computes the constraint automaton (CA) that models the behavior of the circuit. Finally, based on the CA just computed, CircCG generates a Java class. For more details, see [83] and [84].

Unfortunately, step (iii) involves far less straightforward activities which are challenging and not supported by an existing tool. How can we connect the services oblivious to Reo to the Java code generated in step (ii)? How can we exchange data between the services and the Reo part? In Sections 5.5.1 and 5.5.2, we address these questions with the theory behind and our implementation of tools for Reo-based service orchestration. These tools rely heavily on the concept of proxies. In Section 5.5.1, we explain why we need these proxies, what they look like, and how they work. Then, in Section 5.5.2, we detail the working of the tools themselves, which automatically generate Reo-based service orchestrators (including proxies). Postponing the details until Section 5.5.1, a proxy serves as an intermediary between a circuit and a service. Essentially, it relays data items from a Reo circuit to a service and vice versa, bridging the gap between them, addressing step (iii).

5.5.1 Proxies: Motivation and Working

Suppose one wishes to include a service that is deployed remotely in a composition, orchestrated by a Reo circuit. Ideally, we would simply send this service to the synchronization points of the boundary nodes to which it should connect. Subsequently, this service would directly perform the I/O operations on the synchronization points that it has received.

But unfortunately, the above approach does not work for a simple reason: virtually no existing service supports direct communication via the synchronization points of Reo circuits. In order to connect services oblivious to Reo and allow

them to exchange data between services and the Reo in SOCN, we use *proxies*. In this approach, one creates a proxy for each service in an orchestration. These proxies then act as intermediaries between a Reo circuit and the services in the orchestration. Essentially, proxies just relay data items from circuits to services and vice versa, bridging the technological gap that exists between them. Please note that to a service, a proxy looks just like any other client.

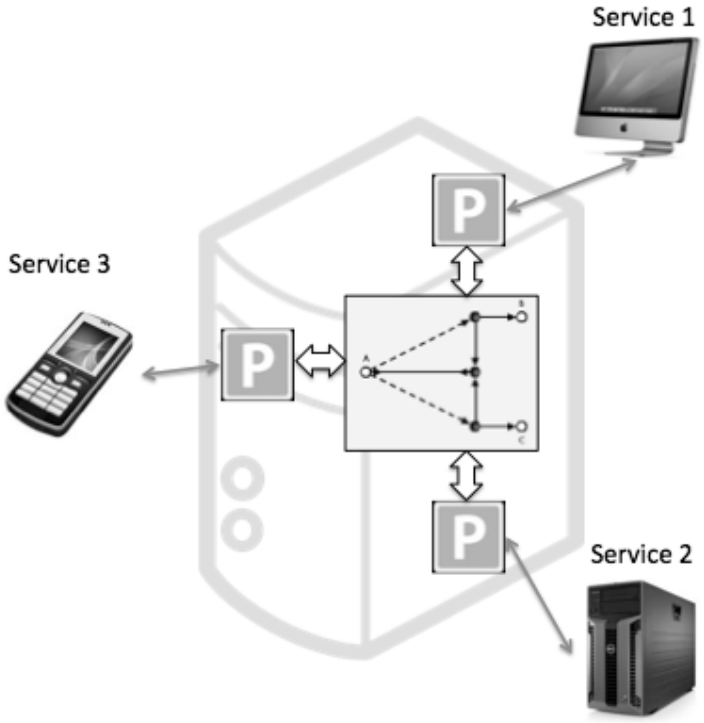


Figure 5.6: Architecture of a Reo-based orchestration scenario with service proxying.

Figure 5.6 shows what the architecture of a Reo-based service orchestration scenario with proxying can look like. This particular scenario involves three services, each of which running on a different machine. The fourth machine, in the center, runs the code generated for the circuit that orchestrates the three services (using CircCG) and one proxy for each of them. The services communicate directly only through their proxies, which relay messages to and from the circuit. As shown in Figure 5.6, we assume that the proxies of services always run on the same machine as the circuit. This has a practical reason: generally, one cannot deploy applications on the machines on which the independent services run.

Henceforth, we call the circuit that orchestrates the services in a Reo-based service orchestration scenario the *orchestrator circuit*. Then, an *orchestrator* consists of (i) an orchestrator circuit and (ii) one proxy for each of the services in the orchestration.

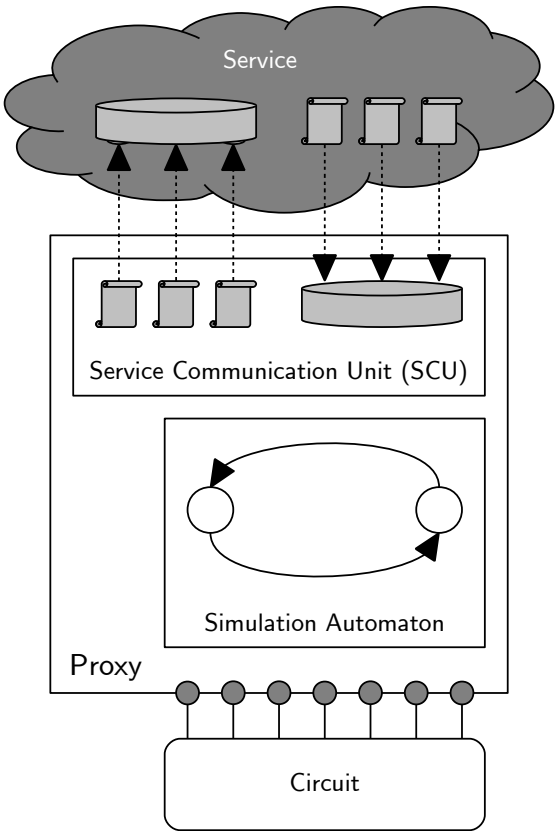


Figure 5.7: Architecture of a proxy.

To explain service proxying in more details, Figure 5.7 shows the architecture of a proxy P, together with a circuit C and a service S. Essentially, P consists of two *sides*: a *circuit side* and a *service side*. On the circuit side, P has access to a number of synchronization points. Thus, this side allows P to write and take data items directly to and from C. On the service side, P has access to the network infrastructure that connects P with S. Thus, this side allows P to directly send and receive messages to and from S. Indeed, P has two following tasks:

- Take data items from C on its circuit side. Encode these data items into messages that can be sent to S on its service side. Send these messages.
- Receive messages from S on its service side. Encode these messages into data items that can be written to C on its circuit side. Write these data items.

In the rest of this section, we elaborate on circuit sides and service sides of proxies, illustrating some of the concepts in terms of the following running example: a simple, yet *stateful*, service for incrementing numbers. This service, called *IncService*, exposes two operations to its clients, each of which takes an integer as input. Operation *SetInc* stores its input value in a state variable `i` (and returns

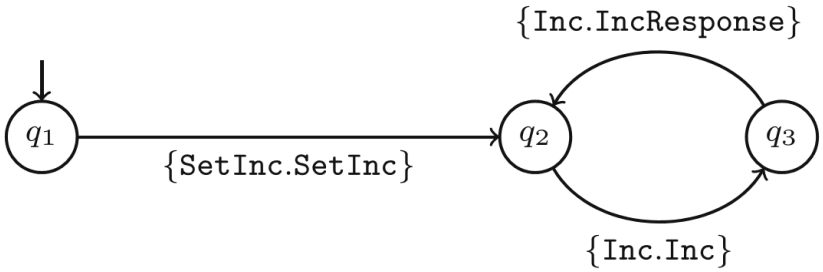


Figure 5.8: Simulation automaton of *IncService*.

nothing); operation *Inc* returns its input value plus `i`. For the former operation, *IncService* and its client exchange only one message, of type *SetIncRequest* (same name as the name of the operation); for the latter operation, they exchange two messages, of type *IncRequest* (same name as the name of the operation) and *IncResponse*. In every session, a client of *IncService* (i.e. a proxy in our examples) can invoke *SetInc* only once and must invoke it before any invocation of *Inc*.

Circuit Side

The circuit side of a proxy interacts with the orchestrator circuit via the latter's synchronization points. Every such synchronization point represents a message type whose instances the proxied service can exchange with its clients; the data items passing through these synchronization points constitute the payloads of actual instances. Note that a CA implementation of each circuit has a synchronization point for each boundary node occurring in one of its transitions [83]. For example, the proxy of *IncService* has access to a synchronization point for the *SetIncRequest* message; the payload of an actual instance of such a message has the form of a serialized data item taken from that synchronization point (e.g., "`1`").

Inside the circuit side of a proxy, an *executable* constraint automaton (CA) has direct access to (references to) the synchronization points of the orchestrator circuit. This means that, while running, this executable CA can directly perform write and take operations, as it sees fit. The box labeled "Simulation Automaton" in Figure 5.7 represents this executable CA.

Simulation automata simulate the behavior of proxied services. In particular, simulation automata simulate their *external view* [115]: every state of a simulation automaton represents an externally observable internal configuration of a service, while every transition represents the exchange of one or more messages by this service.[1] Figure 5.8 shows the simulation automaton—technically a CA without data constraints—for *IncService*. The "nodes" in the synchronization constraint have the syntax `<operationName>.<messageType>`. By letting sim-

[1] One can model stateless services with singleton automata.

ulation automata perform operations on synchronization points shared with the orchestrator circuit, a simulation automaton constrains which synchronization points an orchestrator circuit can *write to* or *take from*, since for that to happen, a simulation automaton always should perform a corresponding take or write operation. Furthermore, since data items exchanged at synchronization points correspond to messages, a simulation automaton effectively controls which messages at any point in time can be passed from the orchestrator circuit to the proxy (i.e. to the wb service) and vice versa.

Proxies require simulation automata for the following reason. First, observe that stateful services, with more than one internal configuration, may permit the exchange of messages of different types in different configurations (e.g., in q_1, in Figure 5.8, *IncService* permits a *SetIncRequest* message but not an *IncRequest* message). The correct functioning of proxies depends crucially on this information: proxies must know which types of messages their services can exchange in every instant to decide which synchronization points to allow interaction on. To illustrate this, suppose that the current state of *IncService* forbids invocations of *SetInc*. In that case, it makes no sense for its proxy to take data items from the synchronization point for *SetIncRequest* messages: the proxy may try to relay data items taken from this synchronization point, but it will certainly fail. After all, *IncService* does not permit *SetIncRequest* messages in its current state.

To prevent such faulty behavior from happening, proxies must know the current configuration of their services. Typically, however, a service encapsulates its state and generally, it provides no means to access it. By simulating the behavior of their services, proxies compensate for this lack of information: every time a proxy exchanges a message with its service, its simulation automaton makes a corresponding transition. In this way, a proxy can always derive which message exchanges its service permits, namely from its simulation automaton.

Service Side

The service side of a proxy contains a component that takes care of the actual network communication with its service. The box labeled "SCU" in Figure 5.7 represents the component of the proxy that does this, called *Service Communication Unit* (SCU). To explain it briefly, it works as follows.

- The SCU monitors the simulation automaton on the circuit side. If this automaton makes a transition, the SCU registers both the data items and the synchronization points involved. It then packs these data items (payloads) and synchronization points (message types) into some appropriate message format, e.g., through SOAP for WSs. Finally, it sends these messages over the network to the actual service.

- Concurrently, the SCU receives messages sent by the actual service. It unpacks these messages (e.g., removes headers) and writes their payload as

data items on the appropriate synchronization points. In doing so, the SCU forces a corresponding transition in the simulation automaton. Otherwise, this automaton and the actual service can diverge.

For exchanging Soap messages between proxies and services (required for implementing SCUs), we use the Dispatch api of the Jax-ws reference implementation[2].

5.5.2 Generating Proxies and Orchestrations

In Section 5.5.1, we explained why we need proxies, what they look like, and how they work. It seems unrealistic however, to ask of programmers to write proxies for every service they wish to include in a Reo-based service orchestration scenario. Therefore, we developed two tools, provisionally called *Proxy Code Generator* (abbreviated "ProxCG") and *Orchestration Code Generator* (abbreviated "OrchCG"), which automatically generate comprehensive Java code for individual proxies and complete service orchestration scenarios—we present these tools in this section. Since we implemented ProxCG and OrchCG for web services (WS), we talk explicitly about WSs instead of services in general. Note however, that the concepts behind our implementation apply equally well to the other kinds of services.

A Tool for Generating Proxies

Figure 5.9 shows the architecture of ProxCG. To generate a proxy for some WS, ProxCG requires two inputs: a WSDL document and a *WS behavior specification* (WSBS) that we addressed earlier in Section 3.3.3. The WSDL document specifies the syntax and technical details of the interface of the WS; the WSBS formally describes its externally observable behavior. In this work, we use CA without data constraints as simple WSBS. WSBS can be also represented in terms of XWSDL(See Chapter 3), but in general, one can use also other specification formats for this purpose, e.g., process algebra.

To explain in more details how ProxCG works, consider a WSDL document `s.wsdl` and its corresponding behavior specification, i.e. WSBS. ProxCG proceeds in three steps.

- Step 1: First, ProxCG parses `s.wsdl` using Axis2 [80]; the box labeled "WSDL Parser" in Figure 5.9 represents the component involved. Both ProxCG and the proxies it generates use Axis2, albeit in different ways: ProxCG uses Axis2 for parsing, while the generated proxies use Axis2 for exchanging SOAP messages. After parsing, based on the information obtained, e.g., the SOAP version that clients should use to send messages to the WS, ProxCG generates code for a service communication unit (SCU); the box labeled "SCU Code Generator" represents the component involved.

[2]http://jax-ws.java.net

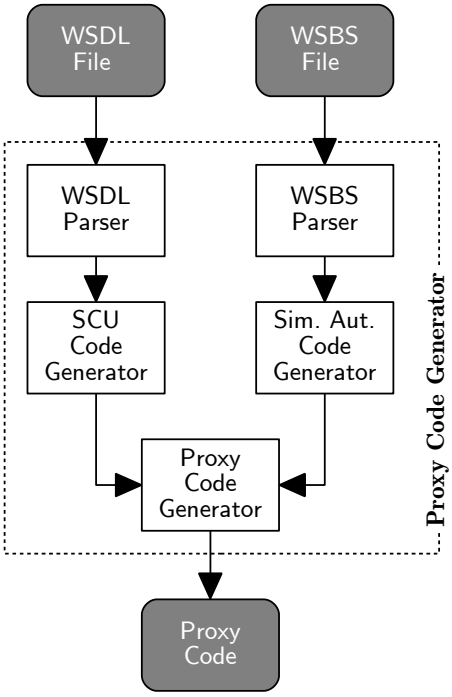


Figure 5.9: Architecture of ProxCG.

- Step 2: Next, ProxCG parses the WSBS; the box labeled "WSBS Parser" in Figure 5.9 represents the component involved. After parsing, ProxCG generates code for a simulation automaton; the box labeled "Sim. Aut. Code Generator" represents the component involved. Internally, among other generation activities, this component calls the functionality for translating CA to Java code provided by CircCG [83].

- Step 3: Finally, ProxCG combines the code generated for the SCU and the simulation automaton by adding glue code between them; the box labeled "Proxy Code Generator" in Figure 5.9 represents the component involved. More concretely, this step yields a Java class `P`; the box labeled "Proxy Code" represents this class. Instances of `P` run as proxies, encapsulating the constituent SCU and simulation automaton.

Instead of generating all the per-operation communication code statically at compile time, our tool generates code for a more general, reusable, dynamic invocation mechanism on top of the dispatch client of *JAX-WS RI*. This gives a bit more flexibility. The code generated for the dynamic invocation mechanism contains (a reference to) the original WSDL file; the initialization code parses that file and extracts all necessary information, e.g., the address to send messages to, types of parameters and results, etc., thus making that information quickly ac-

cessible when the dynamic invocation mechanism needs it later on for sending or receiving messages.

A Tool for Generating Orchestrations

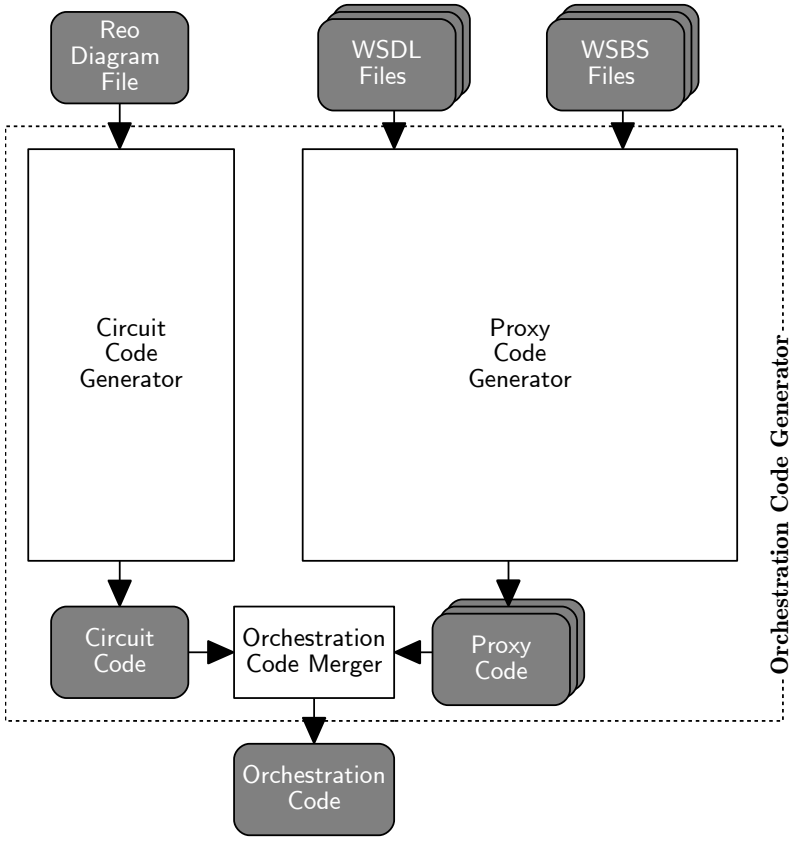


Figure 5.10: Architecture of OrchCG.

Figure 5.10 shows the architecture of OrchCG. Conceptually, OrchCG takes as input a Reo diagram containing the orchestrator circuit and a (WSDL, WSBS)-pair (or a XWSDL document) for each of the WSs in the orchestration of the scenario for which it is generating code. Using CircCG and ProxCG, the OrchCG then generates Java code for the orchestrator circuit and the proxies of the WSs. Below, we briefly explain how OrchCG combines the Java classes generated in this way; the box labeled "Orchestration Code Merger" in Figure 5.10 represents the components involved.

We start with some notation. Suppose that there is an XML document `c.xml` specifying the orchestrator circuit C. Furthermore, suppose a set of (WSDL, WSBS)-pairs $\widehat{S}$ representing the set S of those WSs that C orchestrates. Let `C` denote the Java class generated by CircCG on the input `c.xml` (i.e. the box

labeled "Circuit Code"); let P denote the Java class generated by ProxCG on input of some $\langle$s.wsdl, s.wsbs$\rangle \in \widehat{S}$ (i.e. the box labeled " Proxy Code").

Essentially, the component represented by the box labeled "Orchestration Code Merger" glues together C with the P class for every $\widehat{s} \in \widehat{S}$. This gluing yields a new class O, whose instances orchestrate the WSs in the set S as desired. More precisely, the glue code in O carries out the following activities.

- It creates a synchronization point for each boundary node of C.
- It creates an instance c of class C generated by CircCG on input of c.xml. Moreover, it passes the synchronization points created in the previous step to c. These synchronization points constitute the interface through which proxies communicate with c.
- It creates an instance p of class P generated by ProxCG on input of $\widehat{s} = \langle$s.wsdl, s.wsbs$\rangle$ for every $\widehat{s} \in \widehat{S}$. Importantly, it also passes to this instance the *appropriate* synchronization points created in step (1). The sharing of synchronization points between c and p establishes the link between the orchestrator circuit and a WS via a proxy.

The scenario displayed in Figure 5.11 involves two WSs: IncService and CalcService. The latter, has only one operation, *CalcPrimeFactors*, which decomposes natural numbers into prime factors. In every invocation, it exchanges two messages: an input message of type *CalcPrimeFactorsSoapIn*, which transports a number, and an output message of type *CalcPrimeFactorsSoapOut*, which transports a list of primes. For instance, *CalcService* responds to the *CalcPrimeFactorsSoapIn* message "12" with the *CalcPrimeFactorsSoapOut* message "2 2 3".

Figure 5.11 gives an idea of how users of Reo can use ProxCG and OrchCG in practice. The figure shows a screen shot of the drag-and-drop drawing interface of the ECT (right panel) and the GUI version of ProxCG and OrchCG (left panel).

Suppose that we want to generate just a proxy for *IncService*. To do this, we can fill in the form on the left panel (as done in the figure) and click the "Generate Proxy" button. The tools will then generate all the proxy code and, under the settings as displayed in Figure 5.11, put it in a new project in the current workspace (different settings write the code to a user-defined location on the local file system). Alternatively, we can click the "Draw" button to postpone generating code and, instead, draw a representation of *IncService* on the canvas on the right panel (more precisely: the box in the top–right corner in Figure 5.11). By doing so, we can design a complete Reo-based service orchestration scenario, including the orchestrator circuit and all the WSs in the orchestration, as in Figure 5.11.

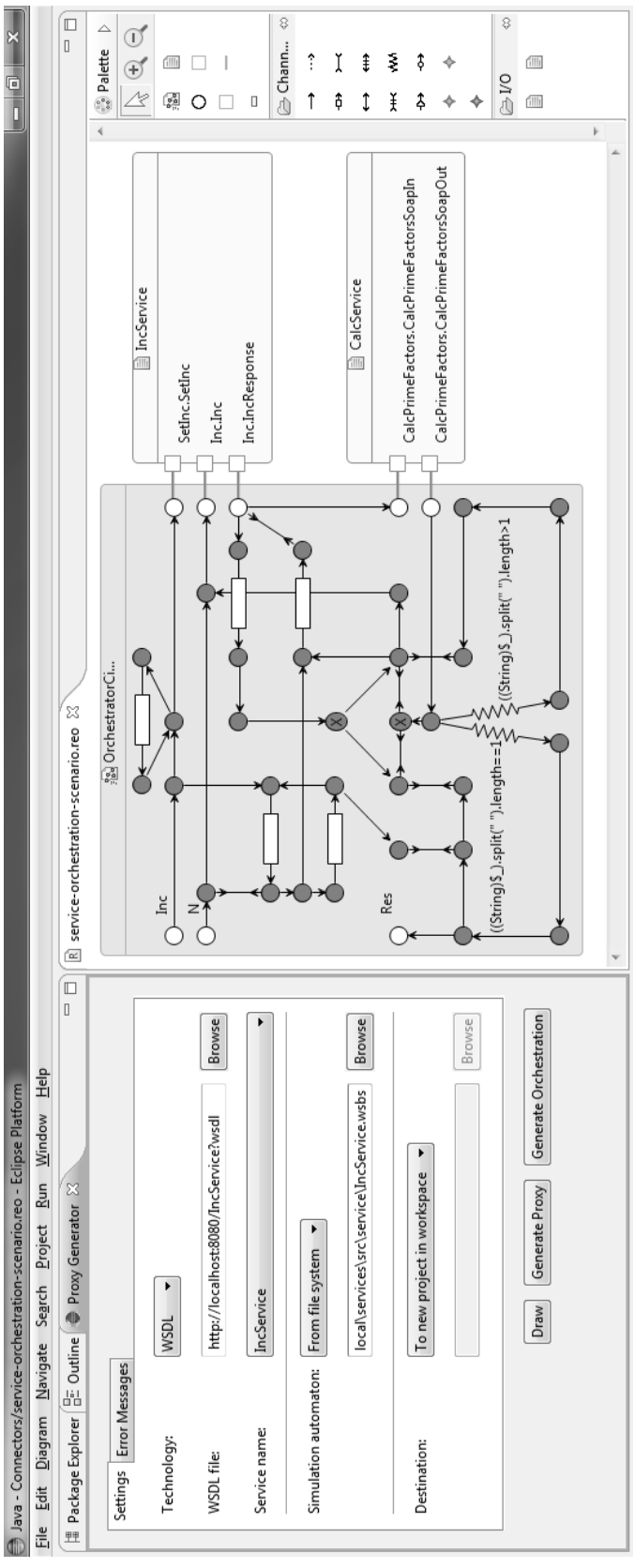


Figure 5.11: Impression of the ECT (right panel) + the GUI version of ProxCG and OrchCG (left panel).

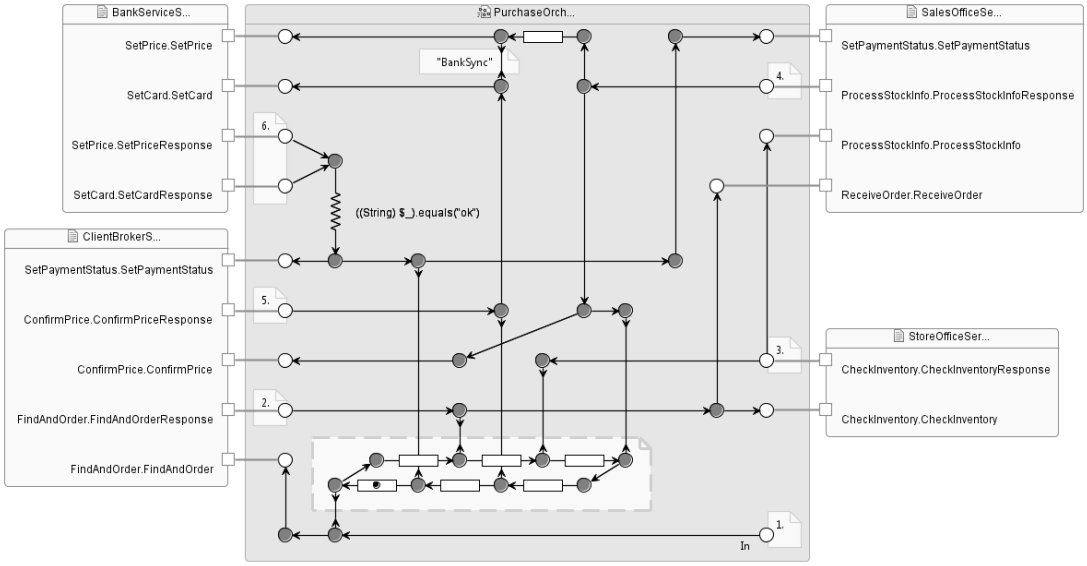


Figure 5.12: The sequential coordination of four WSs represented as a Reo circuit: the numbers, on comment notes, represent the ordering of the exchanged messages.

To generate all the necessary Java code for deploying and running the entire orchestration, we can click the "Generate Orchestration" button on the left panel in Figure 5.11. The resulting collection of classes contains one main class, `O`, for accessing the orchestration, which programmers can use in their Java code as any other class. As such, the WSs involved, as well as the orchestration circuit or the underlying Reo technology, remain completely transparent to the programmer. Note that one can make the functionality of this application available as a WS by encapsulating `O` in a new WS.

5.6 Case Study

In this section, we present a nontrivial example to familiarize the reader with the service orchestration scenario.

Figure 5.12 shows the complete representation of the circuit for the orchestration of four online sales-related WSs. The image has been created with our extension of ECT tool. With the aid of CircCG, we generated the *Purchase* circuit describing the interaction among these four WSs named: *ClientBroker*, *StoreOffice*, *SalesOffice*, and *Bank*. This scenario implements the classical purchase-online example. The *ClientBroker* WS takes care of interfacing the real client to the other WSs, which deal with: the information about the store (i.e. the *StoreOffice* WS), the procedure to prepare the invoice (i.e. the *SalesOffice* WS), and the effective payment management (i.e. the *Bank* WS). The complete high-level

behavior of these WSs is described below:

ClientBroker: The *ClientBroker* interfaces the real client with the other WSs:

1. It receives information about the object to order (issued by the real client), such as a product ID (e.g., #3, corresponding to a pair of shoes) together with a string of complementary data (e.g., "color red"), as an input message of type `FindAndOrder` (of operation `FindAndOrder`).
2. It finds a corresponding product and issues an order for this product (represented by an order ID) as an output message of type `FindAndOrderResponse` (of operation `FindAndOrder`).

   Both the *SalesOffice* and the *StoreOffice* need to receive a copy of this message.
3. It receives the price of the order (issued by the *SalesOffice*), as an input message of type `ConfirmPrice` (of operation `ConfirmPrice`).
4. It issues the credit card information of the real client, as an output message of type `ConfirmPriceResponse` (of operation `ConfirmPrice`).

   The *Bank* needs to receive a copy of this message to proceed with the payment.
5. It receives the confirmation of the payment (issued by the *Bank*), as an input message of type `SetPaymentStatus` (of operation `SetPaymentStatus`).

StoreOffice: The *StoreOffice* checks the availability of the order request:

1. It receives an order ID (issued by the *ClientBroker*), representing the order for some product, as an input message of type `CheckInventory` (of operation `CheckInventory`).
2. It issues a confirmation that the object is in the store or not, as an output message of type `CheckInventoryResponse` (of operation `CheckInventory`).

   The *SalesOffice* needs to receive a copy of this message for further processing.

SalesOffice: The *SalesOffice* computes the final price and sends the corresponding invoice information:

1. It receives an order ID (issued by the *ClientBroker*), representing the order for some product, as an input message of type `ReceiveOrder` (of operation `ReceiveOrder`).

2. It receives the confirmation that the product is in the store or not (issued by the *StoreOffice*), with possible further pricing information, as an input message of type `ProcessStockInfo` (of operation `ProcessStockInfo`.
3. It computes the final price and issues it together with the account number of the company, as an output message of type `ProcessStockInfoResponse` (of operation `ProcessStockInfo`).

   Both the *ClientBroker* and the *Bank* need to receive a copy of this message to proceed with the transaction.
4. It receives the confirmation of the payment (issued by the *Bank*), as an input message of type `SetPaymentStatus` (of operation `SetPaymentStatus`).

Bank: The *Bank* manages the payment, issued by the *ClientBroker*, according to the price information issued by the *SalesOffice*:

1. It synchronously (thanks to the syncdrain channel called *BankSync* in Figure 5.12) receives the card info (from the *Client*) and the price of the transaction (from the *SalesOffice*), as input message of type `SetPrice` (of operation `SetPrice`) and `SetCard` (of operation `SetCard`).
2. While operations `SetPrice` and `SetCard` are invoked synchronously, their responses occur asynchronously. If `SetPrice` responds before `SetCard`, the *Bank* issues a payment status saying that the transaction is in progress, as an output message of type `SetPriceResponse`. Otherwise, if `SetPrice` responds after `SetCard`, the *Bank* issues a payment status saying that the transaction completed. Something similar happens if `SetCard` responds before or after `SetPrice`, as output messages of type `SetCardResponse`.

   The filter channel ensures that the *Client* and the *SalesOffice* receive only the final confirmation of the payment.

From this description, we can generate the corresponding CA as described in Section 5.5.1. By easily adapting the tool in [39], we will be able to directly translate Sequence Diagrams into CA.

The dashed rectangle in Figure 5.12 contains the constituents of a Sequencer of the messages, i.e. a Reo subcircuit that enforces the correct ordering of the messages exchanged among the WSs. Consequently, the interaction of the WSs is sequential: the sequence consists of six steps, whose ordering is shown in Figure 5.12 with ordinal numbers, written on comment notes. The presence of the Sequencer in Figure 5.12 may seem redundant as the WSs themselves already impose an ordering on their interactions. However, the sequencing of the messages *is* also part of the protocol among the WSs and should therefore be part of the

protocol specification—regardless of what the involved WSs do. Therefore, it is a best practice for the orchestration designer to include all the features that the interaction requires, without exclusively relying on the service programmers. In fact, this reasoning led to the development of the "safe" code.

We programmed and deployed the WSs on a server machine, and afterwards we automatically generated their proxies with the help of OrchCG.

5.7 Conclusion

Composing business services to create new value added services in collaborative networks requires standards to model the interactions among the component services. Thus, service coordination constitutes one of the most challenging aspects of this service composition. While orchestration and choreography are two alternative approaches to handle the coordination concerns of the composition, for our purposes we advocate the use of orchestration through some extensions for the Reo language. With Reo and our proposed tools extending some features in this language, web services can be dynamically chained, and dynamically integrated as a composite service in virtual organizations.

In this chapter, we have shown how to automatically generate an orchestration framework for web services. The Reo circuits as the orchestrator are automatically translated into Java code, and used to compose the services in a way transparent to the client and all its involved component services. The input of our generated tool consists of the externally observable behavior of each service in terms of constraint automata, the WSDL description file of each component service, and the specification of their orchestration as a Reo circuit. From all this information, it is then possible to automate the Java code generation process from a Reo circuit and a proxy for each web service. This proxy component is in charge of managing the communication between the technology behind the web services and the Reo environment.

In general, our approach is complete for any Reo circuit with a CA semantics and any web service with a machine-processable interface definition and behavioral specification. The machine processable interface definition for web services supported by our current tools is WSDL, which is then enriched by behavioral specification of the services (see also Section 3.3.3). For further generalizing, please note that our framework is not limited to WSs, but it works also for other kinds of protocol parties, e.g., legacy service components. This makes our approach suitable for composing services implemented using different technologies and standards.

Chapter 6

Validation and Evaluation

Parts of the research results presented in this chapter are previously published in the following papers and technical reports:

- Sargolzaei, M., Shafahi, M., and Afsarmanesh, H. (2017, April). Service-enhanced product: from Specification to Composition. Submitted to the Journal of Intelligent Industrial Systems.

- Afsarmanesh, H., Shafahi, M., and Sargolzaei, M. (2015, February). On service-enhanced product recommendation guiding users through complex product specification. In Computing and Communications Technologies (ICCCT), 2015 International Conference on (pp. 43-48). IEEE.

- Shafahi, M., Afsarmanesh, H., and Sargolzaei, M. (2014, October). A coopetition space for complex product specification. In Working Conference on Virtual Enterprises (pp. 83-97). Springer Berlin Heidelberg.

- Afsarmanesh, H., Shafahi, M., Unal-Karakas, O., and Sargolzaei, M. (2013). D4.1 - Design report on approach and mechanism for effective customized complex product specification.

- Afsarmanesh, H and Sargolzaei, M and Shafahi, M (2014). D4.3 Report on dynamically customizable services enhancing complex products.

- Sargolzaei, M., Afsarmanesh, and H., Shafahi, M. (2014). D4.4 Prototype of the system for enhanced services recommendation.

- Camarinha-Matos, Luis M., Macedo, Patricia, Sargolzaei, M and Afsarmanesh, H (2013). D2.4 Mechanisms for defining composed services to support collaboration.

6.1 Introduction

This chapter addresses where and how the thesis and its results are evaluated and validated. First, we compare the features and appropriateness of the components developed in this dissertation, against a number of other competitor components and systems, for which the results are provided in this section. Additionally, the behavior-based service discovery component of our framework, BehSearch, is evaluated in this chapter. This evaluation is divided into two parts. First, the performance and scalability of the search engine, which is improved by offering a distributed model, is evaluated. Then the correctness of the results produced by the tool is examined through measuring, its information retrieval metrics. For this purpose, we gauge our results according to two most frequent and basic metrics in information retrieval context, i.e., the precision and recall. Although our tool is not directly comparable with related work in this area because they do not support behavior specification in their matching, we represent a short comparison to the extent possible with such tools. The other goal of this chapter is to demonstrate how we propose our results to be applied in industrial products.

The remaining sections of this chapter are structured as follows. First, we briefly introduce a validation and evaluation methods, which we then apply to our results in Section 6.2. In Section 6.3, we present a feature analysis method as a validation mechanism to compare developed tools against some related works. In Section 6.4 we present evaluation of our approach through formal experiments. In Subsections 6.4.1 and 6.4.2, we focus on the evaluation concerns, including performance and results correctness. Section 6.5 briefly recalls our real application case, and the proof of concept that we have developed for the GloNet project. Finally in Section 6.6, we provide some concluding remarks.

6.2 Validation and evaluation methods

In scientific research, it is important to evaluate and validate the results, as well as the proposed applied approaches to achieve those results. There are several validation techniques for software systems. Among different techniques that fit more appropriately to our purpose, we apply the approach suggested by Pfleeger [131]. This approach suggests a number of techniques among which the following three techniques are applicable and chosen for our validation purposes:

- **Feature Analysis (addressed in Section 6.3) -** *This is suitable to rate and rank different features of a developed system, e.g., for its novelty, complying to certain standards, or against competitor research results. This technique is also used to validate our findings in scientific communities. A feature analysis can be represented as a key indicator assessment that compares some competitor systems according to several indicators/criteria. We*

present a key indicator assessment for each of the developed tools in this dissertation, consisting of: XWSDL, BehSearch and ProxCG, as respectively proposed for service specification, discovery and composition.

- **Formal Experiment (addressed in Section 6.4) -** *This technique is mostly applied to evaluate metrics related to effectiveness and accuracy of results produced by algorithms. In this work, a formal experiment is used to evaluate the performance and accuracy of our discovery tool, i.e., BehSearch. We have proposed a decentralized model for BehSearch, and calculate speed-up rate for it. Moreover, we measure the recall/precision curve of the tool for estimation of its accuracy.*

- **Case Study (addressed in Section 6.5) -** *This is suitable for showing a holistic picture of applying the approach and results related to the developed system. This technique suggests that according to what we as the evaluator aim to examine, an application case shall be defined to test the functionalities and usage of the results. In this direction, our experience in the development of Product Service Specification (PSS) sub-system of the GloNet project is reported at the end of this chapter. We discuss the advantages that can be gained through applying our methods in GloNet. GloNet as our application case involves various collaborative networks (including VBEs and goal-oriented VOs) to facilitate the needed infrastructure and tools for networks of SMEs in the context of complex products, applied to solar power plants industry. Some major parts of our approaches especially for service specification and discovery introduced in Chapters 3 and 4, are validated through the real cases in GloNet.*

6.3 Evaluation through Feature Analysis

In this section, we provide three key indicator assessments for our component development validations.

6.3.1 Assessment of XWSDL

As mentioned in Chapter 3, we have proposed a new model and standard to specify services, named the XWSDL. We sought to validate XWSDL using some key indicators to compare with widely used standards/models. First, we briefly introduce these standards/models as our competitors in our validation comparison, as also summarized in Figure 6.1.

- **WSDL** [41] describes all syntactical properties of a web service, including a set of operations' signatures and a set of network endpoints or ports (URIs) at which these operations can be invoked. WSDL is the most adopted

standard for web service description. However, it is unable to explain all the functionalities of a web service, i.e., its semantics and behavioral properties. According to the Model Driven Architecture (MDA), a WSDL metamodel is introduced and then extended to support a number of extensions to WSDL.

- **OWL-S** (Web Ontology Language for Web Services) [100] is built on top of the OWL language, in order to specify semantics of web services and consequently help in the discovery of web services. OWL-S describes "individual" services, together with the set of their "property assertions" relating individual services semantically to each other [4]. OWL-S does not provide a concise and formal description of behavioral and non-functional properties of services, however it offers an unbounded list of service parameters that can contain any type of information.

- **WSDL-S** (WSDL Semantics) [6] proposes an extension of WSDL by semantics annotations. Unlike WSDL, WSDL-S can annotate the information provided in WSDL using different semantic languages, such as RDF and OWL. The elements of WSDL-S, which are added to the standard WSDL document, are modelReference, category, precondition and effect. These provide a rich description for the web services' semantics.

- **SAWSDL** (Semantic Annotations for WSDL) [95] is also defined as an extension of WSDL to describe the semantics of its elements through providing the mechanisms to bind ontology concepts to semantic annotations of WSDL. A metamodel presented for SAWSDL supports Model Driven Architecture (MDA) approach. This metamodel is independent of the SAWSDL language, but helps software developers to understand and work with this language.

- **Session Type** [46] is appeared as a newly emerged issue in the world of web services that allows modeling interactions and reasoning over communicating processes. Thus, a session type can be applied as a formal model to describe the user view of an interaction, i.e., the behavior of services [144] and [47]. This formalism can also describe the signature of services.

- **Q-WSDL**(QoS-enabled WSDL) [43] is a lightweight WSDL extension to specify the QoS characteristics of web services. The extension is carried out as a metamodel transformation, according to the Model Driven Architecture (MDA) paradigm.

- **XWSDL** as our proposed extension of WSDL, which is designed to support collaboration in networks of service providing organization, supports all aspects needed to specify services including syntax, semantics, behavior, QoS and other non-functional properties of services (e.g., cost). We have also

introduced the XWSDL meta-model, as an extension of WSDL metamodel, to support the Model Driven Architecture (MDA) paradigm.

Figure 6.1 shows a summary of our key indicator assessments for validation of XWSDL and the above competitors according to the following criteria: Syntactical description support, Semantics annotation support, behavioral description support, QoS support, Other Non-functional properties support, Having Graphical User Interface, and MDA-based approach.

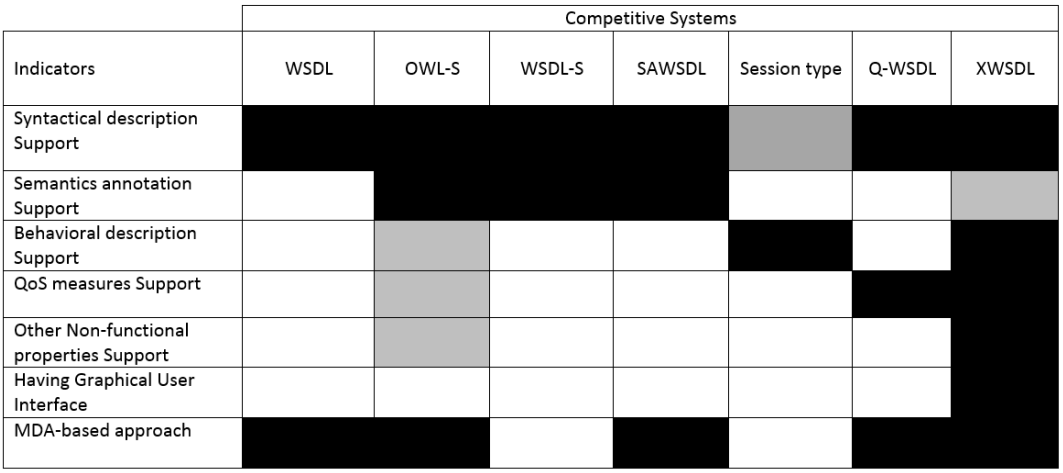


Figure 6.1: Comparison of XWSDL with similar models/standards.

Note that a feature of an approach is shown as a "black" box when it is satisfied well and it can provide all aspects of the criterion. If an approach partially satisfies the required criterion, its corresponding feature is represented as a "gray" box. Finally, if an approach does not provide what the criterion requires, it is exhibited as a "white" box for that criterion.

6.3.2 Assessment of BehSearch

In this section, we compare our discovery tool, BehSearch, against five other related works based on the following indicators: Signature Matching, Semantics Search, behavioral Matching, QoS-aware Selection, and Decentralized Model Support. Note that a strong signature matching should match: similar operations, similar input/output messages, as well as their data-types.

We first briefly introduce several related discovery tools as the competitor systems below.

- **WSXplorer** (Web Service Explorer) [71] presents a method to retrieve desired web-service operations from a given textual description. The method uses the concept of tree edit distance to match similar operation names.

Meanwhile, some algorithms are proposed for measuring and grouping similar operations, based on some other features. The proposed WSXplorer algorithm suggested for measuring and grouping similar operations catches not only their structures but also their semantic information.

- **Woogle** [51] as a search engine for web services, supports similarity search for web services. The similarity search supplements keyword search for web services operations. Moreover, Woogle search engine goes beyond keyword-search by exploring the semantics of web-service operations.

- **Di Modica+** [49] proposes an architectural model for the discovery of services based on a hybrid P2P approach. Di Modica+ partitions the P2P network into several semantic groups of peers, where each group is associated to answer queries of a specific applicative domain. In fact, this approach aims for semantic-based clusterization of nodes that provide services.

- **URBE** [133] offers a recursive pairwise comparison among data types, messages, and operations of web services. Considering the similar tools (e.g., WSXplorer), URBE supports more efficient semantic search and gives better performance with respect to both precision and recall.

- **VU+** [165] provides a semantic description model for the QoS properties of web services. It also performs a QoS enabled discovery and ranking of web services based on their QoS compliance. The framework could be used either as a centralized discovery component or as a decentralized repository system.

- **BehSearch** [144] as our proposed approach, presents a similarity-based method for discovery of web services that is able to rank service descriptions in our registry, according to their similarity with the description of a service desired by a user. The applied approach is primarily focused on service behavior as well as QoS.

Figure 6.1 summarizes our comparison between BehSearch and the above competitor systems according to the mentioned criteria.

6.3.3 Assessment of ProxCG

In this section we compare our tool for service composition, ProxCG and eight of its competitors considering several criteria, defined below. First, we describe these competitors: eFlow, Intalio BPMS, Self-Serv, SHOP2, Sword, XL, TQoS, and YAWL.

- **eFlow** [37] is one of the first frameworks implemented for specifying, executing, and monitoring composite services. eFlow uses visual notation and

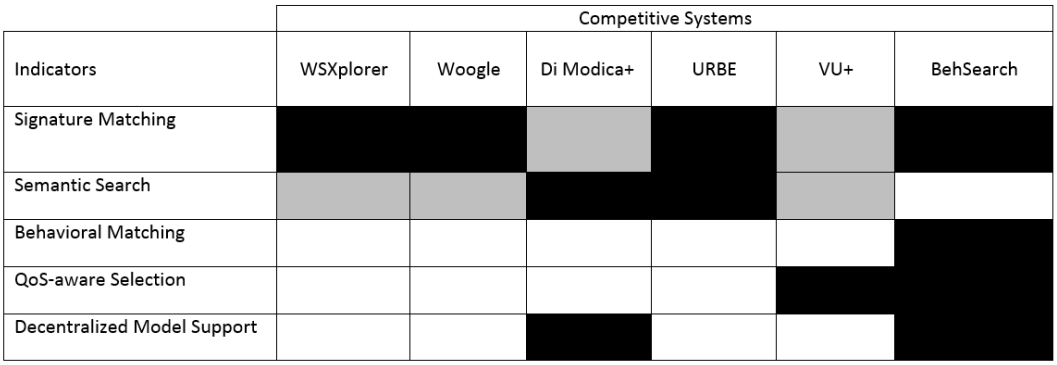


Figure 6.2: Comparison of ProxCG with similar models/systems.

flow diagrams to model interaction patterns among constituent services. This flow-based paradigm allows only basic control-flow patterns, i.e., sequence and parallel [101]. Moreover, some simple data mapping functions are defined in eFlow. Most of the workflow based approaches, like eFlow neglect specification of QoS properties. In eFlow, the selection of service component in an eFlow service composition should be done at the design time.

- **Intalio BPMS** [122] is a set of tools that provide various support functions for workflow interactions. Open sourced Intalio BPM uses "Designer", built on top of Eclipse to supports BPMN for process modeling. Moreover, "Designer" converts BPMN to BPEL to orchestrate web services. In fact, Intalio BPM provides a graphical interface to generate BPEL code for composition of web services. However, "Intalio Designer" does not support all elements of BPMN (e.g., loop transitions), because all BPMN diagrams cannot be converted to BPEL. Note that, BPEL4WS also suffers from the lack of support for non-functional QoS measures [7]. Moreover, Intalio stores XML data of the process instances to a database to support "Business Activity Monitoring" [122].

- **Self-Serv** [21] offers an orchestration model to support the composition of web services. Self-Serv aims to provide a multi-attribute dynamic selection of services within a composition. It also uses state charts to model and to encode the interactions among web services operations.

- **SHOP2** (Simple Hierarchical Ordered Planner 2) [110] is one of the most relevant tools especially implemented for semantic-based service composition. Shop2 is used for composition of OWL-S web services through a hierarchical task network (HTN) planner that works with the hierarchically structured OWL-S description of available web services. It supports only

simple control flow patterns, such as sequence and exclusive choices [101]. OWL-S description of services specifies QoS measures as service parameters.

- **SWORD** [134] is a set of tools for composition of web services using rule-based plan generation. SWORD does not deploy existing service-description standards, such as WSDL, SOAP and RDF. Instead, it defines an Entity-Relationship (ER) model to represent inputs and outputs of services. All entities and relationships among those entities are specified in a "world model" . A rule-based expert system is then used to generate the plan of service composition using existing web services. Finally, SWORD provides an execution environment to run the service(s) specified in the plan that are offered at design time.

- **XL** [59] is a new platform for web service composition, which uses a textual XML-based notation. The main advantage of XL is that a common data model is used for both the description of web services and the programming language. This advantage facilitates an automated data mapping in XL. Moreover, XL supports complex control flow constructs, including sequence, parallel, choice, and loop. The selection of components in XL occurs during the design time, so XL cannot support dynamic binding of web services.

- **TQoS** [54] provides a framework for selecting and composing web services not only according to their functional requirements but also according to their transactional properties and QoS characteristics. TQoS works based on two inputs: a workflow and the user's preferences. The workflow defines the execution order of component web services, and user's preferences are expressed as weights over QoS criteria. TQoS consists of a Planner Engine that generates an execution plan, and an Execution Engine that orchestrates the component web services to execute the instance of the composite web service. Moreover, the QoS of a composite service is evaluated by using several aggregation functions. The aggregation functions consider activities in all execution paths between AND-split and AND-join. TQoS uses a general workflow language to represent the orchestration patterns.

- **YAWL** (Yet Another Workflow Language) [163] is a work-flow language based system that uses Petri nets [170] as a starting point and offers an execution engine, a graphical editor, and a work-list handler. YAWL strongly supports complex control flow patterns by its visual notations. YAWL engine handles the execution of specified workflow instances (also called cases) to run the composite services.

- **ProxCG** (Proxy Code Generator) [83] is our proposed approach to compose web services, based on the Reo coordination language [9]. With Reo and our proposed tools, web services can be dynamically chained, and dynamically

integrated as a composite service in virtual organizations. We define below specific features of ProxCG under each applied criterion.

Figure 6.3 shows a summary of our feature analysis results for the above competitors versus our proposed tool ProxCG. If a tool completely supports a feature specified in the row, the corresponding box is black. If it partially supports the feature, the box is gray, and otherwise, it is white.

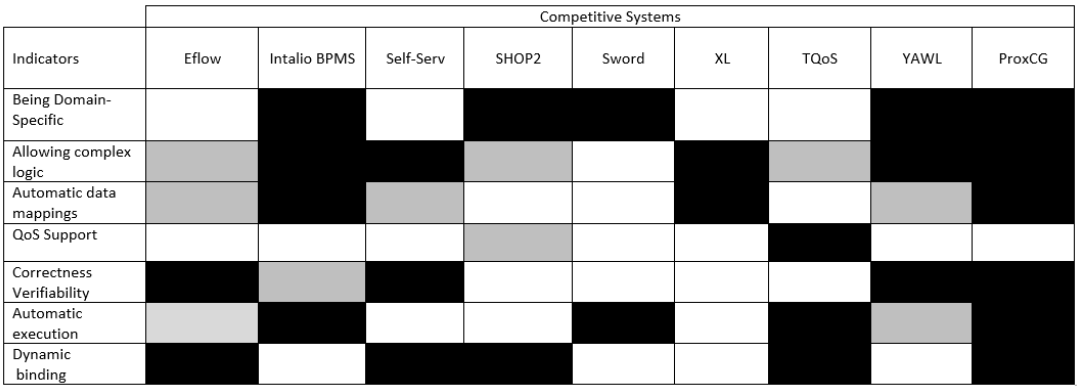


Figure 6.3: Comparison of ProxCG with similar models/systems.

Below, we define the main features as our key indicators to evaluate our composition tool, ProxCG against its eight competitors.

- **Being Domain-Specific.** Each composition approach uses a coordination language at its heart, which can be either a domain-specific language or generic. When the language is specifically built for coordination, it increases its efficiency. Our ProxCG tool works based on Reo, which is defined as a domain-specific language for coordination of the components and services [115].

- **Allowing Complex Logic.** The tools either can handle just simple interaction patterns (e.g., sequence choice) or complex ones. Supporting complex logic is considered to be one of the key highlights of the composition of services. In [84] we show how ProxCG supports complex behaviors of WS's orchestration by discussing some non-trivial examples.

- **Automatic Data Mapping.** Web services communicate with others only through soap messages [105]. Thus, each tool working with web services needs to build and/or parse SOAP messages, while automatic data mapping between input/output of such tools and soap messages skips this step. The Service Communication Unit (SCU) of our tool facilitates this kind of data mapping.

- **QoS Support.** To be satisfied by its clients, a composed business service should try to provide good quality regarding the clients' preferences. The consideration of QoS when composing web services requires a careful QoS aggregation of the QoS criteria of the constituent web services that compute the QoS of the composed service. An excellent research for supporting QoS aspects in composed services can be found in [159]. Most of the workflow based approaches like eFlow neglect specification of QoS properties such as reliability, availability, and throughput. BPEL4WS also suffers from the lack of support for non-functional QoS measures [7]. In semantic-based service composition approaches such as OWL-S and QoS measures are described as service parameters. ProxCG does not consider QoS aggregation when composing web services, however our specification and discovery tools support QoS criteria of the constituent services well [144].

- **Correctness Verifiability.** Verifying correctness is addressed as an important criterion for service composition [155], which shows the correctness of the composed services. While more of the other approaches offers support for the verification of a composed service, our tool is able to ensure the correctness of composed services [84]. For instance, BPEL4WS does not support any formalism to verify the correctness of the flows, because it deals more with implementation than with specification. [155].

- **Automatic Execution.** The feasibility of automatically performing composition of web services significantly reduces the complexity of such systems for end users [132]. ProxCG generates all necessary Java code to orchestrate the services automatically.

- **Dynamic Binding.** The component web services that form a composite service can be dynamically discovered, selected and bound, as in ProxCG. This feature gives more flexibility to a composition platform in order to replace or even find services at runtime.

6.4 Evaluation through Formal Experiments

The formal experiments with the tool are divided into two parts. First, in subsection 6.4.1 we validate the adequacy of the tool by measuring its information retrieval metrics. Then, in subsection 6.4.2 we evaluate the performance and scalability of the search engine in a decentralized implementation of the tool.

6.4.1 Correctness Evaluation

We have evaluated the quality of the computed results according to the two most frequent and basic metrics used to measure the effectiveness of discovered/re-

trieved information: precision and recall [79]. Precision (also called positive predictive value) is defined as the number of relevant returned results divided by the total number of returned results, while recall (also known as sensitivity) is measured as the number of relevant returned results divided by the total number of relevant entries in the database, which should have been retrieved.

Let *Rel* be the set of relevant WSs in the database. Let *Ret* be the set of returned WSs, *RetRel* be the set of returned relevant WSs (true positives), and $RetRel_k$ be the set of relevant results in the top k returned WSs. More precisely, the parameters that we have adopted to evaluate the performance of our approach are defined as follows:

$$Precision = |RetRel|/|Ret|$$

$$Precision_k = |RetRel_k|/k$$

$$Recall = |RetRel|/|Rel|$$

We have examined the results based on 20 test queries. For each of these queries, we have also manually labeled our WSs in the database as relevant or irrelevant. Since all queries had less than 10 results, we considered the precision at 2, 5 and 10 (for parameter k). The average $Precision_2$, $Precision_5$ and $Precision_{10}$ for the test queries are respectively 95%, 73.3% and 65.4%. This preliminary evaluation is promising especially for stateful queries, although for stateless queries our approach shows no advantage over the other similar approaches [133].

The Recall/Precision (R-P) curve is considered as the most informative graph showing the effectiveness of a search engine [51]. An ideal search engine has a horizontal curve with a high precision value and a bad search engine has a horizontal curve with a low precision value. The traditional approach to build a R-P curve is the 11-point interpolated average precision, where precision is measured at the 11 recall levels of 0.0, 0.1, 0.2, ... , 1.0. For each recall level, we then calculate the precision of the results that meet that recall level of the test collection. For example, if the number of relevant WSs ($|Rel|$) is 10, the number of returned WSs ($|Ret|$) at recall level 0.1 would be the minimum number of the top-ranked search results to see 0.1 portion of $|Rel|$, i.e., the first relevant WS.

The blue curve in Figure 6.5 shows the R-P graph of our tool (Beh-Search) for the 20 test queries, where recall level of the graph varies from 0 to 100 percent. For instance, Table 6.1 shows the results of search for one of the 20 queries to draw the R_P curve.

We have five relevant services in our database for this query. As we see in Table 6.1, the first relevant service (i.e., *serviceSMS*) appeared as the first top-ranked discovered service, so the precision at the recall level 0.2 (i.e., 1/5) would be 100%. The precision for the second relevant service (i.e., recall level 0.4) is also 100%, but for the third one it is 60% because the third relevant service is the 5th top-ranked service in the list. The precision at the recall levels 0.8 and 1.0 is respectively, 66.67 and 71.43.

Table 6.1: The top-ranked matched WSs, based on the query: $q0$ *SendSMS* $q0$.

Rank	Service Name	WSBS	Relevant	Score	Precision
1	ServiceSMS	q0 SendSMS q0	R	1.0	100
2	BulkSMS	q0 SendSMS q0	R	1.0	100
3	InfoService	q0 SendSMSInfo q0		0.63	
4	Server_utf8	q0 SendSMS_USC2 q0		0.59	
5	SmsService	q0 AddSMS q0	R	0.52	60
6	SmsWebService	q0 SMS q0	R	0.50	66
7	BSWS	q0 SubmitSms q0	R	0.42	71.43
8	ClickSmsV4	q0 GetSMSInfo q0		0.23	
9	Price	q0 GetSMSPrice q0		0.21	

Figure 6.4 shows one of the other queries that we used for drawing the R-P curve. This query aimed at finding the services that can search and apply for jobs. According to the query behavior represented in Figure 6.4, the intended service should first register the client, then search for his/her desired jobs, and, finally, apply for one of the retrieved jobs.

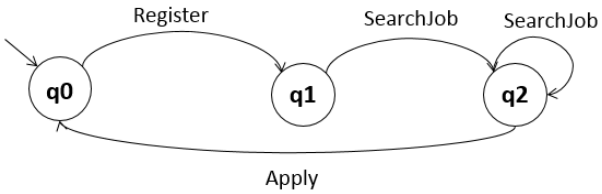


Figure 6.4: The query for discovering WSs that can search and apply for jobs.

Table 6.2 shows the top ranked results of the query represented in Figure 6.4. As we see in the table, the first six top-ranked discovered services are relevant, so the precision at recall levels $0.1, 0.2, \ldots, 0.6$ is 100%, which is very good. The precision at the next recall levels, i.e., 0.7, 0.8 and 0.9 are still high (respectively, 87%, 89%, and 90%). The precision at the last recall level, i.e., 1.0, which is the last relevant service for this query, drops to 59%. This service (behaviorally represented by ***q0 LookingForJob q0***) obtained 0.2 as the global similarity score, and is at the 17th row of the table.

The top-ten ranked services in Table 4.2, which are based on the query represented in Figure 4.12, is another instance of the queries we used to draw our R_P curve. As we see in the table, the first seven top ranked discovered services are relevant, so the precision at recall levels $0.1, 0.2, \ldots, 0.7$ is 100%. The precision at the next recall levels is also still high, e.g., it is 89% at the recall level 0.8. We can argue that the precision for multi-state queries are generally better than those for single state queries, which is the biggest advantage of our tool. The reason is that the transition-similarity score is only a part of the global similarity score, which is used for ranking of services. Therefore, if a service can be matched

Table 6.2: The top-ranked matched WSs, based on the query represented in Figure 6.4.

WSBS	Relevant	θ	σ	Γ	Rk
q0 Register q1; q1 Search q2; q2 Search q2; q2 Apply q0	R	.84	1.0	.92	1
q0 Register q1; q1 LookingForJob q2; q2 LookingForJob q2; q2 Apply q0	R	.62	1.0	.81	2
q0 Search q1; q1 Search q1; q1 Apply q0	R	.79	.67	.73	3
q0 login q1; q1 Register q2; q2 Search q3; q3 Search q3; q3 Apply q4; q4 logout q0	R	.84	.6	.72	4
q0 login q1; q1 SearchJob q2; q2 SearchJob q2; q2 Submit q3; q3 Resubmit q2; q3 logout q0	R	.53	.75	.64	5
q0 login q1; q1 Register q2; q2 LookingForJob q3; q3 LookingForJob q3; q3 Apply q4; q4 logout q0	R	.62	.6	.61	6
q0 Register q1; q1 Vote q2; q2 Vote q2; q2 Submit q0		.19	1.0	.59	7
q0 LookingForJob q1; q1 LookingForJob q1; q1 Apply q0	R	.49	.67	.58	8
q0 login q1; q1 LookingForJob q2; q2 LookingForJob q2; q2 Apply q3; q3 Change q2; q3 logout q0	R	.33	.75	.54	9
q0 SearchForJob q0	R	.74	.33	.53	10
q0 login q1; q1 Vote q2; q2 Vote q2; q2 Submit q0		.02	1.0	.51	11

behaviorally with a query but uses different labels, it is still expected to obtain a high global similarity score because it earns a good state similarity score. For instance, consider the third (service S1) and forth (service S2) ranked services in Table 4.2. The transition-similarity score of S1 (which is derived from a comparison between matched labels) is less than the transition-similarity score of S2, but the state-similarity score of S1 (which is derived from an approximate matching between the two automata) raises its global-similarity score. This means that even in these cases the tool can discover relevant services with a significant global-similarity score.

The R-P graph for our tool (see Figure 6.5) shows that for the first set of relevant results (i.e., at recall level 10%), precision is appropriate. Also precision for the last relevant results (i.e., at recall level 100%) is comparable with that of other existing approaches presented in [133]. Figure 6.5 also shows the Recall/Precision comparison for the studied approaches in [133]. Note that the data set used to derive our R-P curve is different from those used for the approaches studied in [133]. This is due to the lack of widely accepted benchmarks using stateful services and the dearth of tools that work with such services makes it not possible to have a meaningful direct comparison of our R-P curve with those of the other tools.

6.4.2 Scalability and Performance

Distributed computing presents an alternative to traditional centralized systems that can achieve high-performance in executing heavy-workload tasks [78]. In order to improve the performance of our retrieval tool, we have designed a peer-to-peer (P2P) model for its implementation on a network of equivalent computer nodes. P2P systems offer efficiency, scalability, decentralized control, self-

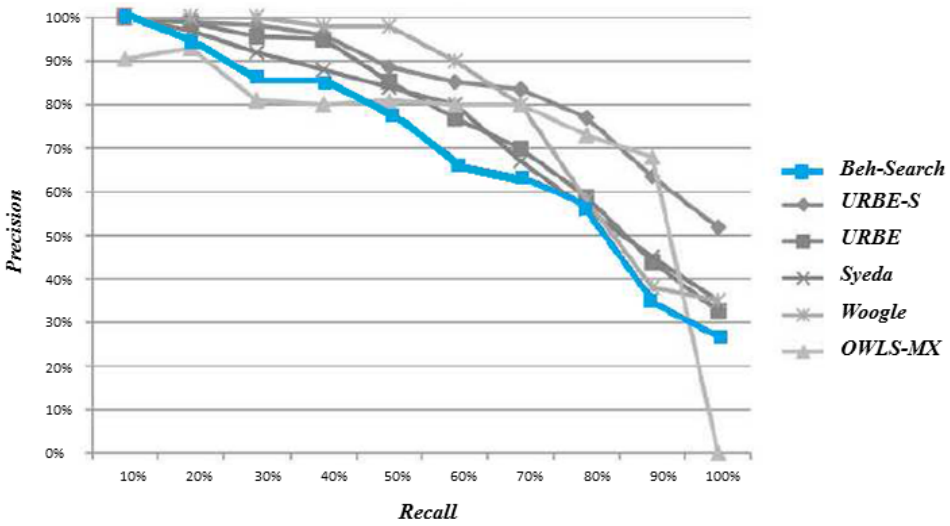


Figure 6.5: The R-P curve of some the related works.

organization, and symmetric communication [139]. In the P2P model of the tool, every node contains a list of services, as well as their searchable attributes for discovery. A query is passed to all of the nodes, and each node computes a list of similarity scores for its own content that matches the query as search results of our discovery tool. Each node sends its set of computed similarity scores to an aggregator, which merges the sets from multiple nodes to create a single set of ranked search results. Aggregators too send their results to other aggregators higher up in a hierarchy, until they reach the top-level aggregator that presents its ranked results to the user. The results of an aggregator node are directly comparable, because it receives the absolute similarity scores from different nodes. As an alternative, each node can send only the ranking of services, and use an aggregation function that weighs the ranking of each service by the number of services in the corresponding sender-node. In our implementation we adopted the former approach, which is based on the absolute similarity score.

The performance statistics that we describe below are calculated for 1000 registered services. Table 6.3 shows the execution time and the speed-up to obtain the similarity scores for the registered services, which are distributed among 1 to 10 nodes. Table 6.3 shows that the minimum execution time is obtained with 10 nodes, which is about 3.5 times smaller than the execution time with a single node, i.e., without distribution. The processing time needed for parsing the WSDL documents and generating the WSBS specifications is not considered in this performance statistics because this parsing is executed only once (in the registration step). The graph on the left in Figure 6.6 shows how the execution time varies with the number of nodes (processors). It shows that the execution time is inversely related with the number of distributed nodes. The graph on the

Table 6.3: Evaluation of the distributed model by matching the achieved speed-up with respect to the number of used nodes.

Number of nodes	Execution time	Speed-up
1	417 Millisecond	1
2	213 Millisecond	1,957746479
5	135 Millisecond	3,088888889
10	117 Millisecond	3,564102564

right in Figure 6.6 shows the rate of speed-up, which is equal to

$$\frac{\textit{the execution time using 1 processor}}{\textit{the execution time using n processors}}$$

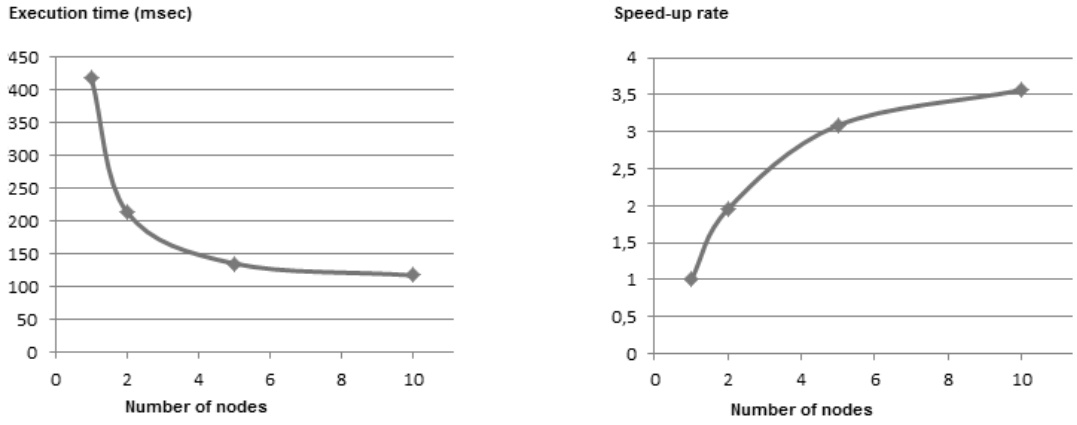


Figure 6.6: The graphs of execution time and speed-up rate for the example mentioned in Table 6.3.

6.5 Case Study: GloNet

Nowadays, with fast development of cooperating/co-working consortiums, collaborative networks paradigm offers an efficient mechanism to enhance enterprises' agility and resilience in disturbed business environments. Collaboration among SMEs and/or organizations has been referred to as an important paradigm to assuage the effects of market turbulence. In this regard, GloNet [1] as a funded European research project has played an important role in addressing, implementing, and supporing an agile virtual enterprise environment for networks of SMEs targetly development of highly customized products, enhanced by a number of

[1] http://www.glonet-fines.eu/.

business services. The development of such a system is realized through end-to-end collaboration, namely co-creation between local product-suppliers/service-providers and their customers. Pieces of this dissertation comprising of business service specification and service discovery are applied and validated in the framework of the GloNet project ([5], [2], [143], [35]).

Development of a collaborative tool to support service-enhanced products (e.g., solar power plants) typically necessitates contributions from many stakeholders/SMEs from diverse knowledge sectors in order to highly customize their production and services (e.g., befitting design of what exactly fits each case). The purpose of GloNet is to support such complex products, and involves a cloud-based platform [34] offering a collaborative networking framework for all involved stakeholders. The framework presents needed functionalities for tailored specification and development of such products and their associated business services, as well as for management of collaborative networks of involved SMEs. There are two virtual spaces over the cloud in GloNet: Collaborative solution space and Service provision space [29]. The former space is a VO for local manufacturers, suppliers and clients to collaboratively co-design the products and enhanced services. The latter space is a registry of such specified service enhanced products, where the clients can access them.

The notion of "glocal" enterprise, which means thinking/acting globally and responding adequately to local specificities, as well as value creation from global networked operations and service-enhanced products is realized in the GloNet. Our contribution in GloNet is mainly focused on implementation of the subsystems for Service-Enhanced Product Support. The GloNet architecture consists of several main components, however, in this thesis, design and implementation of Service Specification & Registration as well as Product/Service Discovery & Recommendation are addressed.

The term of service-enhanced product (also called complex product, and product-service in the literature) defines its numerous sub-products, as well as various business services that enhance those products (e.g., the monitoring service for the status of an electronic equipment). Solar power plants and intelligent buildings are two non-trivial examples of service-enhanced products, that are adopted for the GloNet project as real case studies. These complex products need to be collaboratively designed, developed, and verified by many involved stakeholders. Note that each sub-product and even each business service involved in a complex product might itself be offered by several manufacturers and/or suppliers. In this context, service-enhanced product points out associating business services that configure sub-products, in order to provide enhanced functionality for the complex products, and to differentiate them from similar existing physical products in the market. Such enhancement not only provides a higher level of differentiation from the competing products, but also increases the value of the products that are realized later. Indeed, the notion of enhancing services can also be seen as a marketing mechanism that augments the typical functionalities of a

complex product, especially in global markets [35], [2].

Specification of a complex product is a demanding task, which can take advantage of reusing already existing specifications, and customizing them. Since complex products have a dynamic nature, the supporting tools for their specification must be used at different stages of the Product Life Cycle (PLC) [152]. Complex product specification is mostly performed during the *design & engineering* phase of the PLC. However, it is also infrequently used in the *operation & evaluation* as well as the *Pre-PLC* phases. Note that the specification of a complex product is also updated iteratively and incrementally during its PLC, via collaboration among the consulting EPC (Engineering, Procurement, and Construction) companies, the corresponding customers, and the other involved partners, such as manufacturers, suppliers and service providers. Thus, within the GloNet framework, various stakeholders join together and form a working team, which is called the "Design Group", to provide the complex product's specification, including its sub-product specifications and business service specifications.

The targeted service enhanced products belong to young industries, and thus their stakeholders can very much benefit from sharing assets, such as specification of sub-products that are designed by other cooperators and competitors. Therefore, supporting both the reusability of service enhanced product specifications and the possibility of granting access privileges to other stakeholders are important requirements for this subsystem. Furthermore, considering that the targeted service enhanced products are one-of-a-kind, their designed sub-products and business services can be meant for reusing, if only their specifications follow a modular design approach so that within the VBE the pieces of service-enhanced products' specification can be discovered, accessed and copied for reuse. This point represents another crucial requirement supported by our developed product/service specification subsystem.

To put it in a nutshell, supporting such complex products involves various collaborative networks, operating at different stages of the life-cycle of the products and its enhancing business services development, which motivates the GloNet project [36].

In the remainder of this section we exemplify some real cases taken from GloNet, and we present the application of our approach and the developed prototype.

6.5.1 Product/Service Specification(PSS)

The "Service Enhanced Product Support" of GloNet comprises two sub-systems: Product Portfolio Management and Product/Service Specification (PSS). The Product Portfolio Management aims to provide mechanisms and functionalities supporting search and display of the portfolio of products, i.e., the repository that stores all specifications of the service enhanced products. PSS on the other hand is a sub-system in GloNet, consisting of three main sets of tools: the Product

Specification & Registration Tool (PST), the Service Specification & Registration Tool (SST), and the Product/Service Discovery & Recommendation Engine (PSDR). The PST and SST tools respectively support the reusability of detailed specifications of sub-products and their enhancing business services. The specifications which are generated for complex products in PSS, is then stored as an input to product portfolio sub-system and associated to a customer. Moreover, the PSS sends a request for launching a VO, which starts the configuration/formation process of a VO corresponding to the specified service or product. Therefore, PSS plays a pivotal role in the success of functionalities addressed in the other sub-systems of GloNet [2].

Together, the PST and SST tools effectively support the detailed complex product specification that constitutes the main input to identify the required offers (also called competencies) of supplying organizations, and thus enable the m the seolecsoft-tion fit organizations for the VO creation. However, the main contribution of this dissertation in relation to GloNet is to introduce the advanced service specification tool SST, for specifying business services associated with the complex product, as well as further supporting their reusability, potential integration, and discovery through the PSDR tool.

PSS tool provides access to PST, SST, and PSDR tools, after a user has logged in and authorized. Note that when the Design Group is set to Private, the user works only on his/her own private space without any other partner involved, but he/she can also indicate another Design Group in its sub-menu, involving other partners in product/service specifications. This option provides the possibility to assign different users the needed access right to their defined service specifications. This is mainly used to assist the user with organizing his/her own service specification folders/directories. Figure 6.7 shows the general view of the PSS. In the figure on the left side, three tools (PST, SST and PSDR) developed in the PSS sub-system of GloNet, and their main functionalities are represented, each opening a set of menu items. In the center of PSS, a data entry space for the tools (SST, PST, or PSDR) is represented according to the called functionality, selected from the left side. On the right side, a number of suggestions may be represented, to help the user with the specification processes. These include suggestions of related classes, suitable sub-products, recommended enhancing business services for sub-products, and suggested features, that can be used by the designer in his/her design.

Figure 6.8 demonstrates the overall flow for the specification process of complex products that happen in GloNet during the PLC. The process starts with "Create Tender" and then continued by PSS to specify the complex product. In PSS, a directory is created (labeled as "Create Directory") to store the complex product's specification, and then a rough specification of the complex product is provided as the "initial specification of the complex product". Furthermore, a "Design Group" is formed of the stakeholders that have the access rights to view, edit and launch the complex product's specification. Finally, the initial specifi-

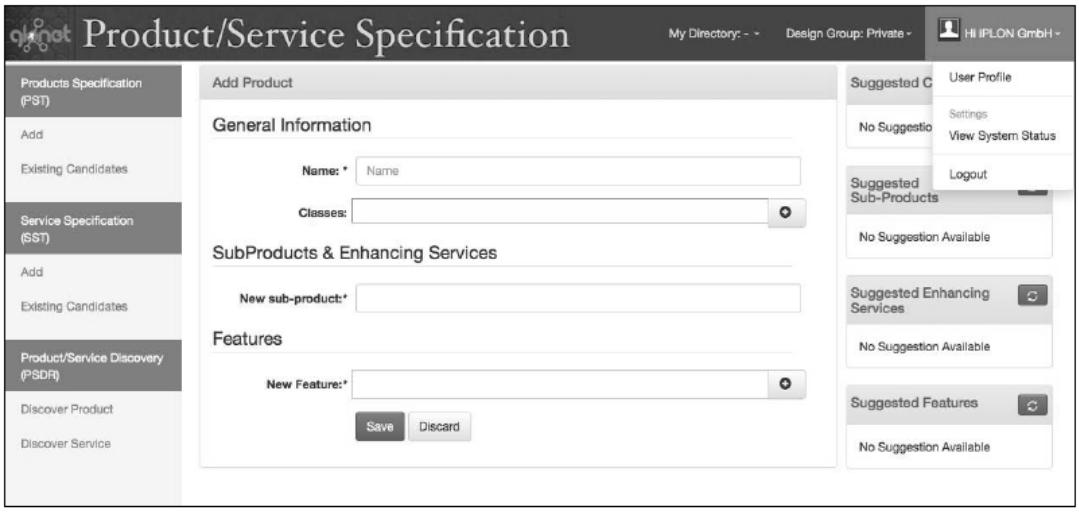


Figure 6.7: the general view of the Product & Service Specification (PSS).

cation is shared among the "Design Group", and it is ready to be extended by detailed specification of the sub-products and enhancing services using PST and SST. The sub-processes of "Detailed Complex Product Specification", which is done by Product Specification Tool (PST) and Service Specification Tool (SST) are represented later below in details. Please note that all specified details related to the sub-products and business services are supported for reusability. Thus, the existing specifications can be reused for the specification of other further sub-products/services. Finally, when a designer completes the process of specifying a complex product, he/she may wish to initialize the process of realizing that service. For this purpose, he/she can announce this fact through building a request for launch, which is labeled as the "Launch the Specification" in Figure 6.8.

6.5.2 Service Specification & Registration Tool (SST)

Although many approaches and standards have been developed by research community in the area of Service Oriented Architecture (SOA) that can be applied for the purposes to specify, formalize and deploy business services, it is still a challenging task to make these services interoperable, so that they can be shared, searched and reused [4]. Another challenge of service industry is how to assist authorized service providers with composing some of the existing services, thus producing value-added new services in support of complex products. Furthermore, there are still some gaps in correlation between existing services and sub-products in the context of complex products.

The specification of services, as the first step to support service reusability and interoperability, can be performed during different phases of the PLC. In GloNet, we formalize a business service specification as a tuple $S =< Sn, Sm, Be,$

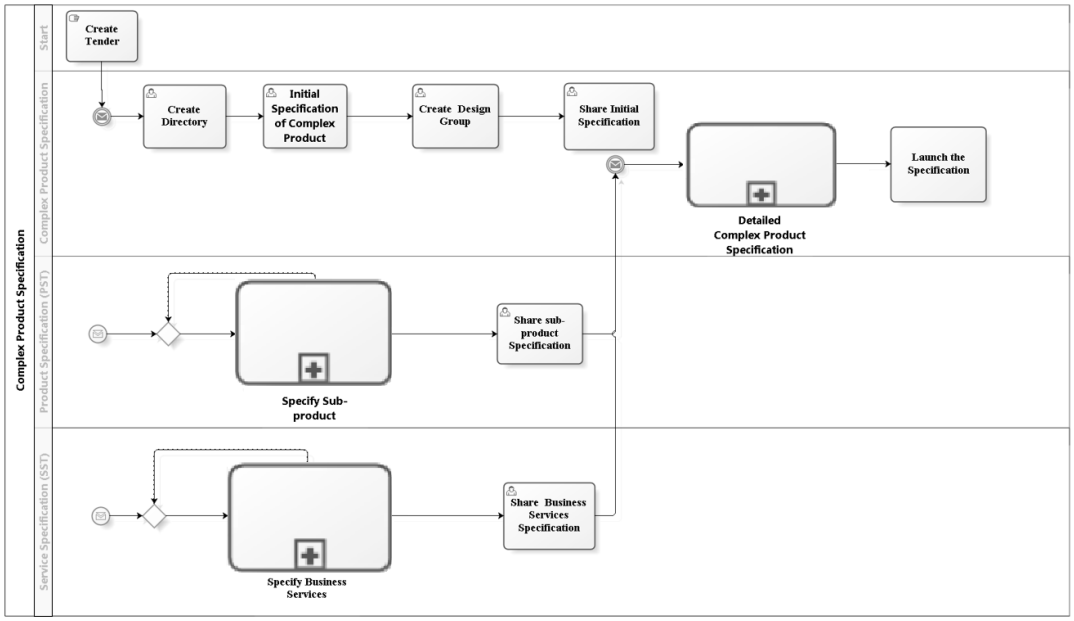


Figure 6.8: Main flow of the Product & Service Specification (PSS) Process.

$Qcs >$, where Sn, Sm, Be, and Qcs respectively represent the Syntax, Semantics, behavior and Quality criteria of the service. As described in chapter 3, each business service can be materialized as a manual tasks, or a software service. The SST tool supports and covers all four aspects of this proposed service specification for both of these kinds of business services, as well as the business services with a combination of some manual tasks and some software services, called composite services. Each element related to the service specification, so-called "feature" is registered in GloNet as a data object, with four properties, including service's Feature-Kind, Type, Value and Unit. In fact, every specified service through SST, is characterized by a set of features. The data stored for specification of the manual tasks is also similar to the software services. In other words, for uniformity purposes, and for the purpose of monitoring service executions, we also define a simple web service for each manual task that includes only two basic operations: Start and Stop. It is also possible to define a composed business service through SST. We call this kind of services as composite business service in the sequence.

Figure 6.9 shows a snapshot of the interface window for SST, where the Add sub-menu is selected from the menu under the service Specifications (SST). This window is used to support specifying a new atomic and/or composite business service. In the New Service window, the user can add an atomic/composite service specification by first providing a unique name for the service. The user can then optionally define one or more classes for the specified service.

Once the user, who is specifying a service, defines a class for it (e.g., the class

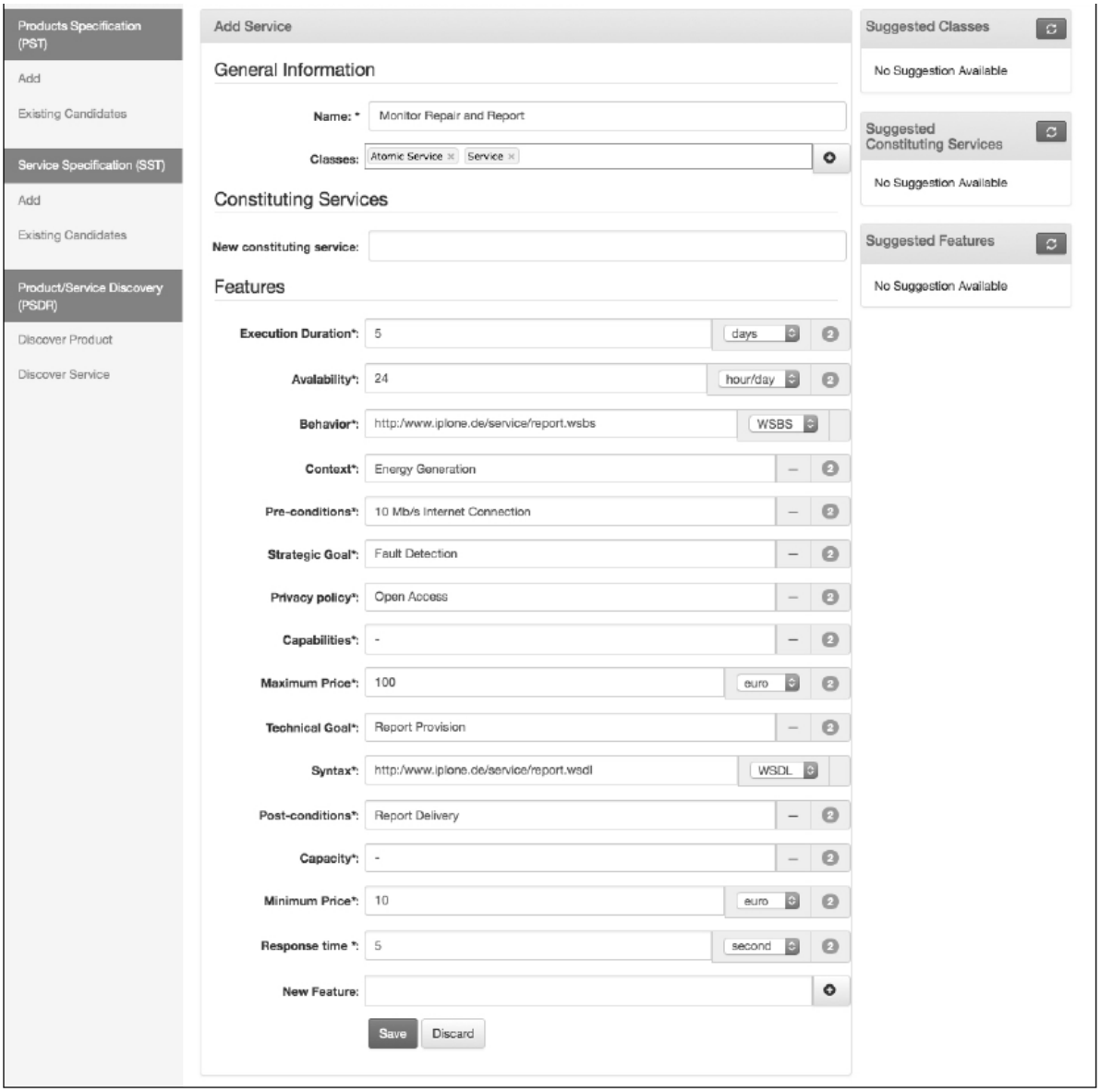


Figure 6.9: New Service specification form.

"Atomic Service" as in the example in Figure 6.9) through the system, all features defined for that specific class will automatically pop up in this window. As such, the user is then assisted with receiving the names of all features, which he/she should fill and specify as a part of the specification of that service. Furthermore, while providing input for some of those features might be optional, some other features of the class might be defined as mandatory, obliging the user to provide the needed input. For example, in Figure 6.9 since the user has added the class "Atomic Service", for the new service which he/she is specifying, a set of feature-kinds for this service have popped up in the center of the New Service specification

form (Figure 6.9) for its further specification. These include the mandatory set of feature-kinds (as marked with "*") such as "Context", "capabilities", and "Response time", which show up automatically on the screen. Therefore, providing input value (i.e., features) for these feature-kinds are obligatory. User must define both value and unit for these features.

The features used for service specification can be added/removed dynamically in SST. If the user wishes to add new features to a service specification, the new feature can be specified at the bottom of the screen through the "New Features" icon of the interface. The user will then need to indicate the feature-kind to which the new feature corresponds, and then the value and the unit for that feature. It is important to note that based on the feature-kind that the user selects/identifies for a new feature, the data-types for its value and unit will differ, according to those that are defined for the corresponding feature-kind. However, if the mentioned feature-kind is not already defined in the system; the user will then be immediately prompted with the window that asks him/her to first create that feature kind, before going further with the definition of the newly introduced feature. Indeed, every specified service through SST, is characterized by a set of features. For consistency reasons, any identified feature in this sub-system requires that its feature-kind is defined priori to its definition.

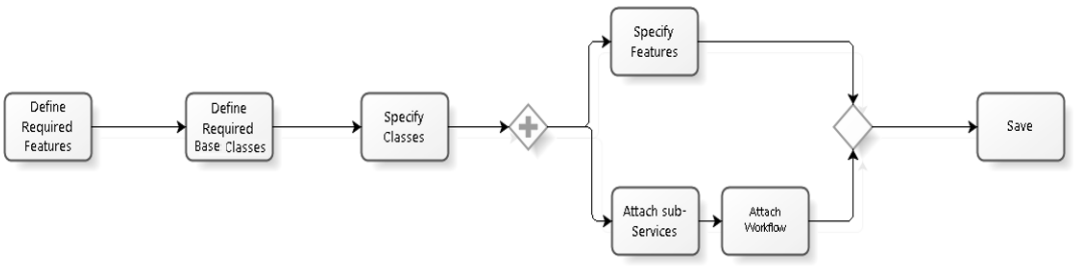


Figure 6.10: Flow of the service specification (within SST).

Figure 6.10 illustrates the sub-processes within the SST for specifying business services. To specify services, first the required features, and then the needed base classes, which do not already exist, should be defined. The next step for the user is to identify which high level class (e.g., Atomic Service) this business service falls under. The box labeled "Specify Classes" in Figure 6.10 represents the specification of the sub-process involved. In the case that the service is atomic, the flow is followed by specifying its features. For the composite services however, it is also needed to both attach the component sub-services and the orchestration workflow, which are described in the next subsection. Finally, the specification of business service is saved.

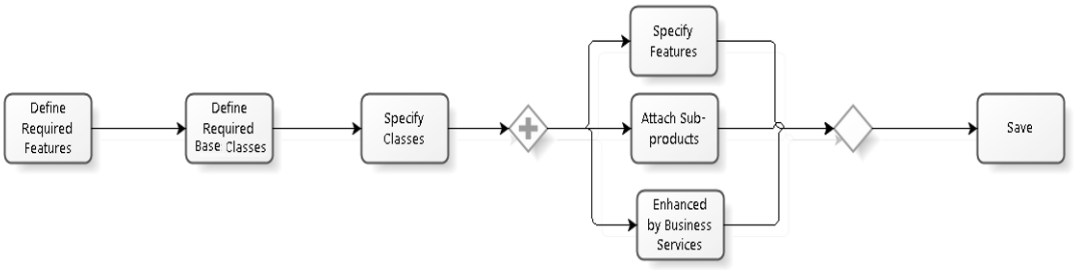


Figure 6.11: Flow of the product specification (within PST).

Although, the focus of this work is on service specification, the flow of the product specification is very similar and represented in Figure 6.11. As you see in this figure, the process of PST is close to the process of SST (see Figure 6.10) while the a set of enhancing business services are also defined to be associated to the product (i.e., the box labeled "Enhanced by business services").

6.5.3 Composite service specification

As mentioned earlier, SST enables users to define atomic business services, which may include both software services and manual tasks. These can in turn be used as constituent services for specification of composite business services. Figure 6.12 shows an example of a composite business service in GloNet, which is a site maintenance service for solar plants consisting of four atomic services as its components, including: "Check & Report", "Vegetation Management", "Wildlife Prevention", and "Water Drainage". As shown in Figure 6.12, these four component business services are provided by three different companies, including: a Security Company, a Site Cleaning Company and a Wildlife Prevention Company. The atomic services as the components of this composite service, are a combination of software services (e.g., Check & Report) and manual tasks (e.g., Water Drainage).

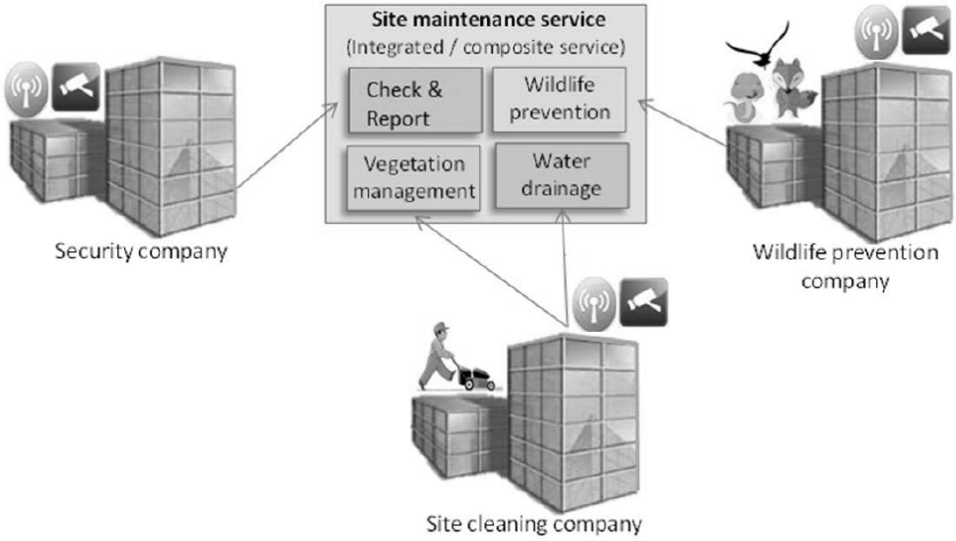


Figure 6.12: An example of composite business service for solar plants called site maintenance service.

The information required for composite business service specification is similar to that for the specification and registration of atomic business services, except that it does not include the syntax and behavior aspects, since these are captured in details within their constituent services, e.g., their corresponding atomic service specifications. But additionally, the composite service, e.g., Site maintenance service, specification includes an aspect called "Bundling", which consists of a feature-kind called "Constituent Services", to clearly specify the set of its component service constituents, e.g., "Check & Report" and "Vegetation Management", etc. Therefore, composite services are defined through their three aspects of Semantics, Quality Criteria, and Bundling. Moreover, the proper specification of a composite service must accompany the concise specification of the interconnections among its constituting atomic services, i.e., its coordination model. For this purpose, an orchestration workflow (e.g., a BPMN diagram) is needed for the composite business services specification in GloNet framework, representing the coordination among its constituent services. This aspect is supported by another subsystem of the GloNet, namely BPMN Modeler [145], which models the interaction patterns among the component services in terms of BPMN diagrams. The PSS sub-system and BPMN Modeler are connected and integrated through the GloNet platform.

There are several other aspects involved in specifying a service that can be defined through this interface. For instance, to define a new composite service, the user can easily indicate the components of this composite service using the

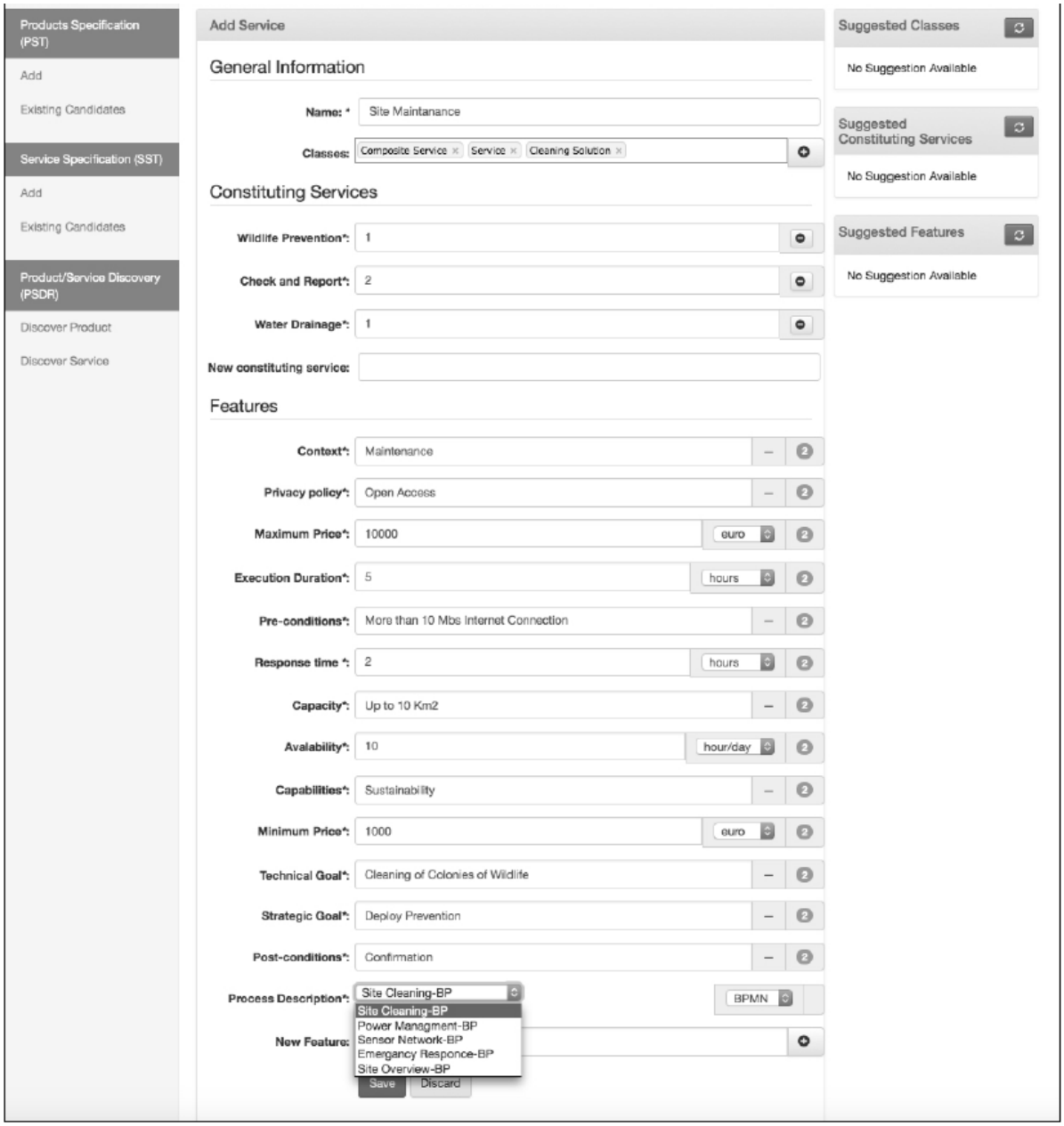


Figure 6.13: New composite service form.

"New constituent Services" part of the "New Service Form". Note that for each constituting service, its needed quantity should also be specified. Figure 6.13 shows the new service form for the maintenance composite service (see Figure 6.12) that needs to deploy the "Check and Report" service twice, assuming that it is needed once before the analysis of the damage and once afterward. Therefore, the user has indicated "Check & Report" with the quantity 2, "Wildlife Prevention" with the quantity 1, and the "Water Drainage" with quantity 1. But

clearly, as well as the quantity, the interaction protocol of the composite service must be specified to model the coordination of constituting atomic services. The "Process Description" feature of new composite service form (see Figure 6.13) is used to provide a reference to the desired orchestration workflow for composite business services. For this purpose, the service designer (current user) creates his/her intended workflow through the BPMN Modeler subsystem of GloNet platform, and uploads it to the platform. Then through the PSS subsystem when specifying composite services, the system provides a selection menu including the list of existing workflows owned by the user, to potentially select from. As shown in Figure 6.13, the designer is choosing his/her desired workflow, the "Site Cleaning-BP", for the composite business service.

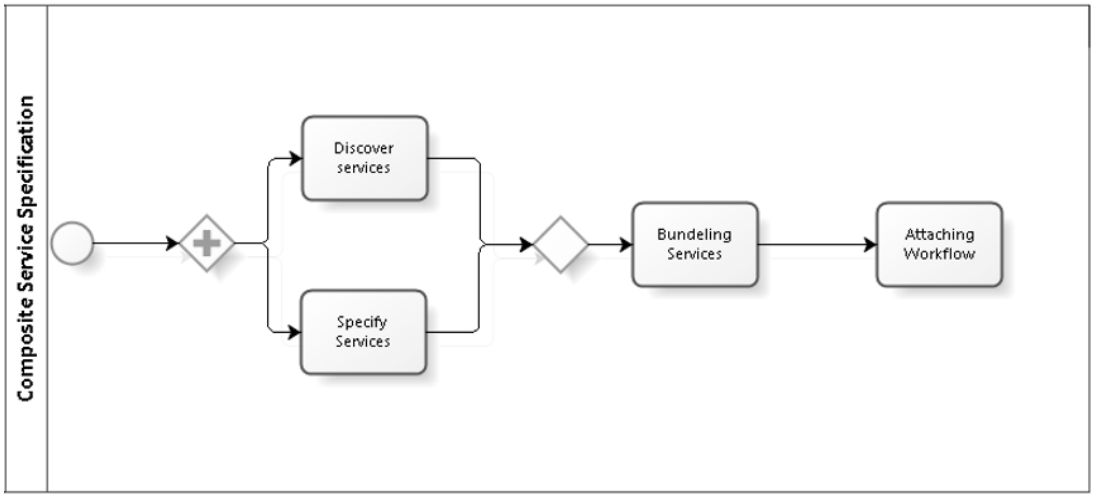


Figure 6.14: Flow of the composite service specification (PST).

Figure 6.14 shows the flow of the process to specify composite services. The flow is commenced with identification of the constituent services. Such services can then be either specified from scratch as a new service, or retrieved through discovery of already existing service specifications. The boxes labeled as "Specify Services" and "Discover Services" in Figure 6.14 respectively represent the subprocesses involved. Please note that a discovered constituent service can be either atomic or composite. The constituent services resulted from the last two steps are then bundled to form a new composite service. As mentioned earlier, we need to attach a workflow to coordinate the composite service as its final step of specification.

6.5.4 Viewing / managing existing service specifications

Once services are specified, they can be viewed by selecting the "Existing Specifications" item under the service specification menu. As such, depending on

the selected Design Group (as indicated in the upper right corner of the screen in Figure 6.7), their associated existing specification window will appear, showing the list of all relevant existing service specifications (sorted by their names), which the user is then authorized to view. In other words, the specifications that are included in this window are all those related to the specified Design Group. For example, in Figure 6.7, the "iPLON" user has selected/indicated his "Private" Design Group. Consequently, in this example, only the restricted service specifications that belong to this Design Group are shown.

Other than viewing the service specifications, authorized users can also manage these specifications by preforming the following set of actions:

- *View*, which takes the user to the view details of the service specification. Figure 6.15 shows an example of this view for the specification of "Analysis Natural Damage" as an atomic service.

- *Duplication*, which takes the user directly to a pre-filled "New Service" form. This simplifies the task of users since in that form the specification information about the selected service is duplicated, which can then be edited by the user, for defining a new similar service specification.

- *Hide*, which allows hiding the corresponding service specification from the specific existing services window, which is restricted for only the Design Group. For instance, the user finds a service that is useless for him/her use and so the user hides this specification from his/her view.

- *Add to Directory*, which provides the possibility to assign an already defined service specification, which user is authorized to access, to an existing directory of the user. This is mainly to assist the user with organizing his/her service specification folders.

- *Share with Design Group* , which provides the user with the option to change the access rights/sharing status of a certain service specification that he/she owns. The share options are available through existing services window, when the user clicks on its icon. Please note that when defining a new service specification, the access right to that specification is made private to the user who defined it by default, that is if the user has not indicated a Design Group on the top right corner of the Figure 6.7, otherwise the specification will become restricted to that Design Group by default. Clearly, at any point in time, the owner of the service specification is allowed to broaden the access to its owned service specification.

- *Request for Launch* , which enables the user to issue a request for launch of a service specification. Using this option, the user can send a new launch request for one of the already specified services. Please note that before a new Launch Request can be built for a service that the service must be

first properly specified. Then through the interface presented in the Launch Request form, the user will identify the specified service (i.e., indicates its specification), which will be included in the package for this request of launch. For more details, we refer the reader to the report presented in [143].

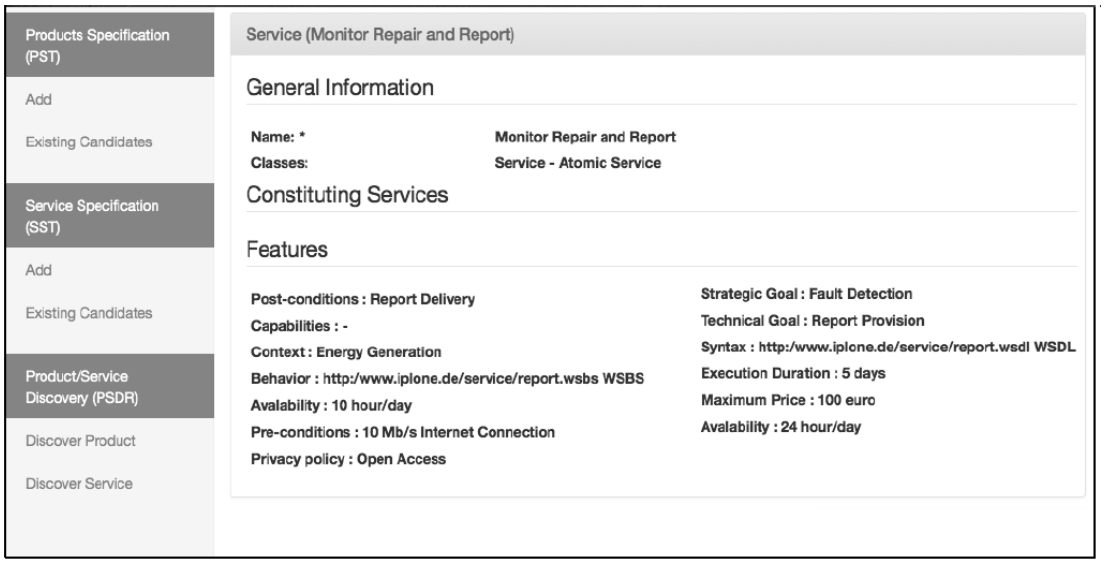


Figure 6.15: View form of a Service Specifications.

PSDR- Product/Service Discovery and Recommendation

In Figure 6.7, the menu item on the left side for Discover Service is located under the Product/Service Discovery (PSDR) menu item. It opens a form to both discover services and to rank the matched suggestions based on the user query. This constitutes a part of the product/service discovery and recommendation engine of the general GloNet architecture. This part of the tool addresses our mechanisms for discovering and matchmaking between the users' required criteria and the existing service specifications, in order to support the service designers with offering them the best-matched business services to their request. As we described in Chapter 4, the ranking is done according to the similarity score that expresses approximate bi-simulation between registered specification of each service and the users-submitted query. The discovery of best-fitting services can be done based on the entire service specification features, including the service's syntax, semantics, behavior and quality criteria aspects. Figure 6.16 shows a screenshot of the Service Discovery form, where the user can select some of the service features as the criteria for his/her search, and also set his/her desired values for them. Figure 6.17 shows the result window of the matched services for the example query, which is represented in Figure 6.16. The figure shows the

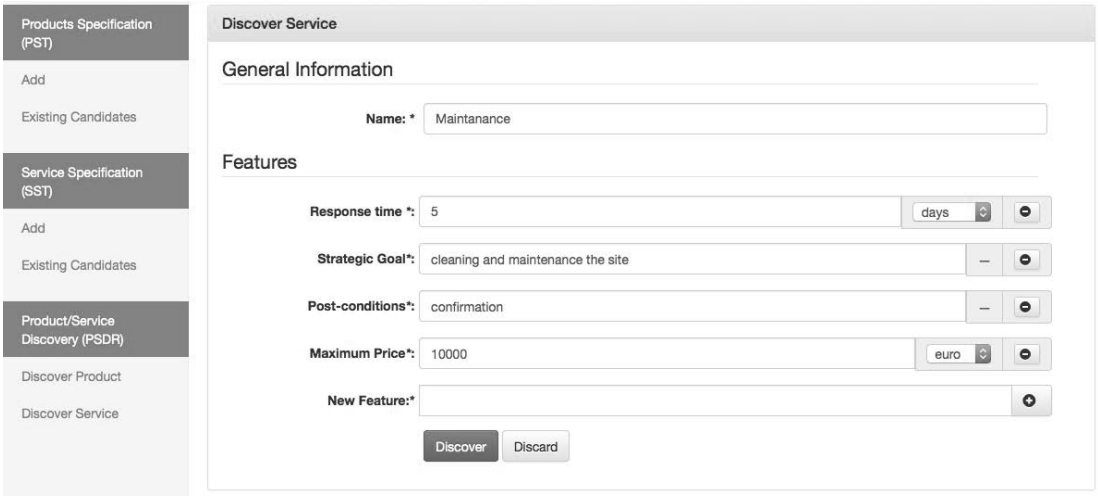


Figure 6.16: Service Discovery form.

first highly ranked discovered services. The results are ranked by the calculated similarity scores of the registered services.

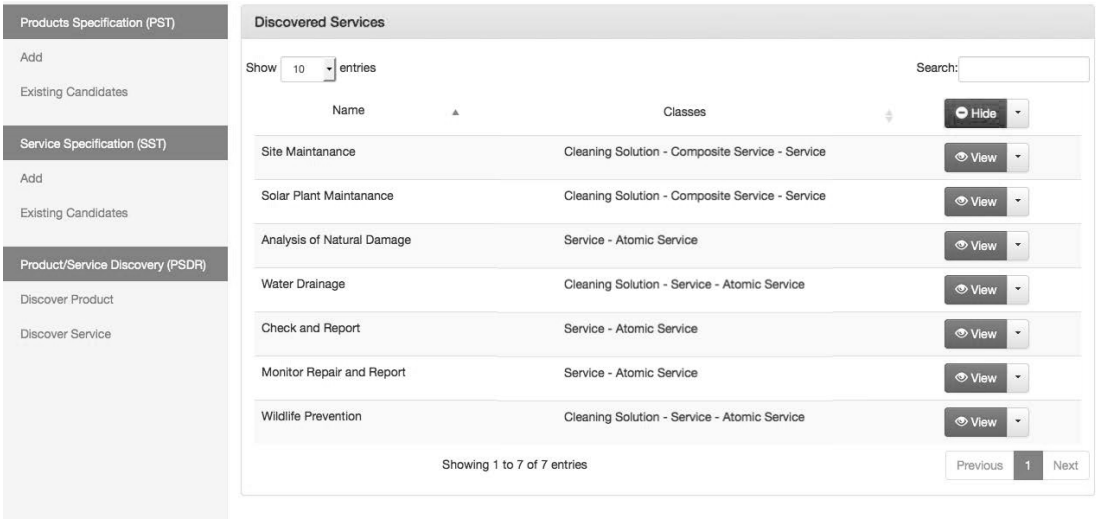


Figure 6.17: An example of Service Result window.

6.6 Conclusion

This chapter presents a summary of the achieved results and their validation. First, it briefly presented how in the thesis the developed tools and results can be evaluated in relation to our targeted research objectives. The chapter then addressed a feature analysis evaluation of our proposed solutions and concepts.

In this way, XWSDL, BehSearch and ProxCG as the new methods and tools introduced in this thesis, are compared with several related works. After that, it showed the evaluation results for the performance and accuracy of the discovery tool, which is at the heart of its framework. The scalability and accuracy of the BehSearch is measured based on some experiments. Finally, GloNet is introduced as the case study to demonstrate and validates the usage of our approach, concepts and prototype in real case from the service enhanced solar power plant industry. We use our methods and tools for service specification and service discovery to support Service-enhanced products in GloNet.

Chapter 7

Conclusion, and Future work

Service oriented architecture (SOA) is gaining more and more attention in business and industry since it offers new, agile and flexible ways of supporting the intra- and inter-organization activities. It propounds the idea of service orientated collaborative networks (SOCNs). However, the current implementations of the SOA approaches often do not deliver the expected advantages [158]. Several major problems in this area can be identified. First, for inter-organization activities a completes and clear understanding of the notion and support for SOCN is still lacking. As a consequence, it is unclear what functionality should be implemented to realize such systems [4]. Second, an appropriate framework for concisely specifying business services is needed, to supplement the insufficient information about business services, and concerning details of various aspects of the respective services [158] and [53]. Third, an enhanced service discovery mechanism is needed in order to give more flexibility and efficiency to search for services, through their various aspect and criteria, rather than only through their general keyword-based delineation [3] and [53]. Finally, to fully leverage the opportunities provided by service oriented architectures within organizations, integration of existing services must be simplified [155], [76] and [4]. As a consequence, approaches for automated service composition, as well as execution of integrated services are needed.

The above problems were posed as the research questions for this thesis in the Introduction Chapter. In this chapter, we briefly recall and discuss the answers to these questions, and point out several future research directions.

7.1 Addressing Research Questions

To answer the first main question RQ1 on how to support organizations to effectively co-work and co-develop in VO's, in order to share a part of their business service capabilities and allow their reuse and integration, we introduce a new framework for service oriented collaborative networks in Chapter 2. Applying

Software services, e.g. web services, and SOA paradigm in collaborative networks using rapid, cost-effective and standard-based means, results in more successful interoperability and collaboration among involved organizations. For this purpose, a new high-level abstract framework for service oriented VOs is introduced in Chapter 2.

This framework consists of three software modules, i.e. Specification Module, Discovery Module, and Composition Module. Specification Module is responsible to fully specify all aspects of software services offered by service providers as VO partners. This module has two functions: Agreement Management and Software Service Specification to capture the agreements and responsibilities of VO members, as well as needed meta-data of services (e.g. syntax, semantics, and behavior of services). The second module is the Discovery Module, which addresses mechanisms for service discovery and selection of the existing shared services in the VOs. Finally, the Composition Module of the abstract framework offers new value-added services to the service designer, through the composition of the existing shared services in the VO.

In order to answer the second main question RQ2 on how to address the business services in service oriented collaborative networks to support their effective reusability and integration, we pinpoint below each of its two subsidiary questions introduced in the Introduction Chapter, and specify where they are addressed further in the thesis.

In Chapter 3, we address the sub-question S1-RQ2:

How to uniformly define business services of independent and autonomous organizations constituting the VO?
To answer this question, the C3Q model of service competency is proposed in Chapter 3 to have a uniform definition for business services. The 4C-model of organizations' competency introduced in [57], forms the basis of our proposed C3Q model of service competency. The C3Q model characterizes Capability, Conspicuity, Cost, and Quality specification criteria. As such, VO business services can be uniformly defined and published in order to support their sharing and reuse. Capability is the most important part of the service profile, which represents the functional properties of the business services including syntax, semantics, and behavior. Nevertheless, one aspect which deserves further emphasis here is the behavioral aspect. To the best of our knowledge, C3Q is the only complete service profile that addresses behavioral properties of the services in addition to their other aspects. The C3Q model can also be considered as the service competency model within the VOs.

The sub-question S2-RQ2 below is also addressed in Chapter 3:

How to specify, in an unambiguous/concise representation manner, the functionality/behavior of business services, as required for developing their equivalent software services?

In our approach, the current business service description approaches and standards are extended and improved in order to support efficient service discovery and composition. First, we present the C3Q data model to represent the various information needed for description of the business services within the VOs.

A number of different notational options can be applied for representation of each of the three aspects of this data model, we have however adopted one specific notation in our proof of concept prototype for formalizing each of these aspects, as addressed in Chapter 3. WSDL that represents the most prominent standard for describing web services, provides only the syntactical description, and lacks representing the other properties of services, e.g. semantics and the behavioral specification which is required to represent the functionalities of services. Therefore, we introduce an extension of WSDL, the XWSDL, to specify web services according to the aspects of services introduced in C3Q model. To the best of our knowledge, XWSDL is the first model that provides a comprehensive description of capabilities over web services and highlights the important role of service behavior in the realization of the semi-automated service oriented computing. Finally, a prototype GUI is developed to assist service integrators and service providers with describing and visualizing the behavioral specification of their web services, correctly and easily.

To answer the third main question RQ3 on how to support the selection of most-fit business services against user-defined desired criteria, we address below each of its two subsidiary questions.

The sub-question S1-RQ3 below is addressed in Chapter 4.

How to enhance the accuracy of service discovery results?

In Chapter 4, a tool is developed for a similarity-based discovery of web services that is able to rank the service descriptions in a database, based on their similarity scores matching each with the description of a service desired by a user. The formal framework behind the tool is based on Soft Constrain Automata (SCA) [15], to represent high-level stateful software services and queries. We use SCA to formally reason on queries, e.g. on their operational similarity. Our tool is based on implementing approximate operational-similarity evaluation with constraints (see Section 4.6), which allows to quantitatively estimate differences between two behaviors. Defining this problem as an SCSP makes it parametric with respect to the chosen similarity metric (i.e. a semiring), and allows using efficient AI techniques for solving it.

The sub-question S2-RQ3 below is also addressed in Chapter 4.

How to ensure the quality of retrieved/discovered services?

Besides the set of functional requirements, which we model as soft constraints in our discovery tool, we can also encode any set of QoS requirements as soft constraints, in order to assist users in their service selection. Therefore, we enhance our tool by also considering QoS metrics to further meet users needs.

In principle, we can compute QoS ranking as an extra criterion for search, but doing so may impact the results by giving higher ranking to completely irrelevant services that have high QoS values. We have therefore used the lexicographic ordering of our soft constraints in order to avoid this problem. Thus, we combine the non-functional requirement constraints of services with all functional constraints in our constraint assembler, but doing this in a given lexicographic order.

To answer the fourth main question RQ4 on how to model the complex interactions and communications among several component services that form a composite service, in Chapter 5 we show how to automatically generate an orchestration framework for web services. Our answer is provided below, through two sub-questions.

The sub-question S1-RQ4 below is addressed in Chapter 4.

How to represent internal configuration (i.e. behavior) of component services, as required for their automated invocation?

We need to model and monitor the internal configuration of stateful services in order to support their automated execution within a complex composed service. Constraint automata [18], we specify the behavior of component services in our proposed framework. In particular, we emphasize the external view of services, namely every state of a simulation automaton represents an externally observable internal configuration of a service, while every transition represents the exchange of one or more messages by this service.

The sub-question S2-RQ4 below is also addressed in Chapter 4.

How to model complex interactions and communications among several component services to be integrated into a composite service?

We have adopted Reo to support the orchestration among composing web services. We then generate the Java code to compose the behavior of the WSs in a

way that is transparent to the client and all the WSs. Reo [9] is then interfaced to constraint automaton, which provides a graphical language for modeling the interaction protocols among composed software services and orchestration their execution. The input to our Java code generation tool includes the web service behavior specification of each WS (in terms of constraint automata), the WSDL description file of each WS, and the specification of the interaction protocols among the component services as a Reo circuit. Using these information, it is then possible to automate the Java code generation process from a Reo circuit and a proxy for each WS. Such a proxy manages the communication between services and the Reo channels. Our developed ProxCG tool automatically translates Reo circuits generating the executable orchestration of the JAVA code for composed services.

7.2 Discussion and Future Work

This research can proceed in the future along a number of different lines. Our first intention is to be able to expand our current approach to generate different communication units for the proxy (the service communication unit, Sect. 5.5.1), in order to include different technologies in the orchestration of components and web services, e.g. CORBA, RPC, or WCF. We intend to support a multi-technology platform that can support integration of services from different kinds of third-parties.

Another related direction for future work is extending our work from Soap-based WSs to Restful WSs. Such an extension would provide a powerful, declarative platform for composing and mixing Soap-based WSs with Restful WSs, with a view similar to the Pautasso's work on Restful WSs for the imperative Bpel platform [127]. We identify several challenges here. Perhaps the most crucial question is how to map resources to boundary nodes, i.e. how to connect a circuit to a Restful WS. The obvious choice of representing every resource location with a boundary node may not work well in practice if a service uses many different resource locations. Finding an optimal solution requires further study. Another challenge here is the lack of standardized, widely adopted, machine-processable interface description language for Restful services. Such a format seems essential to generate proxies completely automatically. A third challenge concerns dealing with the state of web services. Stateful Restful services do not maintain state at the server, but at the client. The client includes all necessary state information in each of its requests to the server; the server includes in each of its responses a resource location for every admissible next transition. Interestingly, Restful WSs thus provide more behavioral information at run-time than Soap-based WSs in their WSDL file at compile time. We have in fact already introduced WSBS files to compensate for this. However, it remains an open question for us how to harness this information dynamically at run-time, instead of asking Restful service

providers to also publish a WSBS file of some kind for this purpose.

Furthermore, in future we would like to study different WS behavioral description schemes to generate the code of WS proxies according to different kinds of input. A possible choice can be various UML diagrams, e.g. activity diagrams or state machines. Finally, we certainly see as a future work to improve the reusability of web services by designing and developing a mechanism for service adaptation mechanism. Adaptation mechanisms are aimed at identifying and resolving of mismatches between service interfaces [119]. While the current service adaptors focus on the interface-level mismatches of services (i.e. focused on syntactical aspects), an adaptation of the behavioral specification of stateful services increases their readability. For example, assume two online purchasing services with similar behavior except that one of them needs an authentication process, i.e. log-in and log-out operations. These two operations add one extra state and two transitions to the web service behavior specification (WSBS). We can match these two WSBSs by merging the extra nodes and transitions, similar to the approach discussed in Section 4.5. We believe that this direction is promising and results would be encouraging.

7.3 Overview and Conclusion

A main goal of this thesis is to present a comprehensive enabling framework for service-oriented collaborative networks in order to increase the efficiency and effectiveness of software service discovery, composition and execution within a network of organizations.

In collaboration networks, software services are simple self-contained applications that perform activities which are triggered by business processes. To support this. web services can be considered as the natural choice for the implementation of Service Oriented Architecture (SOA). They benefit from XML encoded standards, e.g. WSDL and SOAP, which enable web services to become independent of the programming language, operating system, and hardware [53]. Because of this independency that allows computer-to-computer communication in heterogeneous environments, web services are ideally suited to provide dynamic information and functionalities for CNs. In service oriented collaborative networks (SOCNs), e.g. in a VO, the SMEs can modularise their core business into a number of services and reuse them in different business processes. However, to fully achieve the features of SOA, its building blocks, i.e. the services, must be well specified, in order toenable supporting semi-automated service discovery and composition. Currently, the most promising research works in the area of service description are semantic service description frameworks, such as the OWL-S, WSMF, and WSDL-S [53] that provide machine understandable semantic description of services, in order to facilitate the automatic service discovery and composition. However, none of these efforts could effectively achieve this goal

in practice. In Chapter 3, we carried out a comprehensive literature study on the existing web service description standards and found a list of problems that are not adequately addressed. For example, the current web service specifications, even the semantic-based ones, do not sufficiently address the behavioral service information that indicates how a service should be used, and what is the proper configuration of its operations. In Chapter 3, we introduced novel XML-based metadata for business service (BS) competency specification within CNs, providing concrete formal machine readable definitions that improve discovery and composition of services.

As a consequence of insufficient information in the current service specifications, precise matchmaking required to locate a demanded service is also not sufficiently addressed or developed [144]. We have developed a tool for similarity-based discovery of web services matching the user requests. The tool is aimed to rank the identified registered services, according to their similarity/fitting score is with the query presented by the user. The main formalism behind this similarity matching is constraint automata definition, which represents the service behavior, both for registered services and the requested stateful software service. The general approach for the semiring-based search engine of this tool relies on the solution of Soft Constraint Satisfaction Problems (SCSPs), which allows using efficient AI solving techniques for search problem. Furthermore, for identifying the most-fit service(s) among those matched for their functional similarity, the user preferences on the criteria of service quality, such as the availability, and reliability, are considered. In order to combine the functional requirements and the QoS constraints into a single semiring for match-making of services, we define the lexicographic product of semirings. Therefore, the proposed tool efficiently discovers and recommends the most-fit software service(s) to the authorized users, among all those services shared within the VBE environment. In Chapter 4, we demonstrate how the proposed framework improves service discovery. Furthermore, the proposed service search engine tool produces results that measure well against reasonable expectations, as shown in Section 6.2. Moreover, the GloNet project [143] as an application demonstrates the real case application and benefit gained from this tool.

As software services are functional units, they must interact with the external environment, such as other services [53]. There is usually a group of services that a given service can interact with, in order to form a new composite service targeted to achieve certain tasks. If this potential ability of services is ignored, The true essence of service-oriented computing is not realized [76] and [4]. Despite all the efforts spent on composition of services, the current work has inadequately addressed interrelationships among the constituents of a composite service, as well as its automated execution [83]. In other words, the current service description frameworks are unsuitable and incomplete to support automated (or even semi-automated) service composition and execution. To address the above issues, we developed ProxCG that uses Reo circuits as the coordination model to both

compose services and to coordinate complex process interaction protocols. These two issues are among the most challenging for modeling and executing composite BSs and their BPs in CNs. The proxies generated by ProxCG work as wrappers to connect real web services to Reo circuits and thereby support the integration of services.

Chapter 8

Annex I

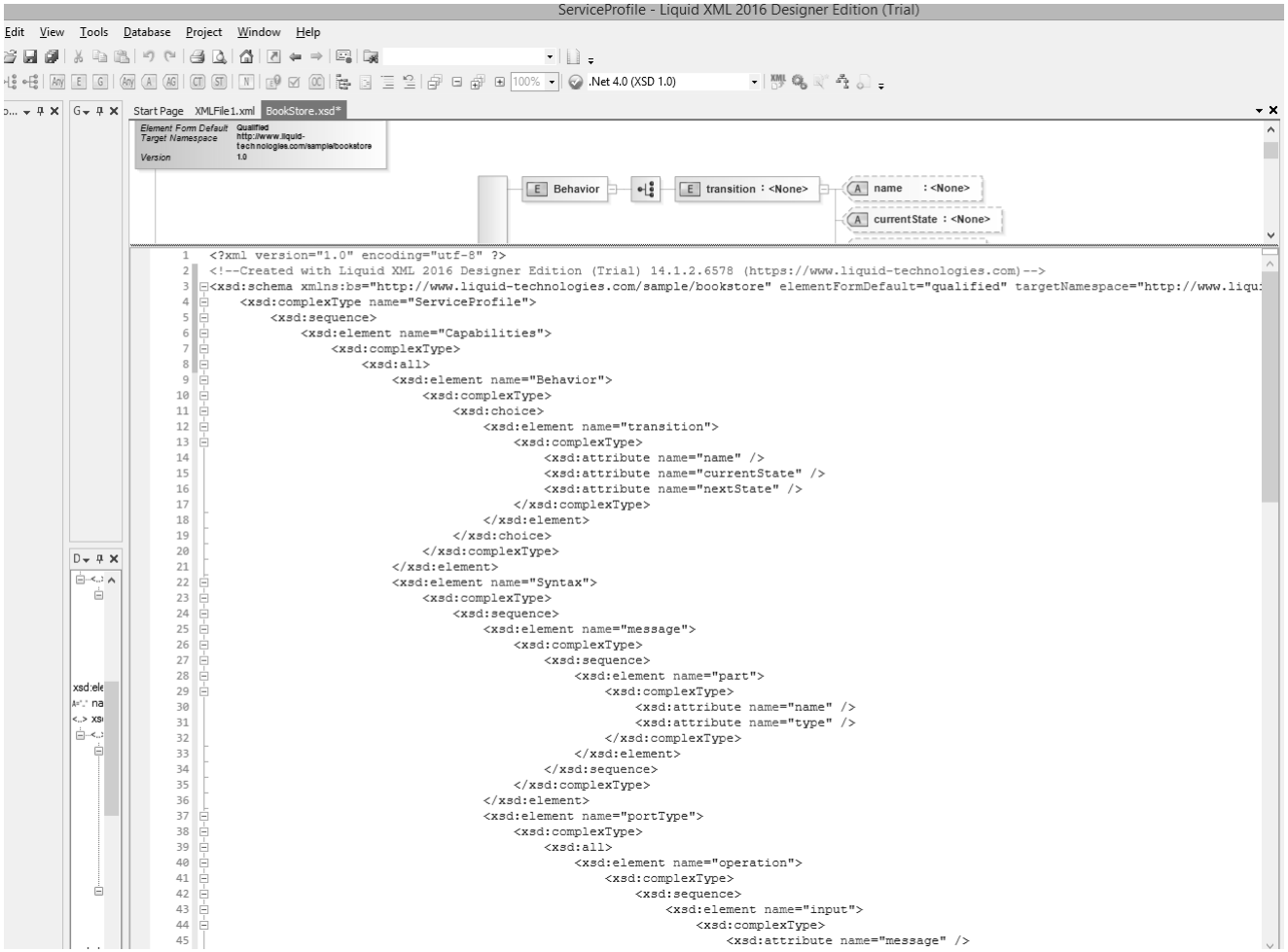


Figure 8.1: The XSD tags of the schema.

Summary

8.1 Summary

In today's economy, collaboration and co-development among organizations has evolved from the traditional format of static supply chains to the dynamic formation of federated organization networks. Networking among business partners has proven to yield lower costs, higher quality, larger service/product portfolio, faster delivery, and more agility. However, the pace by which these trend changes need to occur has raised the demand on ICT-enhanced support for inter-organization collaboration, establishing the area of collaborative networks (CNs). We particularly address in our research two forms of CNs, namely the VO (Virtual Organization) and the VBE (VO Breeding Environment).

To enrich collaboration among potentially distributed partners, a challenging area for CN research is focused on new approaches to specification, sharing and integration of organizations business processes. Aiming to address this challenge, in our research the promising paradigm of Service Oriented Architecture (SOA) is investigated and applied as the base for enhancement of organizations collaboration. However, we argue that the current implementations of the SOA approach do not sufficiently support CN requirements and do not deliver the expected advantages for organizations sharing and integrating their business services. We therefore introduce a new reference framework - the service orientated collaborative networks (SOCN), as a customization of the traditional architecture and generic model of SOA, in order to efficiently support service oriented CNs.

An implementation architecture is also elaborated that addresses our identified requirements for the reference framework, and captures the needed elements and features for establishing the SOCN. Significant sub-components of the reference framework and their inter-relationships are introduced, as an extended SOA model. The architecture seeks to introduce common service semantics and a novel service behavior model, which can be used unambiguously across and between different implementation options. In this PhD work however, we have developed a

proof of concept (POC) for our approach, which employs particular architectures, standards, and technologies. The implementation details that realize our SOCN architecture are provided. The developed service-oriented architecture is conceptually composed of three main software modules: Specification Module, Discovery Module, and Composition Module, as briefly described below.

- **SOCN Specification Module** deals with the specification of software services that are offered by different members/stakeholders in the role of service providers within a VO. Such shared services in the VO are published in a service registry or directory, complying to some agreements, such as the SLA and OLA defined at the VBE level. In this module, we present an extension and improvement to the current web service description approaches and standards, in order to support more efficient service discovery and composition in VOs. First, we depict a data model namely C3Q (addressing the Capability, Costs, Conspicuities, and Quality criteria of services) to represent the various information needed for the description of services. C3Q is considered as the services competency model within the VOs. Then, we introduce a light extension of WSDL that we call XWSDL, to specify web services according to the C3Q model. XWSDL provides a comprehensive description of capabilities of web services and particularly highlights the important role of service behavior in the realization of the semi-automated service oriented computing. We have also developed a user-friendly GUI, assisting service designers to correctly describe and visualize the behavioral specification of their services.

- **SOCN Discovery Module** of the framework provides mechanisms for efficient and accurate discovery and selection of the best-fit service among the existing shared services in the VO. Successful automated application of search-result services of this module requires the description provided through the Specification Module. We have presented a tool for similarity-based discovery of web services that is able to rank service descriptions in our registry, in accordance with a similarity score matching each registry entry with the description of a service desired by a user. The tool is based on implementing approximate operational-similarity evaluation with constraints, which allows to quantitatively estimate the differences between two behaviors. Defining this problem as an SCSP (Soft Constraint Satisfaction Problem) makes it parametric with respect to the chosen similarity metric (i.e. a semiring), and allows using efficient AI techniques for solving it.

- **SOCN Composition Module** of the abstract framework involves the functions introduced to support service composition. Efficient service composition to create new value-added services in the VO requires not only considering the rich meta-data captured in the service specification module,

but also the modeling of the intended coordination among the component services that form a composite service. While orchestration and choreography are two alternative approaches to handle such coordination concerns of the composition, we advocate orchestration for the VO service composition using the coordination language Reo. We have further developed a tool that automatically translates the orchestrators (i.e. the Reo connectors) into Java code, and creates a proxy for each involved component service. This proxy component is in charge of managing the communication between the technology behind the web service and the Reo environment.

Publication List

- Sargolzaei, M., Santini, F., Arbab, F., and Afsarmanesh, H. QoS-aware and Behavior-based Approximate Matching of Stateful Web Services. Submitted to International Journal of Internet Services and Applications.
- Sargolzaei, M. and Afsarmanesh, H., 2017. C3q: A specification model for web services within virtual organizations. In Collaboration in a Data-Rich World. Springer Berlin Heidelberg.
- Sargolzaei, M. and Afsarmanesh, H., 2017. Service oriented collaborative network architecture. In Collaboration in a Data-Rich World. Springer Berlin Heidelberg.
- Sargolzaei, M., Shafahi, M., Afsarmanesh, H., Camarinha-Matos, L., and Thamburaj, V., 2014. D4.4 prototype of the system for enhanced services recommendation. In Glonet WP4.
- Sargolzaei, M., Santini, F., Arbab, F., and Afsarmanesh, H., 2013. A tool for behaviour-based discovery of approximately matching web services. In Software Engineering and Formal Methods, (pp. 152-166). Springer Berlin Heidelberg.
- Afsarmanesh, H., Sargolzaei, M. and Shadi, M., 2015. Semi-automated software service integration in virtual organisations. Enterprise Information Systems, 9(5-6), pp. 528-555.
- Jongmans, S.-S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. , 2014. Orchestrating web services using Reo: from circuits and behaviors to automatically generated code. Service Oriented Computing and Applications, 8(4), pp. 277-297.
- Shafahi, M., Afsarmanesh, H., and Sargolzaei, M., 2014. A coopetition space for complex product specification. In Working Conference on Virtual Enterprises (pp. 83-97). Springer Berlin Heidelberg.

- Afsarmanesh, H., Sargolzaei, M. and Shadi, M., 2012. A framework for automated service composition in collaborative networks. In Collaborative Networks in the Internet of Services (pp. 63-73). Springer Berlin Heidelberg.

- Jongmans, S.-S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. , 2012. Automatic code generation for the orchestration of web services with Reo. In European Conference on Service-Oriented and Cloud Computing, (pp. 1-16). Springer Berlin Heidelberg.

Bibliography

[1] Afsarmanesh, H. and Camarinha-Matos, L. M. (2005). A framework for management of virtual organization breeding environments. In *Working Conference on Virtual Enterprises*, pages 35–48. Springer.

[2] Afsarmanesh, H., Sargolzaei, M., others, and Shafahi, M. (2014). D4.3 report on dynamically customizable services enhancing complex products.

[3] Afsarmanesh, H., Sargolzaei, M., and Shadi, M. (2012). A framework for automated service composition in collaborative networks. In *Working Conference on Virtual Enterprises*, pages 63–73. Springer.

[4] Afsarmanesh, H., Sargolzaei, M., and Shadi, M. (2015). Semi-automated software service integration in virtual organisations. *Enterprise Information Systems*, 9(5-6):528–555.

[5] Afsarmanesh, H., Shafahi, M., Sargolzaei, M., et al. (2013). D4. 1 design report on approach and mechanism for effective customized complex product specification.

[6] Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics-wsdl-s. *International Business Machines Corporation and University of Georgia*.

[7] Al Ridhawi, Y. and Karmouch, A. (2015). Qos-based composition of service specific overlay networks. *IEEE Transactions on Computers*, 64(3):832–846.

[8] Alonso, G. and Casati, F. (2004). *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer.

[9] Arbab, F. (2004). Reo: a channel-based coordination model for component composition. *Mathematical structures in computer science*, 14(03):329–366.

[10] Arbab, F. (2006). Composition of interacting computations. In *Interactive computation*, pages 277–321. Springer.

[11] Arbab, F. (2011). Puff, the magic protocol. In *Formal Modeling: Actors, Open Systems, Biological Systems*, pages 169–206. Springer.

[12] Arbab, F. (2016). Proper protocol. In *Theory and Practice of Formal Methods*, pages 65–87. Springer.

[13] Arbab, F. and Mavaddat, F. (2002). Coordination through channel composition. In *International Conference on Coordination Languages and Models*, pages 22–39. Springer.

[14] Arbab, F. and Rutten, J. (2002). A coinductive calculus of component connectors. In *Recent Trends in Algebraic Development Techniques (WADT) Revised Selected Papers*, volume 2755 of *LNCS*, pages 34–55. Springer.

[15] Arbab, F. and Santini, F. (2012). Preference and similarity-based behavioral discovery of services. In ter Beek, M. H. and Lohmann, N., editors, *WS-FM*, volume 7843 of *Lecture Notes in Computer Science*, pages 118–133. Springer.

[16] Arbab, F., Santini, F., Bistarelli, S., and Pirolandi, D. (2012). Towards a similarity-based web service discovery through soft constraint satisfaction problems. In *Proceedings of the 2nd International Workshop on Semantic Search over the Web, Istanbul, Turkey, August 27, 2012*, page 2. ACM.

[17] Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., et al. (2002). Web service choreography interface (wsci) 1.0, 2002. *URL http://www. w3. org/TR/wsci*.

[18] Baier, C., Sirjani, M., Arbab, F., and Rutten, J. (2006). Modeling component connectors in reo by constraint automata. *Science of computer programming*, 61(2):75–113.

[19] Baresi, L., Miraz, M., and Plebani, P. (2016). A distributed architecture for efficient web service discovery. *Service Oriented Computing and Applications*, 10(1):1–17.

[20] Baruch, Y. (1995). Business globalization–the human resource management aspect. *Human Systems Management*, 14(4):313–326.

[21] Benatallah, B., Sheng, Q. Z., and Dumas, M. (2003). The self-serv environment for web services composition. *IEEE internet computing*, 7(1):40–48.

[22] Benbernou, S., Canaud, E., and Pimont, S. (2004). Semantic web services discovery regarded as a constraint satisfaction problem. In *Flexible Query Answering Systems, 6th International Conference*, volume 3055 of *LNCS*, pages 282–294. Springer.

[23] Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. (2007). Wsqosx–a qos architecture for web service workflows. In *International Conference on Service-Oriented Computing*, pages 623–624. Springer.

[24] Bistarelli, S., Montanari, U., and Rossi, F. (1997). Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236.

[25] Boreale, M., Bruni, R., De Nicola, R., and Loreti, M. (2008). Sessions and pipelines for structured service programming. In *International Conference on Formal Methods for Open Object-Based Distributed Systems*, pages 19–38. Springer.

[26] Börger, E. (2012). Approaches to modeling business processes: a critical analysis of bpmn, workflow patterns and yawl. *Software & Systems Modeling*, 11(3):305–318.

[27] Brafman, R. and Domshlak, C. (2009). Preference handling-an introductory tutorial. *AI magazine*, 30(1):58.

[28] Brogi, A. and Popescu, R. (2006). Automated generation of bpel adapters. In *International Conference on Service-Oriented Computing*, pages 27–39. Springer.

[29] Camarinha-Matos, L. and Afsarmanesh, H. (2011). Behavioral aspects in collaborative enterprise networks. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 12–19. IEEE.

[30] Camarinha-Matos, L. and Afsarmanesh, H. (2012). Taxonomy of collaborative networks forms. *Final document of the FInES Task Force on Collaborative Networks and SOCOLNET-Society of Collaborative Networks*.

[31] Camarinha-Matos, L. M., Afsarmanesh, H., et al. (2006). Collaborative networks: Value creation in a knowledge society. *Knowledge enterprise, IFIP*, 207:26–40.

[32] Camarinha-Matos, L. M., Afsarmanesh, H., Galeano, N., and Molina, A. (2009). Collaborative networked organizations–concepts and practice in manufacturing enterprises. *Computers & Industrial Engineering*, 57(1):46–60.

[33] Camarinha-Matos, L. M., Afsarmanesh, H., Oliveira, A.-I., Ferrada, F., et al. (2013a). Collaborative business services provision. In *ICEIS (2)*, pages 380–390.

[34] Camarinha-Matos, L. M., Juan-Verdejo, A., Alexakis, S., Bär, H., and Surajbali, B. (2015a). Cloud-based collaboration spaces for enterprise networks. In *Computing and Communications Technologies (ICCCT), 2015 International Conference on*, pages 185–190. IEEE.

[35] Camarinha-Matos, L. M., Macedo, P., Sargolzaei, M., and Afsarmanesh, H. (2013b). D2.4 mechanisms for defining composed services to support collaboration.

[36] Camarinha-Matos, L. M., Oliveira, A. I., and Ferrada, F. (2015b). Supporting collaborative networks for complex service-enhanced products. In *Working Conference on Virtual Enterprises*, pages 181–192. Springer.

[37] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., and Shan, M.-C. (2000). Adaptive and dynamic service composition in eflow. In *International Conference on Advanced Information Systems Engineering*, pages 13–31. Springer.

[38] Cesari, L., Pugliese, R., and Tiezzi, F. (2010). A tool for rapid development of ws-bpel applications. *ACM SIGAPP Applied Computing Review*, 11(1):27–40.

[39] Changizi, B., Kokash, N., and Arbab, F. (2010). A Unified Toolset for Business Process Model Formalization. In *Proceedings of FESCA 2010*.

[40] Cheatham, M. and Hitzler, P. (2013). String similarity metrics for ontology alignment. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference*, volume 8219 of *Lecture Notes in Computer Science*, pages 294–309. Springer.

[41] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1: Core language. *W3C recommendation*, 26:19.

[42] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93.

[43] D'Ambrogio, A. (2006). A model-driven wsdl extension for describing the qos ofweb services. In *Web Services, 2006. ICWS'06. International Conference on*, pages 789–796. IEEE.

[44] Decker, G., Kopp, O., Leymann, F., Pfitzner, K., and Weske, M. (2008). Modeling service choreographies using bpmn and bpel4chor. In *International conference on Advanced Information Systems Engineering*, pages 79–93. Springer.

[45] Dekkers, R. (2010). A co-evolutionary perspective on distributed manufacturing. In *Distributed Manufacturing*, pages 29–50. Springer.

[46] Dezani-Ciancaglini, M. and DeLiguoro, U. (2010). Sessions and session types: an overview. In *Web Services and Formal Methods*, pages 1–28. Springer.

[47] Dezani-Ciancaglini, M., Padovani, L., and Pantovic, J. (2014). Session type isomorphisms. In *PLACES*, pages 61–71.

[48] Dhara, K. M., Dharmala, M., and Sharma, C. K. (2015). A survey paper on service oriented architecture approach and modern web services. *All Capstone Projects*, (157).

[49] Di Modica, G., Tomarchio, O., and Vita, L. (2011). Resource and service discovery in soas: A p2p oriented semantic approach. *International Journal of Applied Mathematics and Computer Science*, 21(2):285–294.

[50] Doedt, M. and Steffen, B. (2012). An evaluation of service integration approaches of business process management systems. In *Software Engineering Workshop (SEW), 2012 35th Annual IEEE*, pages 158–167. IEEE.

[51] Dong, X., Halevy, A., Madhavan, J., Nemes, E., and Zhang, J. (2004). Similarity search for web services. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 372–383. VLDB Endowment.

[52] Droste, M., Kuich, W., and Vogler, H. (2009). *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition.

[53] Du, X. (2009). *Semantic service description framework for efficient service discovery and composition*. PhD thesis, Durham University.

[54] El Hadad, J., Manouvrier, M., and Rukoz, M. (2010). Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. *IEEE Transactions on Services Computing*, 3(1):73–85.

[55] Engler, L. (2009). *BPEL gold: Choreography on the Service Bus*. PhD thesis, Stuttgart, Universität Stuttgart, Diplomarbeit.

[56] Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India.

[57] Ermilova, E. and Afsarmanesh, H. (2007). Modeling and management of profiles and competencies in vbes. *Journal of Intelligent Manufacturing*, 18(5):561–586.

[58] Fensel, D. and Bussler, C. (2002). The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137.

[59] Florescu, D., Grünhagen, A., and Kossmann, D. (2003). Xl: An xml programming language for web service specification and composition. *Computer Networks*, 42(5):641–660.

[60] Fournet, C. and Gonthier, G. (2002). The join calculus: a language for distributed mobile programming. In *Applied Semantics*, pages 268–332. Springer.

[61] Fox, E. A. and Shaw, J. A. (1994). Combination of multiple searches. *NIST SPECIAL PUBLICATION SP*, pages 243–243.

[62] Frankel, D. and Parodi, J. (2002). Using model-driven architecture to develop web services. *IONA Technologies white paper*.

[63] Gadducci, F., Hölzl, M., Monreale, G. V., and Wirsing, M. (2013). Soft constraints for lexicographic orders. In *Mexican International Conference on Artificial Intelligence*, pages 68–79. Springer.

[64] Gadducci, F. and Santini, F. (2017). Residuation for bipolar preferences in soft constraints. *Inf. Process. Lett.*, 118:69–74.

[65] Gay, S., Soliman, S., and Fages, F. (2010). A graphical method for reducing and relating models in systems biology. *Bioinformatics*, 26(18).

[66] Gelernter, D. and Carriero, N. (1992). Coordination languages and their significance. *Communications of the ACM*, 35(2):96.

[67] Grigori, D., Corrales, J. C., and Bouzeghoub, M. (2006). Behavioral matchmaking for service retrieval. In *IEEE International Conference on Web Services (ICWS)*, pages 145–152. IEEE Computer Society.

[68] Group, W. S. C. W. et al. (2002). Web services choreography description language.

[69] Gu, X., Wichadakul, D., and Narhstedt, K. (2001). Visual qos programming environment for ubiquitous multimedia services. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 575–578. IEEE.

[70] Guidara, I., Chaari, T., Fakhfakh, K., and Jmaiel, M. (2012). A comprehensive survey on intra and inter organizational agreements. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pages 411–416. IEEE.

[71] Hao, Y., Zhang, Y., and Cao, J. (2007). Wsxplorer: Searching for desired web services. In *International Conference on Advanced Information Systems Engineering*, pages 173–187. Springer.

[72] Hau, J., Lee, W., and Darlington, J. (2005). A semantic similarity measure for semantic web services. In *Web Service Semantics Workshop at WWW*.

[73] Hausmann, J. H., Heckel, R., and Lohmann, M. (2005). Model-based development of web service descriptions enabling a precise matching concept. *International Journal of Web Services Research*, 2(2):67.

[74] Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81.

[75] Hull, R., Benedikt, M., Christophides, V., and Su, J. (2003). E-services: a look behind the curtain. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–14. ACM.

[76] Huma, Z., Gerth, C., Engels, G., and Juwig, O. (2013). Automated service discovery and composition for on-the-fly soas. Technical report, Tech. Rep. TR-RI-13-333, University of Paderborn, Germany. http://is. uni-paderborn. de/uploads/tx_sibibtex/tr-ri-13-333. pdf.

[77] Iordache, R. and Moldoveanu, F. (2012). A conditional lexicographic approach for the elicitation of qos preferences. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 182–193. Springer.

[78] Iosup, A., Sonmez, O., Anoep, S., and Epema, D. (2008). The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing*, pages 97–108. ACM.

[79] Järvelin, K. and Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.

[80] Jayasinghe, D. and Azeez, A. (2011). *Apache Axis2 Web Services*. Packt Publishing.

[81] Jerstad, I., Dustdar, S., and Thanh, D. V. (2005). A service oriented architecture framework for collaborative services. In *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, pages 121–125. IEEE.

[82] Jongmans, S.-S. T. and Arbab, F. (2012). Overview of thirty semantic formalisms for reo. *Sci. Ann. Comp. Sci.*, 22(1):201–251.

[83] Jongmans, S.-S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. (2012). Automatic code generation for the orchestration of web services with reo. In *European Conference on Service-Oriented and Cloud Computing*, pages 1–16. Springer.

[84] Jongmans, S.-S. T., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H. (2014). Orchestrating web services using reo: from circuits and behaviors to automatically generated code. *Service Oriented Computing and Applications*, 8(4):277–297.

[85] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al. (2007). Web services business process execution language version 2.0. *OASIS standard*, 11(120):5.

[86] Juric, M. B. (2006). A hands-on introduction to bpel. *Oracle (white paper)*, page 21.

[87] Koehler, C., Costa, D., Proença, J., and Arbab, F. (2008). Reconfiguration of reo connectors triggered by dataflow. *Electronic Communications of the EASST*, 10.

[88] Koehler, C. and Maraikar, Z. (2008). Eclipse coordination tools user manual.

[89] Kokash, N. and Arbab, F. (2008). Formal behavioral modeling and compliance analysis for service-oriented systems. In Frank S.de Boer, Bonsangue, M. M., and Madelain, E., editors, *FMCO*, volume 5751 of *Lecture Notes in Computer Science*, pages 21–41. Springer.

[90] Kokash, N. and Arbab, F. (2013). Formal design and verification of long-running transactions with extensible coordination tools. *IEEE T. Services Computing*, 6(2):186–200.

[91] Kokash, N., Arbab, F., Changizi, B., and Makhnist, L. (2011). Input-output conformance testing for channel-based service connectors. In Aceto, L. and Mousavi, M. R., editors, *PACO*, volume 60 of *EPTCS*, pages 19–35.

[92] Kokash, N., Krause, C., and de Vink, E. (2010). Data-aware design and verification of service compositions with reo and mcrl2. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 2406–2413. ACM.

[93] Kokash, N., Krause, C., and de Vink, E. (2012). Reo+ mcrl2: A framework for model-checking dataflow in service compositions. *Formal Aspects of Computing*, 24(2):187–216.

[94] Kopecky, J., Gomadam, K., and Vitvar, T. (2008). hrests: An html microformat for describing restful web services. In *Web Intelligence and Intelligent*

Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on, volume 1, pages 619–625. IEEE.

[95] Kopeckỳ, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6).

[96] Kopp, O., Engler, L., and van Lessen, T. (2010). Interaction choreography models in bpel. In *S-BPM ONE 2010-the Subjectoriented BPM Conference (CCIS)*, volume 138. Springer-Verlag.

[97] Kopp, O., Leymann, F., and Wagner, S. (2011). Modeling choreographies: Bpmn 2.0 versus bpel-based approaches. In *EMISA*, pages 225–230.

[98] Krause, C. et al. (2011). *Reconfigurable component connectors*. PhD thesis, Phd thesis, University of Leiden.

[99] Kubovy, J., Geist, V., and Kossak, F. (2012). A formal description of the itil change management process using abstract state machines. In *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, pages 65–69. IEEE.

[100] Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004). A conceptual comparison of wsmo and owl-s. In *Web services*, pages 254–269. Springer.

[101] Lemos, A. L., Daniel, F., and Benatallah, B. (2016). Web service composition: a survey of techniques and tools. *ACM Computing Surveys (CSUR)*, 48(3):33.

[102] Li, G., Han, Y., Zhao, Z., Wang, J., and Wagner, R. M. (2006). Facilitating dynamic service compositions by adaptable service connectors. *International Journal of Web Services Research*, 3(1):67–83.

[103] Ligeza, A. and Potempa, T. (2014). A] approach to formal analysis of bpmn models: Towards a logical model for bpmn diagrams. In *Advances in Business ICT*, pages 69–88. Springer.

[104] Ludwig, H. (2003). Web services qos: external slas and internal policies or: how do we deliver what we promise? In *Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on*, pages 115–120. IEEE.

[105] Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. (2003). Web service level agreement (wsla) language specification. *IBM Corporation*, pages 815–824.

[106] MacKenzie, M. C., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12:18.

[107] Maedche, A., Staab, S., et al. (2003). *Services on the move: towards P2P-enabled Semantic Web services*. na.

[108] Mallayya, D., Ramachandran, B., and Viswanathan, S. (2015). An automatic web service composition framework using qos-based web service ranking algorithm. *The Scientific World Journal*, 2015.

[109] Mariotti, M., Manzini, P., et al. (2012). Choice by lexicographic semiorders. *Theoretical Economics*, 7(1).

[110] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with owl-s. *World Wide Web*, 10(3):243–277.

[111] McCool, M., Du Toit, S., Popa, T., Chan, B., and Moule, K. (2004). Shader algebra. *ACM Transactions on Graphics (TOG)*, 23(3):787–795.

[112] Mellor, S. J. (2004). *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional.

[113] Menasce, D. (2002). QoS issues in web services. *Internet Computing, IEEE*, 6(6):72–75.

[114] Menasce, D. A. (2004). Composing web services: A qos view. *IEEE Internet computing*, 8(6):88–90.

[115] Meng, S. and Arbab, F. (2007). Web services choreography and orchestration in reo and constraint automata. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 346–353. Acm.

[116] Meng, S. and Arbab, F. (2009). Qos-driven service selection and composition using quantitative constraint automata. *Fundamenta Informaticae*, 95(1):103–128.

[117] Meng, S. and Arbab, F. (2010). A model for web service coordination in long-running transactions. In *SOSE*, pages 121–128. IEEE.

[118] Montesi, F., Guidi, C., Lucchi, R., and Zavattaro, G. (2007). Jolie: a java orchestration language interpreter engine. *Electronic Notes in Theoretical Computer Science*, 181:19–33.

[119] Motahari Nezhad, H. R., Benatallah, B., Martens, A., Curbera, F., and Casati, F. (2007). Semi-automated adaptation of service interactions. In *Proceedings of the 16th international conference on World Wide Web*, pages 993–1002. ACM.

[120] Mousavi, M. R., Sirjani, M., and Arbab, F. (2006). Formal semantics and analysis of component connectors in reo. *Electronic Notes in Theoretical Computer Science*, 154(1):83–99.

[121] Msanjila, S. and Afsarmanesh, H. (2008). Trust analysis and assessment in virtual organization breeding environments. *International Journal of Production Research*, 46(5):1253–1295.

[122] Nie, P., Seppälä, R., and Hafrén, M. (2007). Open source power on bpm-a comparison of jboss jbpm and intalio bpms. *Special Course in Information Systems Integration (2007): Business Process Integration*, pages 1–26.

[123] Ouyang, C., Verbeek, E., Van Der Aalst, W. M., Breutel, S., Dumas, M., and Ter Hofstede, A. H. (2007). Formal semantics and analysis of control flow in ws-bpel. *Science of computer programming*, 67(2-3):162–198.

[124] Pan, J. Z., Stamou, G., Stoilos, G., Taylor, S., and Thomas, E. (2008). Scalable querying services over fuzzy ontologies. In *Proceedings of World Wide Web*, WWW '08, pages 575–584. ACM.

[125] Papadopoulos, G. A. and Arbab, F. (1998). Coordination models and languages. *Advances in computers*, 46:329–400.

[126] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255.

[127] Pautasso, C. (2009). Restful web service composition with bpel for rest. *Data & Knowledge Engineering*, 68(9):851–866.

[128] Pei, S. and Chen, D. (2011). Research on dynamic web services composition framework based on quality of service. *Information Technology Journal*, 10(8):1645–1649.

[129] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

[130] Petritsch, H. (2006). Service-oriented architecture (soa) vs. component based architecture. *Vienna University of Technology, Vienna*.

[131] Pfleeger, S. L. and Atlee, J. M. (1998). *Software engineering: theory and practice*. Pearson Education India.

[132] Pistore, M., Traverso, P., Bertoli, P., and Marconi, A. (2005). Automated synthesis of composite bpel4ws web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 293–301. IEEE.

[133] Plebani, P. and Pernici, B. (2009). Urbe: Web service retrieval based on similarity evaluation. *IEEE Trans. on Knowl. and Data Eng.*, 21(11):1629–1642.

[134] Ponnekanti, S. R. and Fox, A. (2002). Sword: A developer toolkit for web service composition. In *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*, volume 45.

[135] Prakash, J., Rohini, M., Ganesh, B., and Maheswari, V. (2013). Hybrid reliability model to enhance the efficiency of composite web services. In *Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), 2013 International Conference on*, pages 79–83. IEEE.

[136] Rohallah, B., Ramdane, M., and Zaidi, S. (2013). Agents and owl-s based semantic web service discovery with user preference support. *arXiv preprint arXiv:1306.1478*.

[137] Rossi, F., P. van Beek, and Walsh, T. (2006a). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA.

[138] Rossi, F., Van Beek, P., and Walsh, T. (2006b). *Handbook of constraint programming*. Elsevier.

[139] Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer.

[140] Saint-Marcq, V., Deville, Y., and Solnon, C. (2009). Constraint-based graph matching. In *CP*, pages 274–288.

[141] Sargolzaei, M. and Afsarmanesh, H. (2017a). C3q: A specification model for web services within virtual organizations. In *Working Conference on Collaboration in a Data-Rich World*. Springer.

[142] Sargolzaei, M. and Afsarmanesh, H. (2017b). Service oriented collaborative network architecture. In *Working Conference on Collaboration in a Data-Rich World*. Springer.

[143] Sargolzaei, M., Afsarmanesh, H., and Shafahi, M. (2014a). D4.4 prototype of the system for enhanced services recommendation.

[144] Sargolzaei, M., Santini, F., Arbab, F., and Afsarmanesh, H. (2013). A tool for behaviour-based discovery of approximately matching web services. In *International Conference on Software Engineering and Formal Methods*, pages 152–166. Springer.

[145] Sargolzaei, M., Shafahi, M., Afsarmanesh, H., Camarinha-Matos, L., and Thamburaj, V. (2014b). Glonet wp4: Customized service-enhanced product specification.

[146] Saxe, J. G. and Schwartzott, C. (1994). The blind men and the elephant.

[147] Scholten, G. (2007). *Mobile channels for exogenous coordination of distributed systems: semantics, implementation and composition.* PhD thesis, Phd thesis, University of Leiden.

[148] Shadi, M. and Afsarmanesh, H. (2013). Behavior modeling in virtual organizations. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 50–55. IEEE.

[149] Shadi, M. and Afsarmanesh, H. (2014). Behavioral norms in virtual organizations. In *Working Conference on Virtual Enterprises*, pages 48–59. Springer.

[150] Shadi, M., Afsarmanesh, H., and Dastani, M. (2013a). Agent behaviour monitoring in virtual organizations. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, pages 9–14. IEEE.

[151] Shadi, M., Afsarmanesh, H., and Dastani, M. (2013b). Agent behaviour monitoring in virtual organizations. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, pages 9–14. IEEE.

[152] Shafahi, M., Afsarmanesh, H., and Sargolzaei, M. (2014). A coopetition space for complex product specification. In *Working Conference on Virtual Enterprises*, pages 83–97. Springer.

[153] Shen, Z. and Su, J. (2005). Web service discovery based on behavior signatures. In *Proceedings of the 2005 IEEE International Conference on Services Computing - Volume 01*, SCC '05, pages 279–286, Washington, DC, USA. IEEE Computer Society.

[154] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decades overview. *Information Sciences*, 280:218–238.

[155] Tabatabaei, S. G. H., Kadir, W. M. N. W., Ibrahim, S., Dastjerdi, A. V., et al. (2009). Integrating discovery and composition of semantic web services based on description logic.

[156] Talcott, C., Sirjani, M., and Ren, S. (2011). Comparing three coordination models: Reo, arc, and pbrd. *Science of Computer Programming*, 76(1):3–22.

[157] Tasharofi, S., Vakilian, M., Moghaddam, R. Z., and Sirjani, M. (2007). Modeling web service interactions using the coordination language reo. In *International Workshop on Web Services and Formal Methods*, pages 108–123. Springer.

[158] Terlouw, L. I. and Albani, A. (2013). An enterprise ontology-based approach to service specification. *IEEE Transactions on Services Computing*, 6(1):89–101.

[159] Thißen, D. and Wesnarat, P. (2006). Considering qos aspects in web service composition. In *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on*, pages 371–377. IEEE.

[160] Toch, E., Gal, A., Reinhartz-Berger, I., and Dori, D. (2007). A semantic approach to approximate service retrieval. *ACM Trans. Internet Technol.*, 8(1).

[161] Ullman, J. D. (1982). *A first course in database systems*. Pearson Education India.

[162] Van Der Aalst, W. and Ter Hofstede, A. (2005). Yawl: yet another workflow language. *Information systems*, 30(4):245–275.

[163] Van Der Aalst, W. M., Aldred, L., Dumas, M., and ter Hofstede, A. H. (2004). Design and implementation of the yawl system. In *International Conference on Advanced Information Systems Engineering*, pages 142–159. Springer.

[164] van der Aalst, W. M., Dumas, M., ter Hofstede, A., and Wohed, P. (2002). Pattern-based analysis of bpml (and wsci). Technical report, Citeseer.

[165] Vu, L.-H., Hauswirth, M., Porto, F., and Aberer, K. (2006). A search engine for qos-enabled discovery of semantic web services. *International Journal of Business Process Integration and Management*, 1(4):244–255.

[166] Wei, D., Wang, T., Wang, J., and Bernstein, A. (2011). Sawsdl-imatcher: A customizable and effective semantic web service matchmaker. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):402–417.

[167] Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., and Russell, N. (2006). On the suitability of bpmn for business process modelling. In *International conference on business process management*, pages 161–176. Springer.

[168] Yu, T. and Lin, K.-J. (2005). Service selection algorithms for composing complex services with multiple qos constraints. In *International Conference on Service-Oriented Computing*, pages 130–143. Springer.

[169] Zemni, M. A., Benbernou, S., and Carro, M. (2010). A soft constraint-based approach to QoS-aware service selection. In *Service-Oriented Computing - 8th International Conference, ICSOC 2010*, volume 6470 of *LNCS*, pages 596–602.

[170] Zhang, J., Chang, C. K., Chung, J.-Y., and Kim, S. W. (2004). Ws-net: A petri-net based specification model for web services. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 420–427. IEEE.

[171] Zimmer, P., Zimmer, M., and Zimmer, B. (2014). Fizzim an open-source fsm design environment. *Enterprise Information Systems*, 9(5-6):528–555.

[172] Zisman, A., Dooley, J., and Spanoudakis, G. (2008). Proactive runtime service discovery. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 1*, SCC '08, pages 237–245, Washington, DC, USA. IEEE Computer Society.

[173] Zoeteweij, P. and Arbab, F. (2004). A component-based parallel constraint solver. In *International Conference on Coordination Languages and Models*, pages 307–322. Springer.

Samenvatting

In de hedendaagse economie zijn de samenwerking tussen, en de gemeenschappelijk ontwikkeling van, organisaties gevolueerd van traditionele, statische supply chains naar dynamische vorming van organisatienetwerken. Netwerkactiviteiten tussen business partners leiden aantoonbaar tot lagere kosten, hogere kwaliteit, grotere service- en/of productportfolios, snellere oplevering, en sterkere wendbaarheid. Echter, door het tempo waarin deze trendveranderingen plaats dienen te vinden, is er vraag ontstaan naar ICT-ondersteuning voor samenwerking tussen organisaties. Dit leidde tot het vakgebied van "collaborative networks" (CN). In ons onderzoek kijken we in het bijzonder naar twee vormen van CN, namelijk "virtual organizations" (VO) en "VO breeding environments" (VBE).

Een uitdagend deelgebied van onderzoek naar CN richt zicht op nieuwe aanpakken voor het specificeren, het delen, en het integreren van bedrijfsprocessen van organisaties, om samenwerking tussen partners, die mogelijk gedistribueerd zijn, te verbeteren. Binnen dit uitdagende deelgebied, bestuderen we in ons onderzoek het veelbelovende paradigma van "service oriented architecture" (SOA); we gebruiken SOA als een basis om de samenwerking tussen organisaties te verbeteren. We beargumenteren dat de huidige implementaties van de SOA-aanpak de eisen die gesteld worden binnen CN onvoldoende ondersteunen, en niet de verwachte voordelen opleveren voor organisaties die services delen en integreren. We introduceren daarom een nieuw referentiekader voor "service oriented collaborative networks" (SOCN). Dit referentiekader is een verfijning van het traditionele SOA en het algemene SOA-model, om SOCNs op een efficinte manier te ondersteunen.

We bespreken een implementatie-architectuur, die de door ons gedentificeerde eisen aan het referentiekader vervult. Significante onderdelen van het referentiekader en hun onderlinge relaties worden gentroduceerd door middel van een uitgebreid SOA-model. De architectuur poogt een gemeenschappelijke servicesemantiek en een vernieuwend servicegedragsmodel te introduceren, die eenduidig met, en tussen, verschillende implementatiemogelijkheden gebruikt kan worden.

In dit PhD-project hebben we een proof-of-concept voor onze aanpak ontwikkeld, die specifieke architecturen, standaarden, en technologien gebruikt. We bespreken de implementatiedetails die onze SOCN-architectuur realiseren. De ontwikkelde SOA-architectuur bestaat conceptueel uit drie hoofdsoftwaremodules: de Specification Module, de Discovery Module, en de Composition Module. Deze modules worden hieronder beschreven.

De **SOCN Specification Module** behandelt de specificatie van software-services die worden aangeboden door verschillende leden/stakeholders binnen een VO, acterend in de rol van serviceaanbieder. Dergelijke gedeelde services in de VO worden gepubliceerd in een serviceregister binnen een VO; ze voldoen aan bepaalde afspraken, zoals de SLA en OLA, die gedefinieerd zijn op het VBE-niveau. Deze module is gebaseerd op een uitbreiding van, en verbetering aan, de huidige web-service-beschrijvingsaanpakken en -standaarden, om efficintere serviceontdekking en -compositie in VO te ondersteunen. Eerst tonen we een datamodel, namelijk C3Q (Capability, Costs, Conspicuities, en Quality-criteria van services), om de verschillende soorten informatie te representeren die nodig zijn om services te beschrijven. C3Q wordt beschouwd als het competentiemodel voor services binnen de VO. Vervolgens introduceren we een kleine uitbreiding van WSDL, die we XWSDL noemen, om web-services te specificeren volgens het C3Q-model. XWSDL ondersteunt het geven van een alomvattende beschrijving van de mogelijkheden van web-services, en belicht in het bijzonder de belangrijke rol van servicegedrag in het realiseren van het semi-geautomatiseerde "service-oriented computing" (SOC). We hebben ook een gebruiksvriendelijke GUI ontwikkeld, die serviceontwikkelaars helpt bij het correct beschrijven en visualiseren van gedragsspecificaties van hun services.

De **SOCN Discovery Module** van het referentiekader biedt mechanismen voor de efficinte en nauwkeurige "discovery" (ontdekking) en selectie van de meest geschikte service tussen de bestaande gedeelde services in een VO. Om de services die gevonden worden door deze module succesvol op geautomatiseerde wijze in te passen, zijn beschrijvingen uit de Specification Module nodig. We hebben een tool ontwikkeld voor "similarity-based discovery" van web-services, die in staat is om service-beschrijvingen in een serviceregister te rangschikken, in overeenstemming met een "similarity score" waarin elke service in het register wordt vergeleken met een beschrijving van de service waarnaar de gebruiker op zoek is. De tool is gebaseerd op een implementatie van "approximate operational-similarity evaluation with constraints", die het mogelijk maakt om op kwantitatieve wijze het verschil tussen twee gedragingen te benaderen. Door het probleem te definiren als een "soft constraint satisfaction problem" (SCCP), wordt het parametrisch in de gekozen "similarity metric" (i.e., een semiring). Dit maakt het mogelijk om efficinte AI-technieken te gebruiken om tot een oplossing te komen.

De **SOCN Composition Module** van het abstracte referentiekader bevat functies die servicecompositie ondersteunen. Efficinte servicecompositie, voor de creatie van nieuwe services die waarde toevoegen in de VO, vereist dat niet alleen

de rijke meta-data uit de Specification Module dient te worden beschouwd, maar ook een model van de beoogde cordinatie tussen de "component services" die de "composite service" vormen. "Orchestration" en "choreography" zijn twee alternatieve manieren om invulling te geven aan dergelijke cordinatieaspecten van een servicecompositie. Wij verdedigen orchestration voor VO-servicecompositie met de cordinatietaal Reo. We hebben een tool ontwikkeld die automatisch "orchestrators" (i.e., "Reo connectors") vertaalt naar Javacode, en een "proxy" creert voor elke betrokken component service. Deze proxy is verantwoordelijk voor het managen van de communicatie tussen de technologie achter de web-service en de Reo-omgeving.

Acknowledgments

I would like to extend my sincere gratitude and special appreciation to my promoter Prof. Dr. Hamideh Afsarmanesh for her great support and guidance during my PhD study, and being a friend for me. Her patience, and immense knowledge make me confident for doing my research and writing of this thesis.

I would also like to express my special thanks to my second promoter Prof. Dr. Farhad Arbab for all his insightful advices and stimulating discussions during my study. His scientific mindset and influential personality have been always inspiring to me.

An special thanks goes to Dr. Francesco Santini, you have been supporting me throughout my research technically and scientifically. You have been always open to my questions and enthusiastic to share your ideas and insights with me.

I would like to convey my great appreciation to my committee members, Prof. Worring, Prof. Meijer, Prof. Grefen, Prof. Groen, Dr. Dastani, Dr. Xu, and Dr Belloum, for agreeing to be on my committee and critically reading my thesis.

I am thankful to Sung-Shik Jongmans for the stimulating discussions, for the days we were working together, and for his invaluable help to write the Samenvatting section. My special thanks also goes to my colleagues and friends, Amirhossein, Fahimeh, Gerben, Hamidreza, Hodjat, Jafar, Marzieh, Masoud, Mirriam, Mohammad, Naser, and Sijali for all the fun we have had and for their help to shape and develop ideas in my research.

Words are powerless to express my heartfelt gratitude to my lovely parents Fatemeh, and Gholamreza. I am grateful to them for their unconditional love and support.

Lastly, I wish to thank my family. My beloved wife for believing in me and for giving endless support in whatever I did. My angel, Sarina, for the joyful sense of life that she gives me. To them I dedicate this thesis.

Abbreviations

AI	Artificial Intelligence
ARC	Actors-Roles-Coordinators
BehSearch	Behavior-based Search
BP	Business Process
BPEL4WS	BPEL for Web Services
BPMN	Business Process Model and Notation
BS	Business Service
C3Q	Capability, Cost, Conspicuity, and Quality criteria
CA	Constraint Automata
CircCG	Circuit Code Generator
CN	Collaborative Network
CNO	Collaborative Network Organization
COP	Constraint Optimization Problem
CSP	Constraint Satisfaction Problems
ECT	Extensible Coordination Tools
EPC	Engineering, Procurement, and Construction
ER	Entity-Relationship
FSM	Finite State Machine
GUI	Graphical User Interface
I/O	Input/Output
ICT	Internet and Communication Technology
IT	Internet Technology
JaCoP	Java Constraint Programming solver
LTS	Labeled Transition System
MDA	Model Driven Architecture
MOF	Meta Object Facility
OLA	Operational Level Agreement
OMG	Object Management Group
OOP	Object-Oriented Programming

OrchCG	Orchestration Code Generator
OWL-S	Web Ontology Language for Web Services
PBRD	Policy-based Russian Dolls
ProxCG	Proxy Code Generator
QWSDL	QoS-enabled WSDL
R-P	Recall/Precision
PLC	Product Life Cycle
RRD	Reflective Russian Dolls
RQ	Research Question
PSDR	Product/Service Discovery & Recommendation
PSS	Product Service Specification
PST	Product Specification Tool
SCA	Soft Constraint Automata
SCSP	Soft Constraint Satisfaction Problems
SCU	Service Communication Unit
SDC	Soft Data-Constraints
SLA	Service Level Agreements
SLC	Service Life Cycle
SME	Small and Medium-sized Enterprise
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SOCN	Service Oriented Collaborative Network
SST	Service Specification Tool
TDS	Timed Data Streams
TWDS	Timed Weighted Data Streams
UML	Unified Modeling Language
URBE	UDDI Registry By Example
URI	Uniform Resource Identifier
VBE	Virtual Organizations Breeding Environment
VO	Virtual Organization
VOSAT	VO Supervisory Assisting Tool
W3C	World Wide Web Consortium
WF	Workflow Foundation
WS	Web Service
WS-CDL	Web Service Choreography Description Languages
WSBS	Web Service Behaviour Specification
WSCI	Web Service Choreography Interface
WSDL	Web Service Description Language
WSDL-S	WSDL Semantics
WSXplorer	Web Service Explorer
XWSDL	Extended WSDL
YAWL	Yet Another Workflow Language

SIKS Dissertation Series

2018-01 Han van der Aa (VUA), Comparing and Aligning Process Representations .

2018-02 Felix Mannhardt (TUE), Multi-perspective Process Mining .

2018-03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction.

2018-04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks .

2018-05 Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process .

2018-06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems .

2018-07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems .

2018-08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems .

2018-09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations .

2017-01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime.

2017-02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation.

2017-03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines.

2017-04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store.

2017-05 Mahdieh Shadi (UVA), Collaboration Behavior.

2017-06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search.

2017-07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly.

2017-08 Rob Konijn (VU) , Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery.

2017-09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text.

2017-10 Robby van Delden (UT), (Steering) Interactive Play Behavior.

2017-11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment.

2017-12 Sander Leemans (TUE), Robust Process Mining with Guarantees.

2017-13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology.

2017-14 Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior.

2017-15 Peter Berck (RUN), Memory-Based Text Correction.

2017-16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines.

2017-17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution.

2017-18 Ridho Reinanda (UVA), Entity Associations for Search.

2017-19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval.

2017-20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility.

2017-21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A

Play on Worlds).

2017-22 Sara Magliacane (VU), Logics for causal inference under uncertainty.

2017-23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces.

2017-24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning.

2017-25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search.

2017-26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch.

2017-27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors.

2017-28 John Klein (VU), Architecture Practices for Complex Contexts.

2017-29 Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT".

2017-30 Wilma Latuny (UvT), The Power of Facial Expressions.

2017-31 Ben Ruijl (UL), Advances in computational methods for QFT calculations.

2017-32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives.

2017-33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity.

2017-34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics .

2017-35 Martine de Vos (VU), Interpreting natural science spreadsheets .

2017-36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging .

2017-37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy .

2017-38 Alex Kayal (TUD), Normative Social Applications .

2017-39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR .

2017-40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems .

2017-41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle.

2017-42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets.

2017-43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval.

2017-44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering.

2017-45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement.

2017-46 Jan Schneider (OU), Sensor-based Learning Support.

2017-47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration.

2017-48 Angel Suarez (OU), Collaborative inquiry-based learning.

2016-01 Syed Saiden Abbas (RUN) , Recognition of Shapes by Humans and Machines.

2016-02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow.

2016-03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support.

2016-04 Laurens Rietveld (VU), Publishing and Consuming Linked Data.

2016-05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers.

2016-06 Michel Wilson (TUD), Robust scheduling in an uncertain environment.

2016-07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training.

2016-08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data.

2016-09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts.

2016-10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms.

2016-11 Anne Schuth (UVA), Search Engines that Learn from Their Users.

2016-12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems.

2016-13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach.

2016-14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization.

2016-15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments.

2016-16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward.

2016-17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms.

2016-18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web.

2016-19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data.

2016-20 Daan Odijk (UVA), Context & Semantics in News & Web Search.

2016-21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground.

2016-22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems.

2016-23 Fei Cai (UVA), Query Auto Completion in Information Retrieval.

2016-24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach.

2016-25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior.

2016-26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains.

2016-27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media.

2016-28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control.

2016-29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning.

2016-30 Ruud Mattheij (UvT), The Eyes Have It.

2016-31 Mohammad Khelghati (UT), Deep web content monitoring.

2016-32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations.

2016-33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example.

2016-34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment.

2016-35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation.

2016-36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies.

2016-37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry.

2016-38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design.

2016-39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect.

2016-40 Christian Detweiler (TUD), Accounting for Values in Design.

2016-41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance.

2016-42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora.

2016-43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice.

2016-44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration.

2016-45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control.

2016-46 Jorge Gallego Perez (UT), Robots to Make you Happy.

2016-47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks.

2016-48 Tanja Buttler (TUD), Collecting Lessons Learned.

2016-49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis.

2016-50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains.

2015-01 Niels Netten (UVA), Machine Learning for Relevance of Information in Crisis Response.

2015-02 Faiza Bukhsh (UVT), Smart auditing: Innovative Compliance Checking in Customs Controls.

2015-03 Twan van Laarhoven (RUN), Machine learning for network data.

2015-04 Howard Spoelstra (OUN), Collaborations in Open Learning Environments.

2015-05 Christoph Bsch (UT), Cryptographically Enforced Search Pattern Hiding.

2015-06 Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes.

2015-07 Maria-Hendrike Peetz (UVA), Time-Aware Online Reputation Analysis.

2015-08 Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions.

2015-09 Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems.

2015-10 Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning.

2015-11 Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins.

2015-12 Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks.

2015-13 Giuseppe Procaccianti (VU), Energy-Efficient Software.

2015-14 Bart van Straalen (UT), A cognitive approach to modeling bad news conversations.

2015-15 Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation.

2015-16 Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork.

2015-17 Andr van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs.

2015-18 Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories.

2015-19 Bernardo Tabuenca (OUN), Waste Not, Want Not! - Managing Relational Data in Asymmetric MemoriesUbiquitous Technology for Lifelong Learners.

2015-20 Los Vanhe (UU), Using Culture and Values to Support Flexible Coordination Using Culture and Values to Support Flexible Coordination.

2015-21 Sibren Fetter (OUN), Using Culture and Values to Support Flexible CoordinationUsing Peer-Support to Expand and Stabilize Online Learning.

2015-22 Zhemin Zhu (UT), Co-occurrence Rate Networks - Towards separate training for undirected graphical models.

2015-23 Luit Gazendam (VU), Using Culture and Values to Support Flexible CoordinationCataloguer Support in Cultural Heritage.

2015-24 Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation.

2015-25 Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection.

2015-26 Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure.

2015-27 Sndor Hman (CWI), Updating compressed colomn stores.

2015-28 Janet Bagorogoza (EUR), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO.

2015-29 Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains.

2015-30 Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning.

2015-31 Yakup Ko (TUD), On the robustness of Power Grids.

2015-32 Jerome Gard (UL), Corporate Venture Management in SMEs.

2015-33 Frederik Schadd (UM), Ontology Mapping with Auxiliary Resources.

2015-34 Victor de Graaff (UT), Gesocial Recommender Systems.

2015-35 Jungxao Xu (TUD), Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction.

2014-01 Nicola Barile (UU), Studies in Learning Monotone Models from Data.

2014-02 Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method.

2014-03 Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions.

2014-04 Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation.

2014-05 Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability.

2014-06 Damian Tamburri (VU), Supporting Networked Software Development.

2014-07 Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior.

2014-08 Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints.

2014-09 Philip Jackson (UVT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language.

2014-10 Ivan Salvador Razo Zapata (VU), Service Value Networks.

2014-11 Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support.

2014-12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control.

2014-13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains.

2014-14 Yangyang Shi (TUD), Language Models With Meta-information.

2014-15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare.

2014-16 Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria.

2014-17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically:

Secondary Use of Patient Data and Semantic Interoperability.

2014-18 Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations.

2014-19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support.

2014-20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link.

2014-21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments.

2014-22 Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training.

2014-23 Eleftherios Sidirourgos (UVA/CWI), Space Efficient Indexes for the Big Data Era.

2014-24 Davide Ceolin (VU), Trusting Semi-structured Web Data.

2014-25 Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction.

2014-26 Tim Baarslag (TUD), What to Bid and When to Stop.

2014-27 Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty.

2014-28 Anna Chmielowiec (VU), Decentralized k-Clique Matching.

2014-29 Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software.

2014-30 Peter de Cock (UVT), Anticipating Criminal Behaviour.

2014-31 Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support.

2014-32 Naser Ayat (UVA), On Entity Resolution in Probabilistic Data.

2014-33 Tesfa Tegegne (RUN), Service Discovery in eHealth.

2014-34 Christina Manteli(VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems..

2014-35 Joost van Oijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach.

2014-36 Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models..

2014-37 Maral Dadvar (UT), Experts and Machines United Against Cyberbullying.

2014-38 Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing..

2014-39 Jasmina Maric (UVT), Web Communities, Immigration, and Social Capital.

2014-40 Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education..

2014-41 Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text.

2014-42 Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models.

2014-43 Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments .

2014-44 Paulien Meesters (UVT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden..

2014-45 Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach.

2014-46 Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity.

2014-47 Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval.

2013-01 Viorel Milea (EUR), News Analytics for Financial Decision Support.

2013-02 Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing.

2013-03 Szymon Klarman (VU), Reasoning with Contexts in Description Logics.

2013-04 Chetan Yadati(TUD), Coordinating autonomous planning and scheduling.

2013-05 Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns.

2013-06 Romulo Goncalves(CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience.

2013-07 Giel van Lankveld (UVT), Quantifying Individual Player Differences.

2013-08 Robbert-Jan Merk(VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators.

2013-09 Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications.

2013-10 Jeewanie Jayasinghe Arachchige(UVT), A Unified Modeling Framework for Service Design..

2013-11 Evangelos Pournaras(TUD), Multi-level Reconfigurable Self-organization in Overlay Services.

2013-12 Marian Razavian(VU), Knowledge-driven Migration to Services.

2013-13 Mohammad Safiri(UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly.

2013-14 Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning.

2013-15 Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications.

2013-16 Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation.

2013-17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid.

2013-18 Jeroen Janssens (UVT), Outlier Selection and One-Class Classification.

2013-19 Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling.

2013-20 Katja Hofmann (UVA), Fast and Reliable Online Learning to Rank for Information Retrieval.

2013-21 Sander Wubben (UVT), Text-to-text generation by monolingual machine translation.

2013-22 Tom Claassen (RUN), Causal Discovery and Logic.

2013-23 Patricio de Alencar Silva(UVT), Value Activity Monitoring.

2013-24 Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning.

2013-25 Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System.

2013-26 Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning.

2013-27 Mohammad Huq (UT), Inference-based Framework Managing Data Provenance.

2013-28 Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience.

2013-29 Iwan de Kok (UT), Listening Heads.

2013-30 Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support.

2013-31 Dinh Khoa Nguyen (UVT), Blueprint Model and Language for Engineering Cloud Applications.

2013-32 Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development.

2013-33 Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere.

2013-34 Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search.

2013-35 Abdallah El Ali (UVA), Minimal Mobile Human Computer Interaction.

2013-36 Than Lam Hoang (TUe), Pattern Mining in Data Streams.

2013-37 Dirk Brner (OUN), Ambient Learning Displays.

2013-38 Eelco den Heijer (VU), Autonomous Evolutionary Art.

2013-39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems.

2013-40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games.

2013-41 Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning.

2013-42 Lon Planken (TUD), Algorithms for Simple Temporal Reasoning.

2013-43 Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts.

2012-01 Terry Kakeeto (UVT), Relationship Marketing for SMEs in Uganda.

2012-02 Muhammad Umair(VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models.

2012-03 Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories.

2012-04 Jurriaan Souer (UU), Development of Content Management System-based Web Applications.

2012-05 Marijn Plomp (UU), Maturing Interorganisational Information Systems.

2012-06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks.

2012-07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions.

2012-08 Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories.

2012-09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms.

2012-10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment.

2012-11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics.

2012-12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems.

2012-13 Suleman Shahid (UVT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions.

2012-14 Evgeny Knutov(TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems.

2012-15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes..

2012-16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment.

2012-17 Amal Elgammal (UVT), Towards a Comprehensive Framework for Business Process Compliance.

2012-18 Eltjo Poort (VU), Improving Solution Architecting Practices.

2012-19 Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution.

2012-20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing.

2012-21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval.

2012-22 Thijs Vis (UVT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?.

2012-23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction.

2012-24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval.

2012-25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application.

2012-26 Emile de Maat (UVA), Making Sense of Legal Text.

2012-27 Hayrettin Grkk(UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games.

2012-28 Nancy Pascall (UVT), Engendering Technology Empowering Women.

2012-29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval.

2012-30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making.

2012-31 Emily Bagarukayo (RUN) , A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure.

2012-32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning.

2012-33 Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON) .

2012-34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications.

2012-35 Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics .

2012-36 Denis Ssebugwawo (RUN) , Analysis and Evaluation of Collaborative Modeling Processes .

2012-37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation.

2012-38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms.

2012-39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks.

2012-40 Agus Gunawan (UVT) , Information Access for SMEs in Indonesia .

2012-41 Sebastian Kelle (OU), Game Design Patterns for Learning.

2012-42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning.

2012-44 Anna Tordai (VU) , On Combining Alignment Techniques.

2012-45 Benedikt Kratz (UVT), A Model and Language for Business-aware Transactions.

2012-46 Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation.

2012-47 Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior.

2012-48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data.

2012-49 Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions.

2012-50 Steven van Kervel (TUD) , Ontologogy driven Enterprise Information Systems Engineering .

2012-51 Jeroen de Jong (TUD) , Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching.

2011-01 Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models.

2011-02 Nick Tinnemeier(UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language.

2011-03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems.

2011-04 Hado Philip van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms.

2011-05 Bas van de Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang(TUE), Semantically-Enhanced Recommendations in Cultural Heritage.

2011-07 Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction.

2011-08 Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues.

2011-09 Tim de Jong (OU), Contextualised Mobile Media for Learning.

2011-10 Bart Bogaert (UVT), Cloud Content Contention.

2011-11 Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI

Perspective.

2011-12 Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining.

2011-13 Xiaoyu Mao (UVT), Airport under Control; Multiagent Scheduling for Airport Ground Handling .

2011-14 Milan Lovric(EUR), Behavioral Finance and Agent-Based Artificial Markets.

2011-15 Marijn Koolen (UVA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval.

2011-16 Maarten Schadd (UM), Selective Search in Games of Different Complexity.

2011-17 Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness.

2011-18 Mark Ponsen (UM), Strategic Decision-Making in complex games.

2011-19 Ellen Rusman (OU), The Mind ' s Eye on Personal Profiles.

2011-20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach.

2011-21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems.

2011-22 Junte Zhang (UVA), System Evaluation of Archival Description and Access.

2011-23 Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media.

2011-24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior.

2011-25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics.

2011-26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots.

2011-27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns.

2011-28 Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure.

2011-29 Faisal Kamiran (TUE), Discrimination-aware Classification.

2011-30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions.

2011-31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality.

2011-32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science.

2011-33 Tom van der Weide (UU), Arguing to Motivate Decisions.

2011-34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations .

2011-35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training .

2011-36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach.

2011-37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference.

2011-38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization.

2011-39 Joost Westra (UU), Organizing Adaptation using Agents in Serious Games .

2011-40 Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development.

2011-41 Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control.

2011-42 Michal Sindlar (UU), Explaining Behavior through Mental State Attribution .

2011-43 Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge .

2011-44 Boris Reuderink (UT), Robust Brain-Computer Interfaces.

2011-45 Herman Stehouwer (UVT), Statistical Language Models for Alternative Sequence Selection.

2011-46 Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work.

2011-47 Azizi Bin Ab Aziz(VU), Exploring Computational Models for Intelligent Support of Persons with Depression.

2011-48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent .

2011-49 Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality.