AN AUTOMATIC FAILURE RECOVERY METHOD FOR WORLD-ALTERING COMPOSITE SEMANTIC WEB SERVICES

HADI SABOOHI

THESIS SUBMITTED IN FULFILLMENT

OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

UNIVERSITY OF MALAYA

KUALA LUMPUR

2013

Abstract

The reliability of services is a critical factor in ensuring positive customer perception on overall quality. To perform complex actions, it is necessary to combine smaller services to generate a synergy of services. However, a composite Web service will not fulfill a user's goal, unless the composite services can be recovered should there be a failure. It is inevitable that several Web services may either perish or fail before and during the execution. One of the responsibilities of the mediator, which stands between providers and the consumers, is to monitor the execution of the services. In case of a failure, the composite service needs to be adapted so that the system survives.

The challenging issues of the recovery of composite services are its probability, the accuracy of the resulting composite service, and the Quality of Service (QoS) deviation from the original. Traditionally, service-based systems often substitute a matching service for the failed service. The accuracy of a matching can be enhanced by employing semantics. Semantic annotations of Web services describe the services' functionality unambiguously. However, finding a matching service is not always successful. There are some approaches, which replace a sequence of services, including the failed service with another set of services to comply with the required task. Their experiments show improvements to the recovery of the unsuccessful composite services. Nevertheless, these methods have not totally rectified the problem.

In the proposed solution, called SRPFR, an automatic renovation plan is utilized to improve the likelihood of impeding the system from a total failure. The method first considers a subgraph of services (including the failed service) in a digraph of Web services, i.e. the Original Subdigraph. The subdigraph may begin not only from the failed service but

also preceding the well-executed services. Then, the method searches for a compatible subdigraph (Replacement Subdigraph) to be placed in lieu of the Original Subdigraph. Broadening the beginning point of the search for the Original Subdigraph increases the probability of discovering a matched replacement. The Original Subdigraph may contain world-altering services with real-world effects, which must be compensated before the replacement. The recovery is divided into two phases, offline and online, to reduce the delay of the replacement at the onset of failure. The time-consuming calculations are done in the offline phase, and the replacement is done in the online phase. This prevents any increment of the response time of the composite service, even though a failure may occur.

The proposed approach is evaluated with synthetic test collections of composite semantic services using the atomic services and their related ontologies of a standard atomic service test collection called OWLS-TC. The simulation results show a recovery probability of more than 55% for a more realistic collection of services. In addition, the actual replacement delay is maintained at a minimum constant value regardless of the size of the test collection. From the QoS deviation aspect, the method can adapt the composite service without a major change in the prioritized QoS properties.

Abstrak

Keyakinan terhadap perkhidmatan-perkhidmatan adalah satu faktor yang amat penting dalam memastikan tanggapan pelanggan-pelanggan yang memberangsangkan terhadap kualiti secara keseluruhan. Untuk melaksanakan tindakan-tindakan yang lebih meluas, maka ia adalah perlu untuk menggabungkan kesemua perkhidmatan yang kecil untuk menjana perkhidmatan yang lebih berkesan secara keseluruhan. Walau bagaimanapun, perkhidmatan web yang terbahagi-bahagi atau terpecah-pecah tidak dapat memenuhi matlamat pengguna-pengguna, melainkan jika ada satu pendekatan yang dapat meningkatkan kemampuan perkhidmatan web tersebut daripada masalah dan kegagalan tersebut. Ia tidak dapat dielakkan kerana beberapa perkhidmatan web sebelum ini berlaku masalah atau kegagalan ketika dalam pelaksanaan dan pelancarannya. Salah satu tanggungjawab pengantara, yang berada di antara pembekal-pembekal dan pengguna-pengguna adalah untuk memantau pelaksanaan perkhidmatan-perkhidmatan tersebut. Dalam kes kegagalan, perkhidmatan web tersebut perlu disesuaikan supaya sistem dapat diteruskan dengan berkesan.

Isu-isu yang mencabar terhadap pemulihan perkhidmatan yang terbahagi-bahagi tersebut adalah kebarangkalian dan ketepatan perkhidmatan yang terbahagi-bahagi yang terhasil, dan Kualiti sisihan Perkhidmatan (QoS) daripada asal. Secara tradisinya, perkhidmatan berdasarkan system-sistem selalunya menggantikan perkhidmatan yang gagal dengan perkhidmatan yang sepadan. Ketepatan padanan boleh dipertingkatkan dengan menggunakan semantik. Penjelasan-penjelasan dalam Semantik perkhidmatan-perkhidmatan sesawang (Web) yang menggambarkan penggunaan perkhidmatan-perkhidmatan yang begitu jelas. Walau bagaimanapun, pencarian perkhidmatan yang sepadan tidak sentiasanya berjaya. Terdapat beberapa pendekatan yang menggantikan urutan perkhidmatan,

termasuk perkhidmatan yang gagal dengan set perkhidmatan-perkhidmatan yang lain untuk melakukan tugas-tugas yang dikehendaki. Melalui penyelidikan yang dijalankan, ia menunjukkan peningkatan kepada pelaksanaan perkhidmatan yang terbahagi-terbahagi itu adalah berjaya. Walaupun demikian, kaedah ini tidak benar-benar memperbetulkan masalah tersebut.

Dalam penyelesaian yang dicadangkan, yang dipanggil SRPFR, pelan pengubahsuaian automatik digunakan untuk meningkatkan kemungkinan yang menghalang sistem daripada jumlah kegagalan. Kaedah pertama menganggap subgraf bagi perkhidmatan-perkhidmatan (termasuk perkhidmatan yang gagal) dalam digraf perkhidmatan Web iaitu Subdigraph Asal. Subdigraph boleh bermula bukan sahaja dari perkhidmatan yang gagal tetapi juga perkhidmatan sebelumnya baik disempurnakan. Kemudian, kaedah tersebut mencari subdigraph yang sesuai (Subdigraph Penggantian) untuk ditukar ganti dengan Subdigraph Asal tersebut. Memperluas titik permulaan carian untuk Subdigraph Asal meningkatkan kebarangkalian tersebut bagi menemukan penggantian dipadankan. Subdigraph asal mungkin mengandungi perkhidmatan yang dunia yang berubah-ubah dengan kesan-kesan dunia sebenar, yang mesti diimbangi sebelum penggantian. Pemulihan dibahagikan kepada dua fasa, luar talian dan dalam talian, untuk mengurangkan kelewatan penggantian pada permulaan kegagalan. Pengiraan yang memakan masa yang dilakukan dalam fasa offline, dan penggantian dilakukan dalam fasa dalam talian. Ini menghalang sebarang kenaikan bagi masa jawaban terhadap perkhidmatan terbahagi-bahagi, walaupun kegagalan mungkin berlaku.

Pendekatan ini dinilai dengan koleksi ujian sintetik terhadap perkhidmatan-perkhidmatan semantik yang terbahagi-bahagi menggunakan perkhidmatan atom dan ontologi yang

berkaitan dengan mereka koleksi ujian perkhidmatan piawaian atom dinamakan sebagai OWLS-TC. Keputusan-keputusan simulasi menunjukkan kebarangkalian pemulihan lebih daripada 55% untuk koleksi perkhidmatan yang lebih realistic dan sebenar. Di samping itu, kelewatan penggantian yang sebenar dikekalkan pada nilai malar yang minimum tanpa mengira saiz koleksi ujian. Dari aspek kelainan QoS, kaedah tersebut boleh menyesuaikan perkhidmatan yang terbahagi-bahagi tanpa perubahan besar dalam ciri-ciri utama bagi QoS.

Acknowledgements

I would like to thank all those people who have helped me to finish this thesis and supported me throughout my whole studies. Of the many people who deserve thanks, some are particularly prominent, such as my supervisor, Associate Professor Datin Dr. Sameem Abdul Kareem, for her support, encouragement and advice. In particular, I wish to express my thanks to her for hosting me in the Artificial Intelligence Research Lab at University of Malaya, which proved to be the most forming experience for my research and for this thesis as well. I would also like to thank many colleagues and friends for their invaluable discussions and stimulating conversations.

I am indebted to my parents and rest of my family for their endless support and patience. Finally, it was my wife, Amineh, and her love that kept me going and helped me tremendously in finishing this thesis. I missed the extraordinary opportunity to spend time with my family and specially with my lovely princess, Saba, for the sake of my studies.

Associate Professor Dr. Hassan Abolhassani is the one who inspired me to start this journey and I am thankful for that for the rest of my life.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACM	Association for Computing Machinery
AFWS	Assumed Failing Web Service
AI	Artificial Intelligence
API	Application Programming Interface
$\mathcal{A}$*-WSCE*	Adaptive Web Service Composition and Execution (Chafle et al., 2006)
CSP	Constrained Shortest Path (Yu & Lin, 2005b)
CSPB	Constrained Shortest Path Backup (Yu & Lin, 2005a)
CSPR	Constrained Shortest Path Replacement (Yu & Lin, 2005a)
CSWS	Composite Semantic Web Service
DAG	Directed Acyclic Graph
DAML	DARPA Agent Markup Language
DAML-S	DAML Services
DARPA	Defense Advanced Research Projects Agency
Digraph	Directed Graph
DL	Description Logics
EAI	Enterprise Application Integration
ET	Execution Time
EC	Execution Cost
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ID	Identification/Identity/Identifier

IEEE	Institute of Electrical and Electronics Engineers
IO	Input and Output
IOPE	Input and Output, and Precondition and Effect
IOPR	Input and Output, and Precondition and Result
IOR	Input and Output, and Result
μs	microseconds
ms	milliseconds
OPOSSum	Online Portal for Semantic Services
OSRM	Original Subdigraph Ranking Measure
OWL	Web Ontology Language
OWL-S	OWL and Services (Semantic Markup for Web Services)
OWLS-TC	OWL-S Service Retrieval Test Collection
PDDL	Planning Domain Definition Language
PE	Precondition and Effect
PR	Precondition and Result
QoS	Quality of Service
R	Reliability
RM	Ringgit Malaysia
RSRM	Replacement Subdigraph Ranking Measure
SAWSDL	Semantic Annotations for Web Services Description Language (WSDL)
SAWSDL-TC	SAWSDL Service Retrieval Test Collection
SBT	Services Behavioral Type
SEALS	Semantic Evaluation At Large Scale
SOA	Service Oriented Architecture

SOAP	Simple Object Access Protocol
SRPFR	Subdigraph Renovation Plan for Failure Recovery (Our proposed method)
Subdigraph	Sub- + Directed Graph (Digraph)
SWRL	Semantic Web Rule Language
SWS	Semantic Web Service
SWS-TC	SWS Test Collection
SWSC	Semantic Web Service Composition
SWSO	Semantic Web Service Ontology
TC	Test Collection
UC	Undo Cost
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WAN	Wide Area Network
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WSC	Web Service Composition
WSDL	Web Services Description Language
WSDL-S	Web Service Semantics
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMO-Lite	Lightweight Semantic Descriptions for Services on the Web
WSMO-Lite-TC	WSMO-Lite Test Collection

Chapter 1: Introduction

1.1 Overview

Web services are the building blocks of today's service-based systems which ease interoperability of the participant enterprises. Following more than a decade of introducing "semantic Web services" (McIlraith et al., 2001), which combine semantic technologies with Web services, a few description languages are introduced. The description languages unambiguously annotate Web services so the machines would be able to process the descriptions; hence, the machines can automatically discover, compose, and execute the Web services. In general, "Semantics" for Web services annotates their functional and non-functional properties.

The proposed languages describe the Web services from two opposite directions. The languages called OWL-S (OWL and Services) (Martin et al., 2004), and WSML (Web Service Modeling Language) (Roman et al., 2005) semantically describe the Web services from a top-down approach, i.e. an expressive framework is proposed. Conversely, other languages such as Web Service Semantics (WSDL-S) (Akkiraju et al., 2005), adopt a bottom-up modeling approach to supplement existing service specifications such as Web Services Description Language (WSDL) (Christensen et al., 2001) with semantic annotations. Further, the World Wide Web Consortium (W3C) has created a standard language called Semantic Annotations for WSDL (SAWSDL) (SAWSDL Working Group, 2007) to describe the Web services unambiguously.

In a Web service-based architecture, a provider publishes its services in a registry. The consumers, based on their needs, search the registry to look for a matching service. The search is done by either the consumer or an agent, which acts as a mediator and stands

between the provider and the consumer (Booth et al., 2004). In a more advanced architecture, the mediator handles other tasks as well (Erl, 2007; McIlraith et al., 2001; W3C Working Group, 2004), which are as follows:

Publish The mediation commences from the publication of service specifications into a registry.

Discovery The mediator discovers the right service to fulfill the consumers' goal.

Composition A single service is usually not able to perform all the required tasks of a business process. The smaller services are integrated to represent a synergy of services, which is capable of managing interoperation among the services, and performing complex actions (Medjahed, 2004). The composite service is executed by the mediator. The mediator handles the negotiations among the providers and their consumers (Kuropka et al., 2008).

Execution and Monitoring The mediator invokes either a single or a composite service. The execution of the services are controlled, and monitored (Vaculín & Sycara, 2008) so that the required service is delivered to the end user (Erl, 2007; W3C Working Group, 2004). The monitoring ensures a smooth execution of the services such that even if an execution failure happens, the consumer receives the desired result. This smooth execution includes fulfillment of both the required functional properties as well as promising non-functional properties, i.e. Quality of Service (QoS).

Failure Recovery The monitoring catches faults or errors, and the mediator recovers the failure in order to continue the execution.

An overall view of the architecture of such a system is depicted in Figure 1.1.

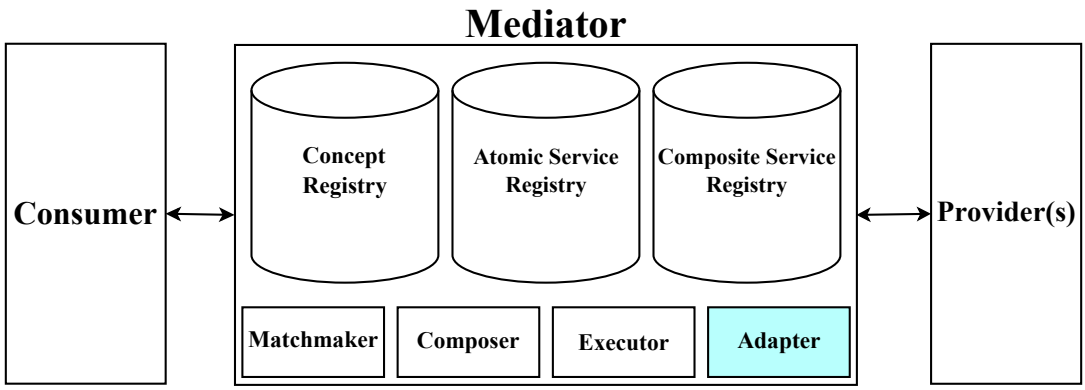


Figure 1.1.: An Architecture of a Web Service-based System

The Web services have functional and non-functional properties. The **functional** properties include input, output and conditions. The availability of the required input and conditions means the generation of the stated outputs and results by a successful execution of the service. The **non-functional** properties are related to the competing properties of different providers, which help the users to choose the particular service among all available functionally equivalent services. The non-functional properties include execution time, cost, and reliability of the service.

An execution of a service might be unsuccessful due to several reasons. The reasons are based on the functional and non-functional causes at the top level. The mediator enhances future executions of the composite service by reasoning about the causes of the failure. This enhancement is done by either avoiding the situation that causes the problem or being prepared for a solution to complete the perturbed execution.

A straightforward solution to failure recovery of a composite service is to substitute the failed atomic service with another atomic service. These two services must have equal functional properties and similar non-functional properties.

1.2 The Problem

There are more than 28000 public Web services available (Lausen & Haselwanter, 2007; Seekda, 2011; Steinmetz et al., 2009). However, according to the literature review, the

approaches are not always successful in finding an atomic Web service to replace a failed service. In order to test unreliability of the atomic replacement approaches, we conducted an experiment to find an answer to such a research question: “What is the success rate of an atomic replacement approach in a registry of atomic Web services?”. The experiment is elaborated in Section 5.2.1, and the results prove the low percentage of successful recoveries.

Existing solutions further replace a number of services, which are the components of the composite services, with other Web services. Generally however, **the likelihood of recovery is not high**, which is proved by the results presented in Section 5.3. Furthermore, approaches with reasonable likelihood of recovery decrease the accuracy of the resulting composite. The decrement does not then, allow users to achieve their goals.

Besides, composition methods consider criteria such as Quality of Service (QoS) apart from accuracy of the functionality. Therefore, any *deviation on the QoS of the adapted composite service is not desirable*.

Other than these problems, even though semantic Web services explicitly distinguish between information-providing and world-altering services, recovery approaches do not take them into account, i.e. they do not take care of the world-altering services’ real-world effects.

1.2.1 The Challenges

There are some challenging issues for the recovery of failures of composite services:

- The **probability of recovery** of a failed system, which is based on composite Web services, must be increased. Therefore, from the consumers point of view the system responds most of the time.

- The **goal of the adapted composite**, i.e. the information transformed and the effects generated, should be similar to the goal of the initial composite service. The similarity ensures that even though a failure may occur, the requirements of the end user is fulfilled.
- Although Web services are erroneous, from the consumers' point of view, their failure should be invisible; hence, the **recovery method should be fast enough** so that the failure and its recovery process is hidden from the user.
- Based on the requirements and the goals of the user, a composite service is generated. However, if the composite service's execution is hindered by a failure, the recovery method should consider both the requirements and the goals of the user. The requirements include the desired non-functional properties, i.e. QoS. Therefore, adaptation of the composite service should ideally ensure no deviation on the QoS and practically a minimum deviation of QoS.

Trade-off

Among the above challenges there are some tricky trade-offs:

- The first trade-off is between the number of replacements and the accuracy of the adapted composite service. Decreasing the threshold for the similarity of the resulting service to the initial composite allows the method to have more candidates for the replacement; hence, the increment of the probability of recovery.
- The second trade-off is that allowing more time for the recovery of the failed composite, enables the adapter to find a better replacement, which increases the final delay for the execution of the composite service.

1.3 The Objectives

The objectives of this research are as follows:

- **To increase the probability of recovery** of the failed composite Web services. The increase on the recovery probability of service failures should not decrease the accuracy of the resulting service in accordance with the original composite service.
- **To minimize the failure recovery delay time**. Any failure occurring during the execution of the composite service must be addressed without any delay.
- **To ensure that the QoS of the adapted service be close enough to the promised QoS** of the original composite service, i.e. there should be no deviation on the QoS.

In order to achieve the objectives, this study proposes a method to differentiate between the time-consuming calculations (of finding the best replacement to adapt the failed composite service) and the actual replacement; which must be done at the failure time.

In the **offline phase**, i.e. before commencing the execution of the composite Web service, a search through the structure of the original composite service needs to be carried out to choose a set of services which are the best to be removed. The search is done for every component of the composite service separately. Then, for this set of services a similar set is chosen. This similar set of services will be the topmost replacement candidate should any failure occurs. The outcome of the offline phase is a list of triple items. The list indicates: 1) a component of the original composite service (which is prone to failure), 2) a set of services including the service that is assumed to fail, 3) a replacement set of services with a similar goal and a close enough set of non-functional properties.

In the **online phase**, during the execution of the composite service, if any failure happens, the list is already made available. So, for the failed service, its related set of services which must be removed and its replacement, are used to adapt the composite service. The execution is monitored by the mediator to discover the failures.

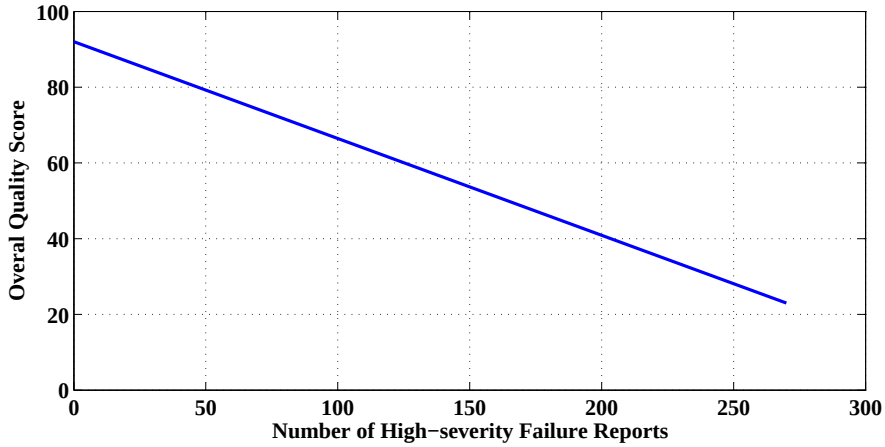


Figure 1.2.: Correlations between Software Quality and High-severity Failure Reports (Adapted from (Lyu, 1996))

Other similar methods, which will be elaborated in Chapter 2, mostly use a one-to-one replacement to switch the failed service with an alternative service. We will prove that the so-called atomic replacement method is not reliable (Section 5.2 on page 80). A more sophisticated method switches the services using a many-to-many replacement. We use a combination of one-to-one, one-to-many and many-to-many replacements to achieve better results. Other than that, we may even remove a well-executed service to increase the probability of recovery, which requires compensation (Bhiri et al., 2006, 2011) of world-altering services if they are in the composite's structure.

1.4 Significance of the Study

Reliability is formally defined as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time" (IEEE, 1990).

Users usually discard a software which is unreliable and does not fulfill their needs. For example, a study in the telecommunication industry showed that the overall quality score (customer satisfaction) significantly decreased by high-severity failures as shown in Figure 1.2 (Lyu, 1996). Hence, it is critical to either prevent a system from a failure or recover it when a failure occurs.

Reliability is a major issue, particularly when it comes to critical domains. Functionally equivalent Web services are created incrementally by various competing providers, which enables the software developers to offer more reliable, fault tolerant systems (Zheng & Lyu, 2010). The agent, which intermediates the negotiations between providers and their consumers, need to adapt the services based on the new changes. Service adaptations have also been studied in other fields such as mobile services (Liotta et al., 2002).

A Web service-based system must recover from a failure to ensure that the user can get his request fulfilled even though a failure is unavoidable. Increasing the failure recovery probability of such a system decreases the likelihood of the system being discarded by the users which in turn decreases the organizations' revenues.

1.5 Thesis Outline

The thesis is structured as follows: This chapter introduced the field, showed the problem and the gap that we discovered, and illuminated the objectives that we tend to achieve. Chapter 2 elaborates background issues regarding the semantic Web services and their mediation. It follows by an investigation of existing works in the area of the adaptation and failure recovery of composite services. Chapter 3 explains the proposed model for the recovery and then describes the components of the framework. Chapter 4 justifies a critical need for today's semantic services research field, a test collection of composite semantic services. Further, we propose an adapted model to generate a big test collection. Chapter 5 illustrates the results of the research from its recovery probability aspect and discusses the evaluation of the proposed model upon the generated test collections. Moreover, Chapter 6 shows the minimum delay time of our approach for the recovery, and the minimum deviation of the adapted services from their initial QoS. Finally, we draw our conclusions in Chapter 7 and make recommendations for future work in this area.

Chapter 2: Literature Review

2.1 Background

Nowadays, enterprises tend to outsource their services. The collaboration and the interoperation made in this way reduce their cost. Furthermore, the required time both to deploy the final application, and to fulfill their users' goals is decreased. Service Oriented Architecture (SOA) is a promising approach to this end (Bieberstein et al., 2008), which enables the interoperability among the participating businesses. A provider publishes its services and the consumers who need the service can discover it and further invoke it to get the desired service (W3C Working Group, 2004). The framework allows the providers to publish the specifications of their services in a repository so as to facilitate the discovery and matchmaking of the services. The consumers search through the specifications to find the most similar service that fulfills their needs (Erl, 2007).

Web service is an implementation proposed for SOA. The provider describes its services using a formal language called Web Services Description Language (WSDL) (Christensen et al., 2001). The consumer contacts the provider by Simple Object Access Protocol (SOAP) (Gudgin et al., 2007). The service descriptions are published in a repository titled Universal Description Discovery and Integration (UDDI). The consumers look for the services either in a UDDI or on the Web pages of the providers (Erl, 2007). There used to be public UDDIs provided by Microsoft and IBM, however the companies discontinued their provision in 2006 (Krill, 2005)

2.1.1 Semantic Service

The syntactic descriptions of the Web services (in WSDL) and their natural language specifications need humans to interpret. On the other hand, the developers need to manu-

ally select, and hard code the systems for a particular service even though they outsource their desired functions. However, having thousands of providers and hence many services available on the Web, the selection is a cumbersome task even for the experts. Therefore, the automation of various mediation tasks on the services such as discovery, execution, and monitoring is highly desired.

Semantic annotations of the services, which was proposed in the early stages of the introduction of "Semantic Web" (Berners-Lee et al., 2001; Hendler, 2001), facilitates the machines to process the service specifications (McIlraith et al., 2001). In order to automate the mediation of the services, they are augmented with extra information using formal languages. Having the machine processable information, users can delegate softwares (as agents) to reason on the information. Hence, the services are automatically and autonomously discovered, and further invoked (McIlraith et al., 2001).

Services are categorized into two major groups, information-providing and world-altering (McIlraith et al., 2001; Wu et al., 2003). The criterion for the categorization is that whether the service causes any change in the world. From this perspective, the categories are as follows:

Information-providing services The services in this group may need to get some information from the user, as an *input*. The successful execution of the service generates an *output* for the user. Moreover, the service may have a *condition* to be true prior to its execution (Wu et al., 2003). Examples of the information-providing services are flight information providers and temperature sensors (McIlraith et al., 2001).

World-altering services The world-altering services have all the features of the information providing group, i.e. input, output, and condition. Additionally, they cause a change (which is a physical side-effect or briefly an *effect*) on the real world. A

flight-booking program and a sensor controller are two examples of world-altering services (McIlraith et al., 2001).

Semantic specifications ease the division of the services into two aforementioned groups. Prior to the use of semantics for the services, when the specifications were all syntactical, there was no way to identify the existence of an effect of the service execution on the world other than mentioning it using natural language as a comment. Hence, the machines were unable to explicitly classify the services. Using semantic specifications, providers even elaborate the effects using specified formal expression languages (such as SWRL (Horrocks et al., 2004)). The division is crucial because usually the manipulation of world-altering services are not free (Greenfield et al., 2003), and so the agents need to consider extra effort in the selection and the execution of this category of services.

Formal Languages

Semantic services are described using specific languages. To this end, there are some languages proposed for the semantic description of the services. The languages describe the services from two perspectives which results in two conceptual models: *bottom-up* and *top-down*.

The **bottom-up** approaches annotate the underlying existing syntactic languages like WSDL (Chinnici et al., 2007; Christensen et al., 2001) with semantic specifications. The languages are Web Service Semantics (WSDL-S) (Akkiraju et al., 2005), Semantic Annotations for WSDL (SAWSDL) (SAWSDL Working Group, 2007), and Lightweight Semantic Descriptions for Services on the Web (WSMO-Lite) (Fensel et al., 2010).

On the other hand, the **top-down** approaches propose a new semantic based framework, i.e. high-level ontologies, to express the services. The major top-down languages are

OWL and Services (OWL-S) (Martin et al., 2004, 2006), and Web Service Modeling Language (WSML) (Bruijn et al., 2005).

Additionally, to express the conditions and effects of the semantic services, some expression languages have been used including Semantic Web Rule Language (SWRL) (Horrocks et al., 2004), Planning Domain Definition Language (PDDL) (Ghallab et al., 1998), and Web Service Modeling Language (WSML).

2.1.2 Mediation Techniques

The mediator, a software agent, stands between the service providers and the consumers and mediates their negotiations (Kuropka et al., 2008). The mediation tasks include but are not restricted to discovery, composition, invocation, and monitoring.

The discovery, together with matchmaking is the finding of the service and matching it to the user's request. If among the provided services there is no single service capable of complying with the request, the mediator composes a set of services, which are considered distributed business processes, to fulfill the user's demand. In a paradigm of service composition called "service orchestration" (Peltz, 2003), the mediator arranges and manages the interaction between services. The mediator issues the invocation command and monitors the execution of the services. If an execution fails, the mediator takes several approaches to complete the execution (Erl, 2007), which are investigated in Section 2.2.

2.1.3 Failure Reasons

Web processes contain services as their components. A thorough execution of a Web process highly depends on the correct execution of its constructing services. An execution of a service might be unsuccessful due to several reasons. Investigating the reasons of an unsuccessful execution of a service enables the mediators either to avoid the situation

that causes the problem or at least to prepare a solution to hinder the system from a total unsuccessful completion in future. The failure reasons are based on the functional and non-functional causes at the top level.

A Web service, as a participant in a structure of a composite service may fail due to the following causes (Di Nitto et al., 2008; Lin et al., 2009; Mahbub & Zisman, 2009; Subramanian et al., 2008; Vaculín et al., 2008; Zhai et al., 2009):

Functional causes

- Malfunctioning of the service: This is usually because of the application level errors.
- Unavailability of the service: The unavailability can be temporary or permanent.
  - The service may disappear permanently. The provider may not provide the service anymore or it may replace the service with a new one.
  - The connecting network has a failure, for example infrastructure breakdown.
  - Host overload. The number of requests are too high that the hosting is unable to serve.
  - User mobility. For example the user changes an accessing network which restricts its external access.

  The first unavailability cause is usually permanent and the others are temporary. The temporary causes can be remedied by for example repairing the network or through the introduction of an extra host.
- Software compatibility issues
  - The mismatches among the composed services. For example the changes related to the input and output formats.

  - Malformed response or errors related to serialization/deserialization. For example the changes of negotiating messages.

- The emergence of new requirements: Usually because of the reconfiguration which aims to enhance the fulfillment of needs.
- Changes in the context and the environment. Sometimes even though all the conditions at the provider side are constant, the context changes; such as, the accessing device may cause a disorder in getting the desired service.

Non-functional causes

A deviation from a promised Quality of Service (QoS) influences the user not to be satisfied with the service. The factors are related to time and cost of the execution which are as follows:

- Response time-out
- Network delay: For example because of network congestion
- Host overload: The provider cannot answer all the requests on time because of unexpectedly:
  - large number of requests for a service
  - large number of invocation of various services of the provider
- Unexpected input
- Unexpected data size

2.2 Failure Recovery

Handling the errors that might happen during the execution of a Web process is important. The error should be handled to provide a smooth execution of the Web process. A lack of such a monitoring and handling enforces the applications to specify their own error

handling mechanisms, hence the interoperability might be diminished.

2.2.1 Recomposition vs. Repair

A naive solution to handle an execution failure of a composite service is to stop the process, and recompose a new set of services that are able to perform the same tasks. However, it has been shown that the recomposition causes a long delay which is undesired (Yan et al., 2010). Therefore, it is highly demanded to avoid the prohibitively expensive overheads of recomposition from scratch and repair the structure, i.e. to adapt the composite service (Chafle et al., 2006).

The adaptation of a process has different aims as follows (Kazhamiakin et al., 2010):

Perfective Adaptation: Improving the current situation of the application in its quality aspects, etc.

Corrective Adaptation: Removing the undesired or faulty behavior of a system

Adaptive Adaptation: Responding to the context, interaction, and requirement changes of the application

Preventive Adaptation: Preventing future failures

Extending Adaptation: Adding newly required functions

Scope: In this research we emphasize more on the *corrective adaptation* of the Web processes, which adapts the system in order to eliminate the perished or the failed services.

2.2.2 Requirements

In order to propose a solution for the failure recovery of a composite service, the requirements for an approach which is able to rectify such a problem is investigated (Di Nitto et al., 2008; Lin et al., 2009; Mahbub & Zisman, 2009; Subramanian et al., 2008; Vaculín et al., 2008; Zhai et al., 2009). The requirements are as follows:

- Automation
- Adaptation Probability
- Time Complexity
- Accuracy
- QoS Deviation
- Consideration of World-altering Actions
- Experiments on a Standard Test Collection

In the following sections each item is elaborated.

Automation

The first and foremost requirement is that the adaptation is done automatically and autonomously. The method must minimize the human intervention for the adaptation. Hence, the applications must be able to discover, rank, and compose new services. The semantic descriptions of services are obvious necessities which help to automate the discovery, matchmaking and composition issues.

Adaptation Probability

The adaptation process tries to amend the structure so that its execution can be completed. The probability of the success of such an approach differs. The probability must be high enough so that in most cases the mediator can recover the system from a failure and the system works smoothly even with an unavoidable failure.

Time Complexity

The adaptation requires some calculations and interactions. The time complexity of such extra processes must be reasonable so that the adaptation is done in a minimum delay. Minimizing the delay isolates the awareness of the user from any likely failure occurrence

and its recovery process.

Accuracy

The adaptation demands for a replacement and an amendment in the primary structure of the services. Some of the smaller services of the composite service must be switched with others. Ideally, the replaced services should have exactly the same functionality or at least be similar practically.

The accuracy is critical since the end-user should get the requested goal. That is to say, the goal of the “Adapted Composite Service” must be similar to the goal of the “Original Composite Service”.

Semantic Web services ensure an unambiguous description of the Web services such that their discovery and composition occur automatically and most importantly accurately. Furthermore, the clear distinction of two major groups of the services, i.e. information providing and world-altering services based on the existence of an effect is a major contribution of using semantic Web services for Web processes. Hence, the mediator discovers and composes world-altering services along with the information providing services.

QoS Deviation

The adapted structure of the composite service must be functionally equivalent to the original structure. Additionally, non-functional properties of the composite service must be the same as the promised service to the end-user. Thus, there must be no deviation on the QoS.

Consideration of World-altering Actions

The approach must contemplate all kinds of services, i.e. information providing and world-altering. Hence, there should be methods to cope with the specific features of

world-altering services such as their effects on the real world.

Experiments on a Standard Test Collection

Ultimately, there is a strong need for a standard test collection of composite Web services. Current test collections of Web services and semantic Web services do not contain any composite service and they are just sets of single (atomic) services. The standard test collection definitely needs a set of world-altering services to be used for a test on both the information-providing and world-altering services (Küster & König-Ries, 2008).

A failure recovery approach for composite Web services needs to be tested on a standard test collection to prove its applicability, accuracy, etc.

2.2.3 Approaches to Failure Recovery

According to the studied requirements as discussed in Section 2.2.2, a number of approaches proposed various methods taking into account some of the mentioned features. The following sections overview the approaches to failure recovery. The categorization is based on the number of services which are substituted during the adaptation.

2.3 Atomic Repair

2.3.1 Re-invocation

Sometimes, a failure, especially the one that is caused by the environmental conditions of the service, might be temporary. In this case, a re-execution of the service may solve the problem. Usually the approaches try to re-invoke the service hoping that the re-invocations succeed. The retries are performed for a limited number or a limited time to keep the time constraints, i.e. the total execution time of the composite service does not exceed a pre-defined value (Subramanian et al., 2008; Vaculín et al., 2008).

However, repairing the execution of a failed service may not be successful. This happens when the problem is permanent or at least not solvable for a long time. So, the mediator takes the next step.

2.3.2 Atomic Replacement (one-to-one)

It is straightforward that an adaptation approach replaces the failed service with another service as shown in Figure 2.1. The other service can either be a redundant service or a similar service with a close functionality (Ganek & Corbi, 2003). A provider may replicate its Web service(s) on distributed servers so that if a temporary problem causes the unresponsiveness of the Web service, the users can get their service from the other servers; hence, the reliability improves (Sayal et al., 1998). According to IEEE standards (IEEE, 1990), this is called "**software diversity**":

> "A software development technique in which two or more functionally identical variants of a program are developed from the same specification by different programmers or programming teams with the intent of providing error detection, increased reliability, additional documentation, or reduced probability that programming or compiler errors will influence the end results."

Salas et al. (2006) provided an infrastructure, WS-Replication, for Wide Area Network (WAN) replication of web services. The proposed framework allows the providers to deploy their Web service in multiple replicas which addresses the Web services' low avail-

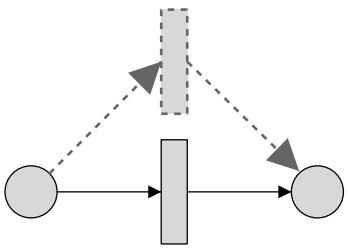


Figure 2.1.: Atomic Replacement (one-to-one), Adapted from (Möller & Schuldt, 2010)

ability problem. Furthermore, ANGEL, a redundant-based service composition method, increases the availability of composite services by replacing a failed component service with an alternative member of the redundant services (Guo et al., 2007).

Nevertheless, the problem remains unanswered when the failure is caused by for example a permanent unavailability of the Web service because of the business changes. So, it is required that the mediator finds another service to perform the task. The mediator matches the similar service (to the failed one) and replaces it into the structure.

Vaculín et al. (2008) and Wiesner et al. (2008) presented an approach firstly for handling the exceptions that may occur for invocation of a Web service and secondly for their recovery. They explained the conditions that may cause an erroneous state of the execution and provided their handlers, Constraint Violation handlers (CV-handlers). The pre-defined states of a service execution are *uninitialized*, *started*, *finished*, and *failed*. They provided recovery actions called *retry*, *replaceBy*, and *replaceByEquivalent* together with other action categories including *neutral*, *fault emitting*, *termination*, *compensation* and *adaptation* actions. The adaptation is based on a run-time discovery of the matching services. The work is based on a specific semantic Web service language, OWL-S; however, they claimed that the advantages are independent of the language. They extended the language to support the proposed features. In addition, the works used the semantic definitions of *effects* in OWL-S for *Automatic Compensation* of the so-called *finished* services preceding the failure by calling another run-time discovered suitable service. For monitoring purposes, the authors used the event-based execution monitoring and error handling approach, which explained mechanisms of monitoring for semantic web services based on OWL-S (Vaculín & Sycara, 2008).

Merits and Limitations:

- The recovery action called *replaceByEquivalent* uses the information about the service capabilities to discover and to match a similar service during run-time. The dynamic finding of an alternative service increases the probability of recovery. However, the delay of such a discovery at run-time is very crucial.
- The additional proposed tags of being a service *Vital*/*Non-Vital* or *replaceable*, which are not supported by OWL-S, are also required for their *Advanced Back & Forward Recovery*.
- The approaches are not purely Atomic Replacement methods because they go backwards through the structure of the composite service up to a point where they can move forward using an alternative path; however, they did not explicitly identify the details of the backward and forward strategies.

It has been proposed in Web Services Business Process Execution Language (WS-BPEL) (Alves et al., 2007) to define a compensation handler to undo the effects of a Web service and to execute them in reverse order (Greenfield et al., 2003). This is a static solution for the recovery, and if this static solution, i.e. the service, is not reachable the system would go to an inconsistent state (Wiesner et al., 2008).

There are some other approaches in which the main idea is the atomic replacement with different perspectives and approaches (Angarita et al., 2012; Liu et al., 2010; Subramanian et al., 2008; Taher et al., 2006).

2.3.3 Atomic Replacement: Discussion

Performance of an atomic replacement approach depends on its main bottleneck point, which is the discovery and matchmaking of the alternative service. Hence, the performance highly depends on the search time of the discovery method used.

The *Atomic Replacement* approaches are not sufficient for an advance and automatic failure recovery for composite services. The reason is that finding a similar service to be replaced in lieu of the failed service is not reliable (cf. Section 5.2). There should be other approaches in case the discovery of such an atomic replacement is unsuccessful.

2.4 Composite Replacement: 1-n (one-to-many)

In composite replacement, either or both the "to be removed" service(s) or the "to be placed" service(s) are composite.

The 1-n replacement, which is also called Atomic-to-Composite Replacement, is to find or to compose a composite Web service which fulfills the task of the failed (atomic) service. "Process reorganization" is an approach which is able to replace the failed Web service with a group of Web services (Subramanian et al., 2008). The composition time can be either before the execution or at the failure time. Figure 3.9b in Section 3.2.2 shows an overview of this replacement type.

2.5 Composite Replacement: n-m (many-to-many)

The n-m replacement, which is a Composite-to-Composite replacement, replaces a sequence of services including the failed service with another set of services. The replacement can also be a many-to-one replacement, in which a composite service is substituted with an atomic service. Figures 3.9c and 3.9d in Section 3.2.2 depict the overview of these replacement types.

There are various methods for composite replacement which are reviewed as follows.

2.5.1 Backup Path

Yu & Lin (2005a) proposed two algorithms to adapt a distributed business process. The adaptation is based on a creation of a backup path for every service in the execution

path. They adapt an autonomous business process if a failure occurs to ensure a non-interrupting execution. Further, they reconfigure the system not to use the failed service in future executions.

The proposed algorithms are based on abstract services (service classes). A business process is modeled as a Directed Acyclic Graph (DAG), and the business process and its service execution paths are constrained not to exceed a defined end-to-end delay.

An algorithm called **CSPB** is triggered when a failure occurs. The algorithm generates a backup path for every service to the end of the composite service in advance as shown in Figure 2.2. Hence, from a service in the structure there would be at least two paths to the end. The execution will continue through the secondary path if the first optimal path fails, i.e. the service preceding the failed service chooses the backup path. In order to calculate the backup path, the reverse graph of the original graph is considered. Then, from the source node of the reverse graph (which is the sink node of the original graph), at least two paths are calculated to every graph node. Hence, in the original graph every node has two paths, the optimal and the backup path, to the sink node.

The second algorithm, **CSPR**, is invoked whenever the failure of a service persists. It reconfigures the structure in an offline mode so that the executor refrains from the invocation of a service with a high probability of failure, i.e. a service which is persistently unresponsive. The paths are re-generated such that they ignore the failed service (of that specific service class) or they do not use that particular whole service class.

In brief, the CSPB algorithm is used to adapt a running composite and the CSPR is used to reconfigure the new composites having a previously persistent failed service in their structures.

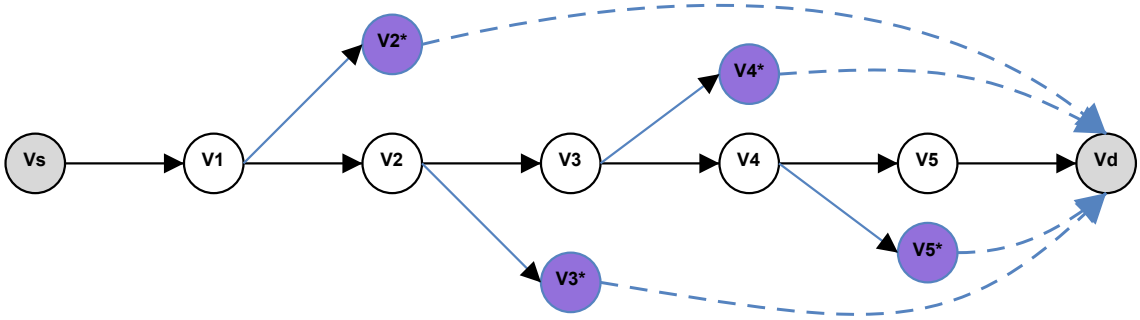


Figure 2.2.: CSPB Backup Paths (Yu & Lin, 2005a)

Merits and Limitations:

- For a running process including a failed service, the computed backup path commences just from the immediate predecessor. In order to increase the probability of a failure solution, the replacement may start from a service located not only one service before but also any service prior to the failed service.
- The approach considers the delay, cost and benefit. Other QoS properties should be considered to enhance the total QoS deviation of the composite service.
- For the purpose of evaluations, the authors simulated the execution of the composite Web services. They created the DAG of services using a **random** generation of edges among the service nodes.
- The researchers compared the algorithms' performance to their previous proposed algorithm called CSP (Yu & Lin, 2005b).
- The authors claimed that if there is a node (service) with no backup path for its execution, it is called a critical point for the business process. However, the problem can be eased if they go further backwards to find a backup path.

2.5.2 Region Reconfiguration

Lin et al. (2010) proposed a dynamic reconfiguration of the service processes. The objective is to preserve the primary QoS constraints. The approach replaces faulty services and their neighboring services.

The reconfiguration is threefold: identifying faulty regions of services, calculating constraints for each region, and recomposing regions. They proposed three service function replacement models including one-to-one, one-to-many and many-to-one as shown in Figure 3.9. Another motivation for the work is to heal the system if multiple failures happen in a service execution. The authors implemented the method in a system called "Llama middleware".

Merits and Limitations:

- The replacement approach gradually expands the region including the failed service as depicted in Figure 2.3 contemplating QoS constraints. Hence, the topmost criterion is the number of services in their "to be replaced" services, and the other criteria such as QoS are of less importance. It would be better if the approach uses an alternative method such as "weighted multiple-criteria measures" which equally balances other criteria.

  For example, if for a failed service, there are two regions with n, and m services, and $n < m$, their approach would definitely choose the first replacement with n services. However, if the region of m services are better to be reconfigured, e.g. in terms of QoS, this set of m service with even higher number of services must be selected.

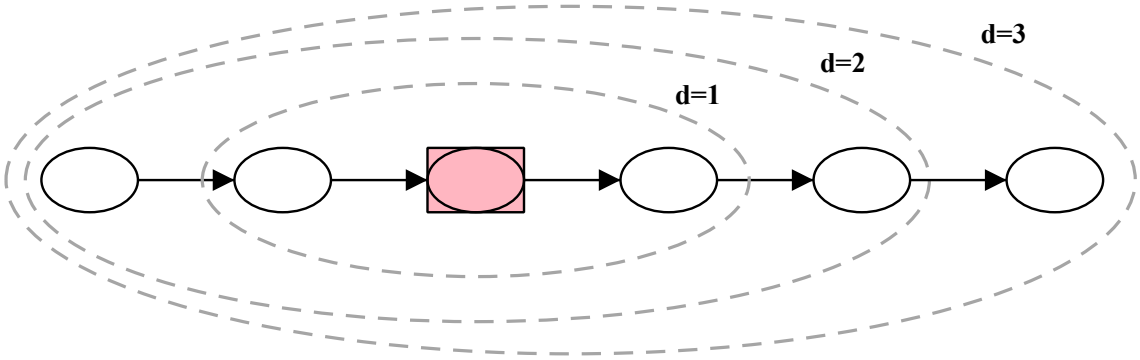


Figure 2.3.: Reconfiguration Region Example (Lin et al., 2010)

Region reconfiguration is also used to ensure the correctness using Services Behavioral Type (SBT) (Li et al., 2011).

2.5.3 Multiple QoS Constraints

Feng et al. (2007a) presented a model for the composition of Web services along with their invocation, and the recovery of its probable failures. The goal is to satisfy multiple QoS constraints for the component services. They declared that it is severe to have a backup path for every component Web service, e.g. in the CSPB algorithm (Yu & Lin, 2005a). The method moves backwards on the failed execution path, and finds another path with the highest utility using their proposed algorithm called Web Service Composition Partial Re-composition (WSCPR) (Feng et al., 2007b).

The above method considers multiple QoS properties. The actual values of QoS properties (q) are stored in a matrix with n rows for the service candidates and m columns for the QoS metrics. The properties are grouped to benefit properties, and cost properties, which are to be maximized (α metric values) and minimized (β metric values) respectively. Every QoS metric is weighted (w_x). The utility function is used for the selection of a service as a QoS evaluation metric. ∂_x is the difference value between the maximum and the minimum

values:

$$\mathcal{F}(k) = \sum_{i=1}^{\alpha} w_i * \left(\frac{q_{ai}(k) - q_{ai_{min}}}{\partial_i}\right) + \sum_{j=1}^{\beta} w_j * \left(\frac{q_{bj_{max}} - q_{bj}(k)}{\partial_j}\right)$$

$$\partial_i = q_{ai_{max}} - q_{ai_{min}}, \text{ if } q_{ai_{max}} - q_{ai_{min}} \neq 0, \text{ else } q_{ai}(k) - q_{ai_{min}} = 1 \tag{2.1}$$

$$\partial_j = q_{bj_{max}} - q_{bj_{min}}, \text{ if } q_{bj_{max}} - q_{bj_{min}} \neq 0, \text{ else } q_{bj_{max}} - q_{bj}(k) = 1$$

where:

$$0 < w_i, w_j < 1; \sum_{i=1}^{\alpha} w_i + \sum_{j=1}^{\beta} w_j = 1, \alpha + \beta = m \tag{2.2}$$

The WSCPR algorithm finds a new path with the highest utility as in Equation 2.1 which maximizes the number of common services between the broken graph and the new one. The time complexity is reported as $O(N^2 lm)$, N as the number of service classes each with l candidates, and m as the number of QoS requirements. The WSCPR's performance is compared with CSPB (Yu & Lin, 2005a), and it is illustrated that it performs faster during run-time.

Merits and Limitations

- The WSCPR continues backwards in the structure of the composite service to find a node with an out-degree of more than one. The algorithm should not only consider the out-degree of the predecessor services (being one or more) but also their QoS at the same time.

2.5.4 Multi-stage Adaptation

Chafle et al. (2006) proposed a multi-staged approach to the adaptation of Web services called Adaptive Web Service Composition and Execution ($\mathcal{A}$-*WSCE*). They defined the problem as a composition approach being able to adapt to dynamic changes in terms of functional and non-functional requirements of the service. They figured out the problem by a staged solution in which several workflows are generated.

The $\mathcal{A}$-*WSCE* approach uses two levels of adaptation. First, replacing an alternative instance of a service in a service type (class) such that the functionality is similar but the non-functional properties might be different. Second, a different template of services using other service types are replaced. Each level is done in a different stage. The researchers used a ranking function to select the best choices in every stage. The rankings are based on several criteria such as the length of the workflow, user comprehensibility of the workflow, QoS values of the instantiated workflows. For the purpose of adaptation, they follow an incremental recovery process. If any of the stages is unsuccessful to provide an alternative, the previous stage is triggered. Moreover, a feedback mechanism is also defined to health check the alternatives.

The $\mathcal{A}$-*WSCE* algorithms are improved in terms of first, the number of queries to get new QoS changes from the providers which yields better aggregate QoS, and second, stability of the processes (Chafle et al., 2007). This is done by approximating between the gains and their costs. The value of the changed information (VOC) (Harney & Doshi, 2006) determines the reactions of a process in the changed environment. The VOC helps to reduce the substitution of the workflows by selectively querying their providers and switching them when it is reasonable.

Merits and Limitations

- The $\mathcal{A}$-*WSCE* algorithm does not deal with the effects of the previously executed services.
- The composite structures are only sequential and other structures are not addressed.

2.5.5 Rebinding

A binding and re-binding approach is presented in (Canfora et al., 2005) and later in (Canfora et al., 2008). The approach focuses on a QoS-aware run-time binding of the abstract services to the concrete services based on Genetic Algorithms (GA) for optimizations. Their research is motivated by the fact that knowing the final QoS would be changed due to a measurement of the actual QoS attributes of the executed services, the composite service is required to be altered. The abstract services of *still to be executed slice* of the composite service will be rebound immediately after a determination of the re-binding need.

This approach re-estimates the workflow's QoS by receiving new information during execution. The new information is from either the QoS measurement of the executed services or the decisions made for the control flow structures such as *switch*. If the services are unavailable or the QoS deviates from an estimated value, and it is off the limits of a pre-defined threshold, then it re-binds the non-executed abstract services to new concrete services.

Merits and Limitations

- The approach triggers re-binding by predicting the deviation of the QoS estimated from the actual values.
- It merely re-binds the slice of the composite service from the point of failure or the node in which the QoS deviation is discovered. Hence, it just changes the non-executed services. Looking backwards through the structure of the composite, i.e. even the well-executed services, it is probable that the approach finds a better substitution solution in terms of QoS.

- The re-binding will not change the structure of the composite service and it simply replaces the instances. The advantage is that the business structure is saved; however, the probability of recovery will be increased by checking for a new structure, i.e. a new workflow for the slice. In such cases not only will the QoS be changed, but also the quantity of the services may change.

2.5.6 Dynamic Substitution

Möller & Schuldt (2010) dynamically modify the control flow of a composite service at run-time in case of a failure. The approach is motivated by a demand to handle the failure of a composite service without the need to be aware of all available services at design time. They provided a forward-oriented failure recovery strategy in which they automatically substitute a set of services by another semantically equivalent set. The strategy is called *Control Flow Intervention* (CFI).

Firstly, the authors formally defined the service profile and the process model of the services similar to OWL-S. Then, they used the initiated idea of subgraph replacement of a composite's process model to survive the execution of the failed composite. A subgraph of the original control flow is substituted by a semantically equivalent set of services. For the matchmaking purposes of the subgraphs, they use "bijective mappings" between Input and Output, and Precondition and Effect (IOPE) pairs of the subgraphs. Actually, Precondition is excluded because they claimed that two services may semantically be similar having different preconditions. That is to say, the subgraph pairs are said to be matched if their number of functional properties are equal and each property of a subgraph is mapped to a property of the other subgraph.

Merits and Limitations

- The evaluation of the work represents a low time complexity for the replacement which shows a strength of the subdigraph replacement approach.
- They claimed the effectiveness of the approach for all possible mapping cardinalities of the replacements, i.e. the subgraphs' sizes including 1:1, 1:n, n:1, and n:m; however, they evaluate the approach just for 1:1 replacement.
- The replacement used in the approach does not move backwards through the structure to find a better so-called *initial place*. Therefore, even though they use the **effects** property of the services for the matchmaking purposes, the world-altering services are not considered.
- The determination of the *final* point for the "to be replaced" subgraph is not explicitly stated even though it is crucial (It is not required in their 1:1 replacement of the practiced evaluation).
- The approach does not support non-functional properties for the replacements, hence the QoS deviation is not investigated.

2.5.7 Performance Prediction

Dai et al. (2009) presented a self-healing solution for composite services using a performance prediction method. They integrated the ideas of backup in selection and reselection in execution. The problem they claimed to solve is twofold. First, the time complexity of a reselection of a service class and its successors is high. Second, a sub-optimal replacement composite service backed up in the selection may also not be available with its initial QoS. The pivotal idea of the approach is performance prediction. The component services' violation of QoS is predicted so that the reselection is done before the

failure. Hence, the composite service is repaired before its invocation.

A Semi-Markov Process (SMP), an extension of Markov's is used for the speed of data transmission. The SMP models time-dependent stochastic behaviors. The approach predicts if there is a deviation in QoS particularly in data transmission speed from its estimated speed. The predicted unavailable (or unpromising) service is re-selected and in case that an alternative does not exist, the backed up services would be replaced. The re-selection and replacement is triggered to be performed before the invocation of the composite service based on its calculated reliability. Therefore, the composite service will switch to the repaired structure and automatically will heal itself.

Merits and Limitations

- Predicting the QoS deviations helps the composite service to be repaired even before its execution which decreases the delay.
- The approach is applied to one metric of QoS. However, the other metrics should be considered as well.
- There is no solution for the replacement of functionally different services.
- Semantics of the services' functionality is not contemplated.
- The approach does not go backwards through the structure of the composite service, so the effects of the world-altering services are not considered.

2.6 Summary: Inferences from Literature Reviewed

In this chapter we reviewed the approaches to the failure recovery of Web services. The approaches were categorized based on the cardinality of two sets of services, the set of services to be taken from the structure of composites and the set of services to be placed.

Table 2.1 shows a summary of the most related approaches to our research problem. It identifies the main idea and the contributions of the researches, their binding methods, consideration of world-altering services and compensation of their effects, replacement types, and the use of semantics to enhance the automation of recovery. The comparison (other than replacement types) embraces the following main factors:

- **Binding method**: The method in which the services are represented in composition structures, i.e. *Static* (for concrete service) or *Dynamic* (for abstract service) Binding.
- **World-altering** services: The method considers existence of the world-altering services in the composite services.
- **Semantics**: The method uses semantics in the specification of the services, their mediation, and in the replacement strategy.

Table 2.1.: The Most Related Researches' in Chronological Order

Paper	Idea	Binding	Considering World-altering Services	Atomic Replacement	Atomic to Composite Replacement	Composite to Composite Replacement	Semantics
(Yu & Lin, 2005a)	Backup path	Abstract	✗	✓	✗	✓	✗
(Chafle et al., 2006)	Multi stage adaptation	Abstract	✗	✓	✓	✓	✗
(Feng et al., 2007a,b)	Partial Re-composition	Abstract	✗	✓	✗	✓	✗
(Vaculín et al., 2008),							
(Wiesner et al., 2008)	1-to-1 Replacement	Concrete	✓	✓	✗	✗	✓
(Canfora et al., 2008)	Rebinding	Abstract	✗	✓	✗	✓	✗
(Dai et al., 2009)	Performance Prediction	Abstract	✗	✓	✗	✓	✗
(Lin et al., 2010)	- Region Reconfiguration - Llama Middleware	Concrete	✗	✓	✓	✓	✗
(Möller & Schuldt, 2010)	Dynamic Substitution	Concrete	✗	✓	✓	✓	✓

Furthermore, Table 2.2 shows the failure recovery approaches particularly for OWL-S services. This table also distinguishes the main idea and the contributions of the researches, consideration of world-altering services and compensation of their effects, replacement types, and QoS aspects.

The approaches were investigated from the aspect of evaluation purposes. We have seen that none of the methods were evaluated on a single data set due to the unavailability of such a test data. All the approaches were tested on a small set of manually generated composite services or a set of randomly generated composites. This generation of synthetic services does not allow for comparisons of the methods. The test collections used for evaluation purposes of the investigated methods are shown in Table 2.3.

Even though there are some test collections of semantic Web services such as OWL-S Service Retrieval Test Collection (OWLS-TC), SWS Test Collection (SWS-TC), SAWSDL Service Retrieval Test Collection (SAWSDL-TC), and WSMO-Lite Test Collection (WSMO-Lite-TC), there is no data set for the composite (semantic) Web service. The

Table 2.2.: Failure Recovery Approaches for OWL-S

Research Paper	Idea	Considering World-altering Services	Atomic Replacement	Atomic to Composite Replacement	Composite to Composite Replacement	QoS Consideration
(Vaculín et al., 2008)	- Exception Handlers - Multiple Recovery Actions	✓	✓	✗	✗	✗
(Wiesner et al., 2008)	- Enhanced Recovery Actions - Automatic Compensation	✓	✓	✗	✗	✗
(Möller & Schuldt, 2010)	- Dynamic Substitution - IOPE Matchmaking	✗	✓	✓	✓	✗

Table 2.3.: Evaluations of Failure Recovery Approaches

Research Paper	Test Collection
(Yu & Lin, 2005a)	Randomly Generated
(Chafle et al., 2006)	Randomly Generated
(Feng et al., 2007a,b)	Randomly Generated, using *nem* (network emulator)
(Canfora et al., 2008)	Two manually made composite services (9 and 6 services)
(Dai et al., 2009)	Randomly Generated
(Lin et al., 2010)	A business process with 49 nodes and 8 parallel structures for simulation
(Möller & Schuldt, 2010)	- 10+4 OWL-S Ontologies - Synthetically generated service specifications
(Angarita et al., 2012)	Synthetic Data sets comprised by 800 WSs with 7 replicas each

use of a randomly generated set of services among all the researchers proves the lack of a standard test collection of **composite** Web services in the field.

The literature review in this chapter suggests that:

- It is better to use a combination of replacement methods, i.e. one-to-one, one-to-many, and many-to-many replacements, to enhance the probability of a feasible recovery.
- In order to find the best set of services to be replaced, the approach should go backwards through even the well-executed services.
- The method must minimize the required calculations at a failure time so that the repairing delay time is decreased.
- The semantics of the services must be contemplated for an accurate replacement of the services in terms of their functionality. Moreover, the approach should not be restricted to a particular formalism.
- The world-altering category of services must be considered and their effects on the knowledge world and the real world must be contemplated.

2.7 Problem Identification: Problems Identified and Conclusions

The results of the tests on existing methods and inferences drawn from the literature reviewed revealed that four fundamental problems appear to exist.

- The lack of a test collection stays on top which is a fundamental problem shared among all the methods and needs to be dealt with. The test collection needs to support semantics of the services and to be built using both categories of Web services, information-providing and world-altering, to enable the approaches to examine both. Additionally, it has to be big enough so that tests such as probability checking of the failure recovery methods can be performed.
- The failure recovery probability of existing methods particularly the well-known and often-used replacement, Atomic Replacement (one-to-one), is low. Hence, it needs to be tackled and increased to enable a more reliable service-based system.
- The proposed method must contemplate world-altering category of Web services.
- In addition to the accuracy of the adapted service in terms of functional properties, its non-functional properties, i.e. QoS, must be considered in order not to have a major change after an adaptation.

Chapter 3: Subdigraph Renovation Plan for Failure Recovery

In the usual case, a user asks a service mediator in the Service Oriented Architecture (SOA) for a Web service to carry out some tasks. The user can even be a software agent which requests a specific service. The service mediator searches its own service registry (local or remote) to discover a service that matches the user's request (Erl, 2007; W3C Working Group, 2004). According to the first semantic service's matching proposal, there are four conceivable levels of match: exact, subsume, plug-in, and fail (Paolucci et al., 2002). The service with the highest matching requirements is selected.

The mediator orchestrates the execution of the discovered service to ensure the achievement of the goal for the user. Business processes which undertake the implementation of the Web services are typically long-lasting and susceptible to fail. In case of a failure the mediator exchanges the jammed service with a matched service. Discovering the alternative service is similar to finding the first service, unless it chooses the second topmost service available.

In more complex cases, if the mediator is unable to discover at least a plug-in match for the request, a composer bundles smaller Web services to create a value-added service (Medjahed, 2004). Some of these smaller services can be other composite services as well. The composer stores the synergy service in a composite service registry for future uses.

The execution of the newly generated service structure depends on the correct execution of all sub-processes. These smaller services have the contingency of a failure as well. Moreover, due to the dependency of the components on each other, the probability of the failure of the system is higher. Therefore, a mechanism is required to ensure that the

running process is not interrupted. This strategy should quickly and efficiently exchange the component service, which has failed or became unavailable, with a similar service.

If a failure occurs for any of the constituent services of the composite service, the mediator is responsible to switch to an alternative service like the action taken for the atomic service failure. Moreover, the effects made by the previous executed Web services of the construct need to be undone.

3.1 Formal Definitions

The following are some definitions of terms used in this research. The definitions are mostly adapted from the works by Möller & Schuldt (2010), Yu & Bouguettaya (2008), and Van Riemsdijk & Wirsing (2007).

Definition 1. ***Functional Properties:*** *The set of functional properties of a service $\mathcal{S}$ is a four-tuple:*

$$\mathcal{S} = \langle \mathcal{I}, \mathcal{O}, \mathcal{P}, \mathcal{R} \rangle \tag{3.1}$$

In Equation 3.1, $\mathcal{I}$ is the set of Inputs, $\mathcal{O}$ is the set of Outputs, $\mathcal{P}$ is the set of Preconditions, $\mathcal{R}$ is the set of Results (Effects). The cardinality of each of these sets is greater than or equal to zero. $|x|$ is the cardinality of the set x:

$$|I| \geq 0 \tag{3.2}$$

$$|O| \geq 0 \tag{3.3}$$

$$|P| \geq 0 \tag{3.4}$$

$$|R| \geq 0 \tag{3.5}$$

The inputs and the outputs are the information-transformation done by the service. The precondition is the condition which must be true before the execution of the service. The results, which are sometimes called effects, are the changes made in the world through the correct execution of the service.

The only restriction for the cardinalities is that every service must have at least one output or a result, i.e. the sum of the number of outputs and the results must be greater than 1 as shown in Equation 3.6.

$$(|\mathcal{O}| + |\mathcal{R}|) \geq 1 \tag{3.6}$$

Although, the overall approach of our failure recovery method is not restricted to a specific formalism, we use definitions that are similar to a Semantic Web Service (SWS) description language called OWL and Services (OWL-S) (Martin et al., 2007). We use these definitions due to the similarity of our needed definitions to the latest pre-release (1.2) of the language (Martin et al., 2006). However, any formalism, i.e. modeling language, which has the definitions of inputs, outputs, preconditions and results such as Web Service Modeling Ontology (WSMO) language, is applicable and can be used.

Definition 2. ***Non-Functional Properties:*** *We define the set of non-functional properties $\mathcal{N}$ of a service $\mathcal{S}$ as a four-tuple:*

$$\mathcal{N} = \langle \mathcal{ET}, \mathcal{EC}, \mathcal{UC}, \mathcal{R} \rangle \tag{3.7}$$

This set contains the execution time ($\mathcal{ET}$) and cost ($\mathcal{EC}$) of the service, the undo cost ($\mathcal{UC}$) needed to compensate the service effects. $\mathcal{R}$ is the historical reliability of the service in previous iterated executions.

- *The* execution time *of the service is measured by the mediator which is the mean interval of the execution of the service during its life time.*
- *The provider of the service denotes the* execution cost *of the Web service. The provider of the service himself or a third-party supporter of the service regulates the cost (*undo cost*) used to compensate all the effects made by the service.*
- *The reliability of the service is usually measured as a real number* $[0..1]$*. The value is governed by the service executor component of the mediator and it is the average*

value of the ability of the service to respond. It is increased by every successful execution of the service, and decreased by failure occurrences. The value can easily be calculated by a simple formula shown in Equation 3.8.

$$\frac{Number\,of\,Successful\,Executions}{Total\,Executions} \tag{3.8}$$

Although we use the simple formula; however, the value of the reliability of the service can be identified by more complex formulae considering other criteria, e.g. time.

Definition 3. ***Atomic Web Service:*** *An atomic Web service is a Web service w which has some of the functional properties of a Web service and is not breakable into two or more smaller services. The web service also has non-functional properties. This is the core functionality of a service-based system, i.e. the atomic Web service does the actual work (or part of it) for the user.*

Definition 4. ***A Graph of a Web Service:*** *The structure of the Web service is shown as a directed graph (hereafter, digraph (Bang-Jensen & Gutin, 2009)). Our approach of graph representation is motivated and adapted from Hashemian & Mavaddat (2005), and Yu & Lin (2005a).*

Every graph of service(s) contains some finite vertices which are the services, and some edges. Every edge e in the graph is a triple: $e = \langle w_t, w_h, c \rangle$ *where e is a directed edge from vertex (service)* w_t *to* w_h (w_t *dominates* w_h *(Bondy & Murty, 2008)). According to Equation 3.6, the minimum out-degree of the vertices is one, i.e. every Web service must have at least an output or a result (for graph D,* $\delta^+(D) = 1$ *(Bang-Jensen & Gutin, 2009)). c is an ontology concept used as the label of e. This concept is pre-defined in some shared ontologies among all the graphs of services in the repository.*

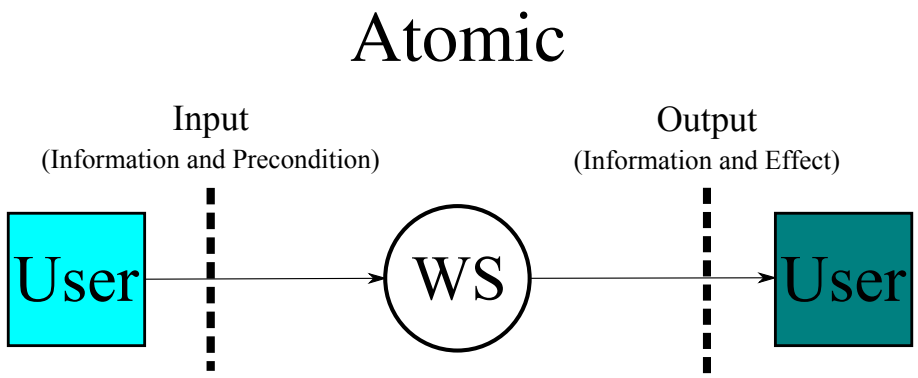


Figure 3.1.: A Graph of an Atomic Web Service

An example of such a digraph of an atomic Web service is given in Figure 3.1. Moreover, Figures 3.2, 3.3, and 3.4 show other examples of digraphs of (composite) Web services.

Neither w_t nor w_h can be null, i.e. there is no orphan edge in our graphs. Moreover, c cannot be null, i.e. the concept which is used to label the edge must be explicitly specified.

We define three functions to access the vertices and labels of the graphs:

- ***tail(e)** returns the vertex w_t.*
- ***head(e)** returns the vertex w_h.*
- ***concept(e)** returns the label of the edge, i.e. the ontology concept c.*

Definition 5. ***Composite Web Service:** A composite Web service $\mathcal{S}$ is a seven-tuple:*

$$\mathcal{S} = \langle \mathcal{W}, \mathcal{E}, I, O, \mathcal{P}, \mathcal{R}, \mathcal{N} \rangle \tag{3.9}$$

$\mathcal{S}$ is modeled as a directed, connected, labeled (both edge-labeled, and vertex-labeled) graph. A $User_{Input}$ vertex (virtual source) and a $User_{Output}$ vertex (target node) are added to the graph (Yu & Lin, 2005a). Figure 3.2 depicts a sample composite graph.

- *$\mathcal{W}$ is a finite set of vertices. $\mathcal{W} = \{w_1, w_2, w_3, ..., w_n\}$.*

  w_i is a Web service, either an atomic Web service or another composite Web service. Although, a constituent of a composite service can contain more than one atomic service, we consider the smaller composite service as a single service. Hence, the order of the graph of the whole composite service $\mathcal{S}$, i.e., the cardinality of the set of

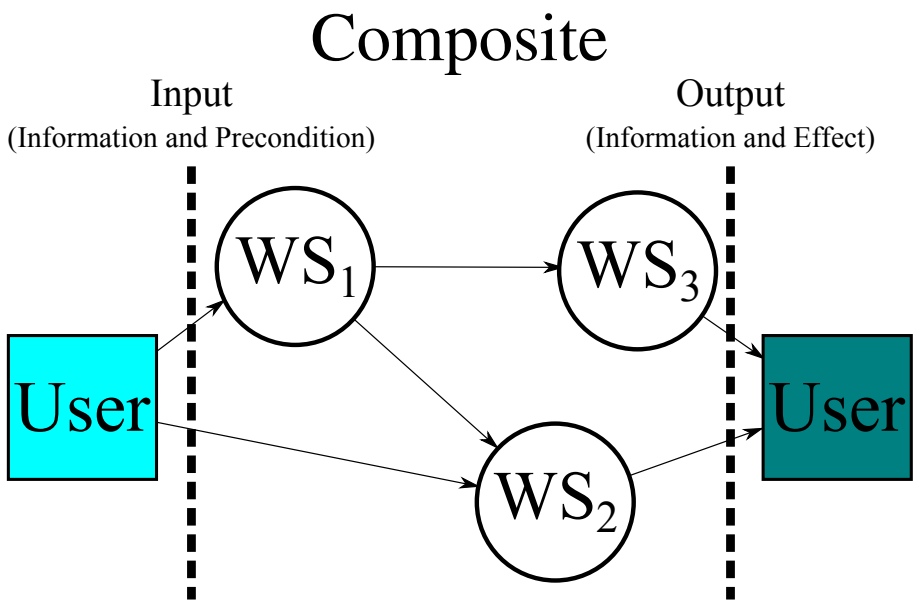


Figure 3.2.: A Graph of a Composite Web Service

component services is n ($|\mathcal{W}| = n$). Service names are the labels of the vertices.

$|\mathcal{W}| \geq 1$. *if* $|\mathcal{W}| = 1$ *then the service is an atomic service (Figure 3.1).*

- $\mathcal{E}$ *is a union of all of the functional properties of the Web services in* $\mathcal{W}$*. The functional properties make the edges of the digraph and some of the edges link* $w_1, w_2, \ldots$ *together. The other edges that do not link the Web services are the ones which define the functional properties of the composite Web service. The ontology concepts used for the functional properties are the labels of the edges.*
- *Functional properties* $\mathcal{I}, \mathcal{O}, \mathcal{P}, \mathcal{R}$ *are as follows.*
  - $\mathcal{I}$ *is the Inputs set of the composite service* $\mathcal{S}$ *(*$\mathcal{I} \subset \mathcal{E}$*).*

    $\forall e \in \mathcal{I}, tail(e) = User_{Input}, head(e) \in W$
  - $\mathcal{O}$ *is the Outputs set of the composite service* $\mathcal{S}$ *(*$\mathcal{O} \subseteq \mathcal{E}$*).*

    $\forall e \in \mathcal{O}, tail(e) \in W, head(e) = User_{Output}$
  - $\mathcal{P}$ *is the Preconditions set of the composite service* $\mathcal{S}$ *(*$\mathcal{P} \subset \mathcal{E}$*).*

    $\forall e \in \mathcal{P}, tail(e) = User_{Input}, head(e) \in W$
  - $\mathcal{R}$ *is the Results (Effects) set of the composite service* $\mathcal{S}$ *(*$\mathcal{R} \subseteq \mathcal{E}$*).*

    $\forall e \in \mathcal{R}, tail(e) \in W, head(e) = User_{Output}$

$\mathcal{I}$, and $\mathcal{P}$ cannot be equal to $\mathcal{E}$ because the composite service must have at least one output or a result (Equation 3.6).

- $\mathcal{N}$ is a set of non-functional properties of the composite service $\mathcal{S}$. The following calculations are adapted from the well-known works presented by Zeng et al. (2003), Zeng et al. (2004), Cardoso et al. (2004), Jaeger et al. (2005), Chafle et al. (2006), and Canfora et al. (2008).
  - The execution time ($\mathcal{ET}$) of the composite service $\mathcal{S}$ is the sum of the execution times of the constituent services in $\mathcal{W}$.
  $$\mathcal{ET}_{\mathcal{S}} = \sum_{i=1}^{|\mathcal{W}|} \mathcal{ET}_{w_i} \tag{3.10}$$
  - The execution cost ($\mathcal{EC}$) of the composite service $\mathcal{S}$ is the sum of the execution costs of the constituent services in $\mathcal{W}$.
  $$\mathcal{EC}_{\mathcal{S}} = \sum_{i=1}^{|\mathcal{W}|} \mathcal{EC}_{w_i} \tag{3.11}$$
  - The undo cost ($\mathcal{UC}$) of the composite service $\mathcal{S}$ is the sum of the undo costs of the constituent services in $\mathcal{W}$.
  $$\mathcal{UC}_{\mathcal{S}} = \sum_{i=1}^{|\mathcal{W}|} \mathcal{UC}_{w_i} \tag{3.12}$$
  - The reliability ($\mathcal{R}$) of the composite service $\mathcal{S}$ is the multiplication of the reliability values of the constituent services in $\mathcal{W}$.
  $$\mathcal{R}_{\mathcal{S}} = \prod_{i=1}^{|\mathcal{W}|} \mathcal{R}_{w_i} \tag{3.13}$$

The direction of the edges $\mathcal{E}$ in the graph $\mathcal{S}$ is from the services which generate the output (concepts) to the services that use these outputs (as their inputs). Similarly, the direction of the edges is from the services generating the effects to the services which use them as their preconditions. There are edges which do not join two of the services together, and only have one side incident to $User_{Input}$ or $User_{Output}$. These edges are $\mathcal{I} \cup \mathcal{O} \cup \mathcal{P} \cup \mathcal{R}$ and they make the inputs, the outputs, the preconditions, and the results of the composite

service $\mathcal{S}$.

By the term "connected" we mean that the underlying graph of $\mathcal{S}$ is a connected graph (Bondy & Murty, 2008).

We assume that the composite Web service's graph is a DAG in which:

1. *There is no loop in the graph, i.e., none of the edges joins a service to itself directly. If a loop in the structure of the composite service is needed, it has been proposed to convert its graph to a DAG by unfolding after estimating the number of iterations (Zeng et al., 2004). Moreover, Canfora et al. (2008) unloop such a graph to a DAG by multiplying the abstract nodes to a weight which is the estimated iteration count of the loop.*
2. *There is no cycle (including any finite number of the services) from a service $\mathcal{W}$ to itself. In case that a composite Web services needs the execution of a service more than once, several instances of the same service (with variant names) are added to the graph as different vertices.*

Example 1. *As an example for the definition of a composite Web service (Definition 5), we define a sample composite SWS. Figure 3.3 represents a composite SWS with four constituent services:*

- $\mathcal{W} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$
  - $\mathcal{A} = \{I_{\mathcal{A}}, O_{\mathcal{A}}, \mathcal{P}_{\mathcal{A}}, \mathcal{R}_{\mathcal{A}}\}$
  - $\mathcal{B} = \{\{I1_{\mathcal{B}}, I2_{\mathcal{B}}\}, \{O1_{\mathcal{B}}, O2_{\mathcal{B}}\}, \{\}, \{\}\}$ *:: no precondition and result*
  - $\mathcal{C} = \{I_{\mathcal{C}}, O_{\mathcal{C}}, \{\}, \{\}\}$ *:: no precondition and result*
  - $\mathcal{D} = \{I_{\mathcal{D}}, \{\}, \{\}, \mathcal{R}_{\mathcal{D}}\}$ *:: no output and precondition*
- $\mathcal{E} = \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{D} =$

  $\{I_{\mathcal{A}}, O_{\mathcal{A}}, \mathcal{P}_{\mathcal{A}}, \mathcal{R}_{\mathcal{A}}, I1_{\mathcal{B}}, I2_{\mathcal{B}}, O1_{\mathcal{B}}, O2_{\mathcal{B}}, I_{\mathcal{C}}, O_{\mathcal{C}}, I_{\mathcal{D}}, \mathcal{R}_{\mathcal{D}}\}$

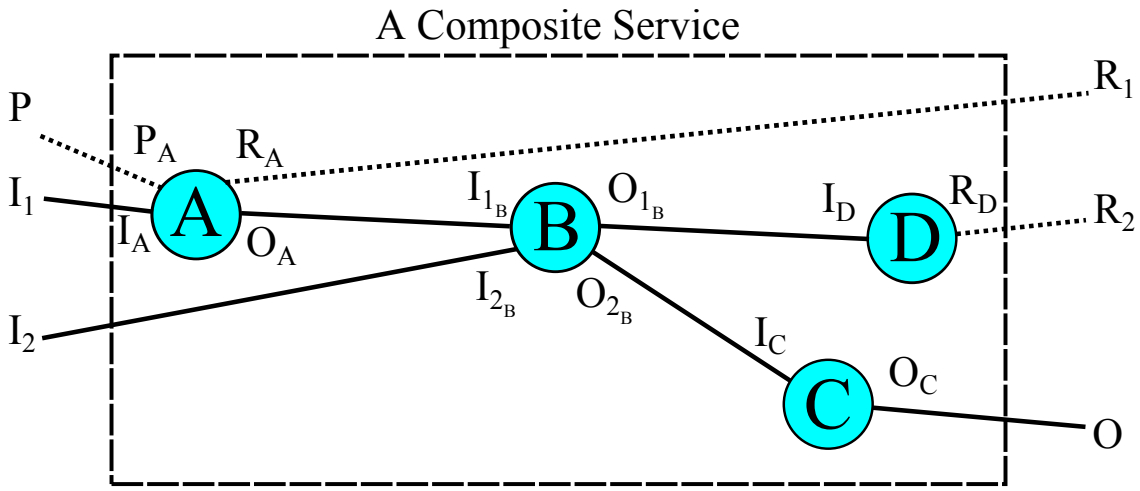


Figure 3.3.: A Composite Semantic Web Service Containing Four Smaller Services

- $I = \{I_{\mathcal{A}}, I2_{\mathcal{B}}\}$, $I_1 = I_{\mathcal{A}}$, $I_2 = I2_{\mathcal{B}}$
- $O = \{O_C\}$
- $\mathcal{P} = \{\mathcal{P}_{\mathcal{A}}\}$
- $\mathcal{R} = \{\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{D}}\}$, $\mathcal{R}1 = \mathcal{R}_{\mathcal{A}}$, $\mathcal{R}2 = \mathcal{R}_{\mathcal{D}}$

*In the example, $\mathcal{A}$ and $\mathcal{D}$ are **world-altering** services, and $\mathcal{B}$ and $\mathcal{C}$ are **information-providing** services (McIlraith et al., 2001).*

Assumption 1. *The actual digraphs of atomic and composite services are as represented on the left side of Figure 3.4. We intend to symbolize the points of functionality as the vertices in the graphs. So, sometimes for the sake of brevity and clarity we omit the representation of the user vertex in both sides (as shown in the right graphs of the figure), regarded as the **source**, and the **sink** of the graph.*

Assumption 2. *The users in Figure 3.4 before and after the execution are the same but because the situation (the information and conditions available before and after the execution) changed, we separated these two vertices (Yu & Lin, 2005a).*

Definition 6. ***Induced Subdigraph:*** *An induced subgraph of a digraph $\mathcal{G}$ (hereafter, subdigraph (Bang-Jensen & Gutin, 2009)) is a digraph whose vertex set is a subset of $\mathcal{G}$, and*

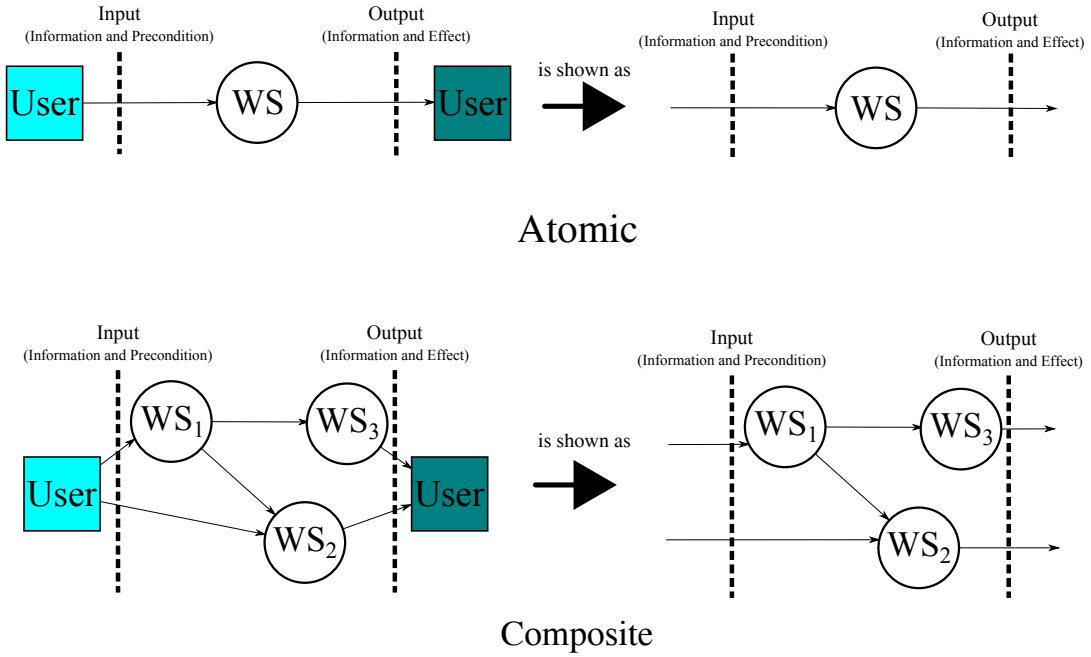


Figure 3.4.: Actual Digraphs of an Atomic and a Composite Service

whose adjacency relations set (edges) is a subset of the edges in $\mathcal{G}$ *restricted to this subset (Bondy & Murty, 2008). In our definitions, a subdigraph of the whole composite SWS* $\mathcal{S}$ *is another composite SWS,* $\mathcal{S}' = \langle \mathcal{W}', \mathcal{E}', \mathcal{I}', \mathcal{O}', \mathcal{P}', \mathcal{R}', \mathcal{N}' \rangle$ *with the following additional properties rather than the usual subdigraph definitions:*

- $\mathcal{W}' \subseteq \mathcal{W}$. $\mathcal{S}'$ *has at least one of the services of* $\mathcal{S}$. $|\mathcal{W}'| > 0$, *so, we do not admit the "null" graph as a subgraph of* $\mathcal{S}$. *If any service* $w \in \mathcal{W}'$ *then all of the functional properties of* w *are members of* $\mathcal{E}'$.
- $\mathcal{S}'$ *is a connected subdigraph of the digraph* $\mathcal{S}$, *i.e., there is at least one path from every vertex to the others in the underlying graph of* $\mathcal{S}$.
- $\mathcal{E}' \subseteq \mathcal{E}$. $\mathcal{E}'$ *has some of the edges of the digraph* $\mathcal{S}$. *The subdigraphs used are induced subgraphs, i.e. edges in the subgraph include* all *linking edges of the graph* $\mathcal{S}'$, *which link services* $\mathcal{W}'$ *and exist in* $\mathcal{S}$. *So, if any edge exists in* $\mathcal{S}$, *linking two of its services, e.g.* $\mathcal{A}$ *and* $\mathcal{B}$, *and if* $\mathcal{A}$ *and* $\mathcal{B}$ *are both in* $\mathcal{S}'$ *then this linking edge must be in* $\mathcal{S}'$.

$$\forall w_t \in \mathcal{W}, \forall w_h \in \mathcal{W}, e = \langle w_t, w_h, c \rangle, \ \ if \ \ e \in \mathcal{E} \wedge w_t \in \mathcal{W}' \wedge w_h \in \mathcal{W}' \Rightarrow e \in \mathcal{E}'$$

- *labels of the edges (ontology concepts) in $\mathcal{S}'$ are exactly the same labels as in $\mathcal{S}$. One of our future work is to consider sub-concept and super-concept for the subdigraph edge labels.*
- *$\mathcal{I}' \subset \mathcal{E}$, $\mathcal{O}' \subset \mathcal{E}$, $\mathcal{P}' \subset \mathcal{E}$, $\mathcal{R}' \subset \mathcal{E}$. There is no determined relation between $\mathcal{I}, \mathcal{O}, \mathcal{P}$, and $\mathcal{R}$ and $\mathcal{I}', \mathcal{O}', \mathcal{P}'$, and $\mathcal{R}'$ respectively, i.e. they may have similar elements.*
- *$\mathcal{N}'$ is calculated separately and the values of $\mathcal{N}$, and $\mathcal{N}'$ are not equal.*

Definition 7. ***Exact compatibility ($\bowtie$):*** *Two digraphs of the composite semantic Web services are exactly compatible if the number of functional properties and even the concepts used for their functional properties are equal ($\doteq$). We use the symbol ($\doteq$) to show the equality of two sets and ($\neq$) for their inequality.*

$$\mathcal{S}' = \langle \mathcal{W}', \mathcal{E}', \mathcal{I}', \mathcal{O}', \mathcal{P}', \mathcal{R}', \mathcal{N}' \rangle \tag{3.14}$$

$$\mathcal{S}'' = \langle \mathcal{W}'', \mathcal{E}'', \mathcal{I}'', \mathcal{O}'', \mathcal{P}'', \mathcal{R}'', \mathcal{N}'' \rangle \tag{3.15}$$

$$\mathcal{S}' \overset{IOPR}{\bowtie} \mathcal{S}'' \Leftrightarrow (\mathcal{I}' \doteq \mathcal{I}'') \wedge (\mathcal{O}' \doteq \mathcal{O}'') \wedge (\mathcal{P}' \doteq \mathcal{P}'') \wedge (\mathcal{R}' \doteq \mathcal{R}'') \tag{3.16}$$

The sets of services $\mathcal{W}'$ and $\mathcal{W}''$ may be different in cardinality, and in functionality. So, $\mathcal{E}'$ and $\mathcal{E}''$ might be different as well.

The exact compatibility, $\overset{IOPR}{\bowtie}$, refers to the Input and Output, and Precondition and Result (IOPR) exact compatibility which considers all functional properties of the services. However, for some reasons we sometimes need Input and Output (IO) exact compatibility, $\overset{IO}{\bowtie}$, which is defined as follows:

$$\mathcal{S}' \overset{IO}{\bowtie} \mathcal{S}'' \Leftrightarrow (\mathcal{I}' \doteq \mathcal{I}'') \wedge (\mathcal{O}' \doteq \mathcal{O}'') \tag{3.17}$$

It is worth noting that in simple cases of IOPR matching it is possible that $\mathcal{P}' \neq \mathcal{P}''$, i.e. services (atomic or composite) may have similar functionality with different preconditions. For example there might be two selling Web services that are similar but with different precondition as follows: The first service requires the user to create an account

before his first purchase. So, its precondition is having a user account. The second one immediately creates an account if the user wants to buy for the first time. Hence, there is no precondition for the second selling Web service. However, we assume that $\mathcal{P}'$ should be equal to $\mathcal{P}''$ to cover all the anticipated cases.

The non-functional properties ($\mathcal{N}'$ and $\mathcal{N}''$) are typically different.

3.2 SRPFR: An Automatic Subdigraph Renovation Plan

We name our approach as Subdigraph Renovation Plan for Failure Recovery (SRPFR) for composite semantic Web services. We divide the plan into two distinctive phases, offline and online. The mediator of the architecture continuously executes the offline phase to prepare the requirements for the online phase. Furthermore, the online phase comprises a forward approach, and if this fails it utilizes a backward approach. The separation helps our approach to respond significantly faster at the failure time.

In the **offline** phase, which is executed continuously, we do some pre-calculations and we constantly process the incoming new descriptions of the services published in a local repository. The current research trend is to compose the Web services at run-time, so the assumption of the availability of the composite structures in a registry seems unreasonable. Nevertheless, the reason that we assume to have the structures in a registry is that in the current industry use of composite services, business experts commonly predefine the composite structures and store them for next use. In other words, enterprises prefer to examine the composite structures before deployment. Hence, consideration of a composite service registry is not illogical. This is supported by works such as (Lin et al., 2010) in which a service repository is contemplated in their framework. The **online** phase is triggered immediately at the failure time to adapt the service and restore the normal execution flow of the services.

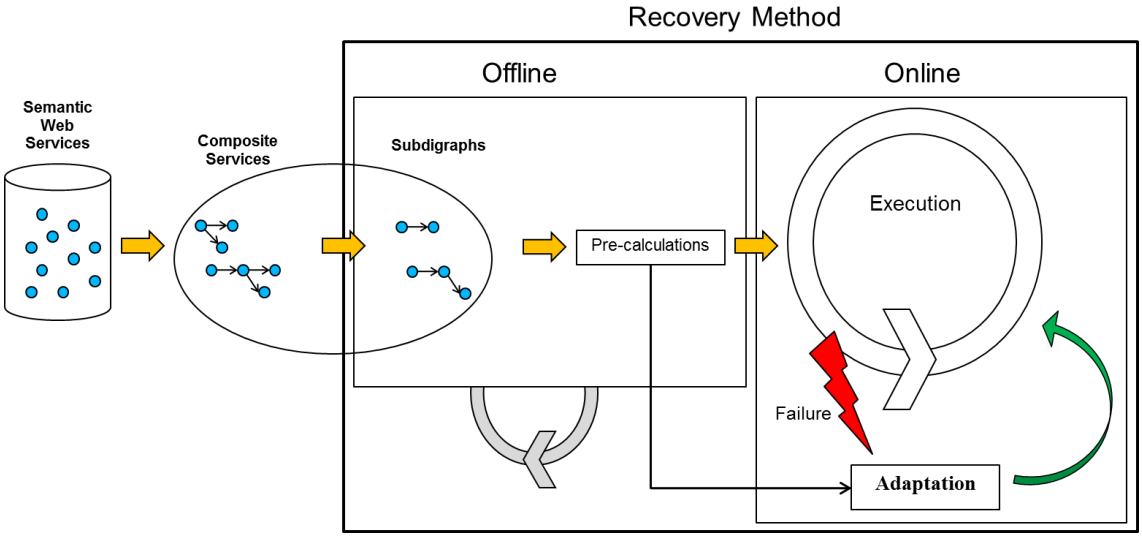
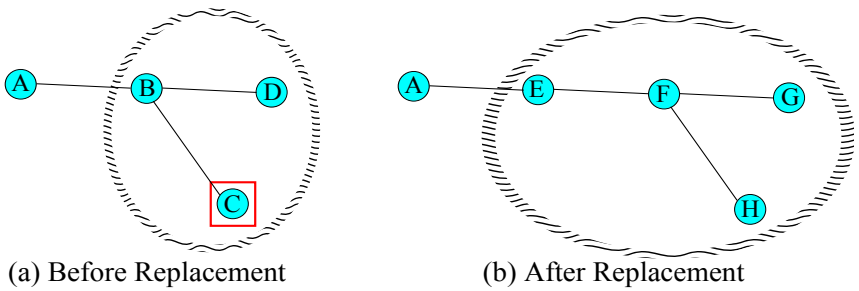


Figure 3.5.: Overview of the Recovery Method: SRPFR

Figure 3.5 depicts an overview of SRPFR. It shows the registry of semantic services. They may be combined to make the composite services. SRPFR calculates their subdigraphs and the requirements of our proposed adaptation method. The pre-calculations are done in an ongoing manner and continuously to ensure that if a failure occurs it can be recovered with a minimum delay time.

Figure 3.6 illustrates an overview of a sample replacement done by the algorithm. In the figure, Web service C is assumed to have failed and the mediator could not find any atomic replacement, or even a composite service matching it. The adapter replaces the subgraph containing $\{B,C,D\}$ services (Figure 3.6a) to the subgraph with $\{E,F,G,H\}$ services (Figure 3.6b). We suppose that the replacement is feasible due to the exact compatibility of BCD and EFGH according to Definition 7.



(a) Before Replacement (b) After Replacement

Figure 3.6.: A Sample Replacement

The following two sections elaborate the tasks done in each phase.

3.2.1 Offline Phase

The mediator has the definitions of the semantic Web services in two related service repositories, namely, the atomic service repository and the composite service repository.

The atomic service repository contains all the definitions of the services together with their functional, and non-functional properties (Definitions 3, 1, and 2 in the order given).

The composite service repository contains the composite structures (generated from the atomic services) as digraphs (Definition 5). The definitions of the atomic services in the composite structures are referred to the atomic service repository.

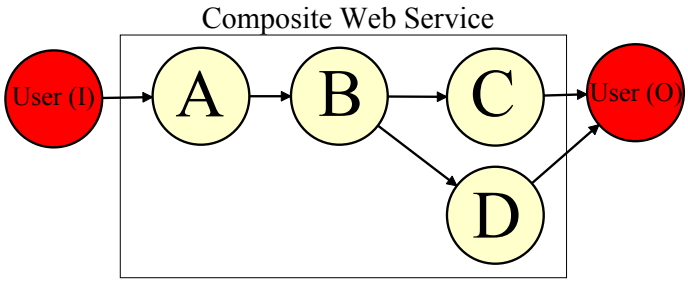
Subdigraphs Calculation

The mediator calculates all possible subdigraphs of the composite semantic Web services. These subdigraphs are also cached as separate composite semantic Web services in the repository. The calculation is a time-consuming task of our approach. Due to its high time complexity we propose the calculation to be done in an ongoing manner in which every time a new composite semantic Web service is added to the repository the task is triggered to be efficiently performed simultaneously. The calculation method for subdigraphs is elaborated in Chapter 4.

Following the calculation of subdigraphs of composite services, there are three sub-tasks as described below.

Measuring Original Subdigraphs

Having all subdigraphs of each composite semantic Web service, we build a selection list to check all subdigraphs to see how many of them can be replaced. We categorize and in-



(a) A sample composite Web service

	Web Services			
Subdigraphs	A	B	C	D
A	✓			
B		✓		
C			✓	
D				✓
AB	✓	✓		
BC		✓	✓	
BD		✓		✓
ABC	✓	✓	✓	
ABD	✓	✓		✓
BCD		✓	✓	✓
ABCD	✓	✓	✓	✓

(b) The connected subdigraphs with theirs constituent Web services' check list

Figure 3.7.: Original Subdigraphs Indexing Example

dex the subdigraphs based on the presence of a Web service in their structure. Henceforth "Original Subdigraph" means a subdigraph of the Composite Semantic Web Service.

We need the selection list because we want to know if any of the Web services fails to respond, among all "Original Subdigraphs" which of them are better to be replaced. Figure 3.7 represents an example of a composite Web service constituting four Web services with a table that lists the connected subdigraphs of this composite service. For every constituent Web service the subdigraphs containing the Web service is marked.

Henceforth, we call the Web service that we assume may fail as Assumed Failing Web Service (AFWS). We presume that every component, i.e. constituent Web service, of the

composite structure may fail with an equal probability.

In an atomic replacement approach the unique choice here is the subdigraph with a single service; however, in our approach there are several potential subdigraphs that may be possible to be replaced. Therefore, we believe that the probability of finding a replacement is increased.

All the available choices are ranked based on six criteria. The ranking result is used in the next steps. The ranking criteria are as follows:

- The *order* of the subdigraphs, i.e. the number of Web services in the subdigraph ($|\mathcal{W}|$). The fewer the number of Web services to be replaced (in case of failure) the better is the performance.
- The execution *time* and *cost* of the Web services in the subdigraph ($\mathcal{ET}$ and $\mathcal{EC}$ respectively) as in Definition 5. The higher the execution time and cost of the services the better the "Original Subdigraph" is to be nominated to be removed from the composite Web service structure. The replacement may make the composite Web service faster or cheaper respectively because it is probable that we may find faster or cheaper Web services to be substituted.
- The *number* and the *cost* of undoing the effects of the world-altering services preceding the AFWS. The structure of the composite service is available in the repository, so it is handy to count the number of world-altering services located before the AFWS. To replace a subdigraph of Web services containing the executed world-altering Web services, the mediator must pay extra cost to compensate the effects. The *information-providing* services need not be compensated because they did not make any change in the world. In order to select a subdigraph to be substituted, it is reasonable enough to choose in terms of paying less to replace, so the subdigraph

should be ranked higher.

- The *reliability* ($\mathcal{R}$) value of the subdigraph. For every subdigraph, which is also another composite Web service, we store a quality value called reliability. The value shows the historical probability of the service response to execution requests. If the reliability of the nominated subdigraph is low we rank it higher to be replaced.

Equation 3.18 shows a ranking measure called Original Subdigraph Ranking Measure (OSRM) based on the six explained criteria. For the purpose of having a normal value in the range of $[0,1]$ for the rank, we use the normalized values of the criteria (Feng et al., 2007a). The normal values are calculated based on Equation 3.19. The $Minimum(X)$ and $Maximum(X)$ are calculated for every criterion X separately on the list of available "Original Subdigraphs".

$$\begin{aligned} &\text{"}Original\ Subdigraph\text{"}\ Ranking\ Measure = \\ &\alpha_1(1-|\mathcal{W}|_{normal}) + \alpha_2\mathcal{ET}_{normal} + \alpha_3\mathcal{EC}_{normal} + \\ &\alpha_4(1-Undo_{Count_{normal}}) + \alpha_5(1-Undo_{Cost_{normal}}) + \alpha_6(1-\mathcal{R}_{normal}) \end{aligned} \tag{3.18}$$

$$X_{normal} = \frac{X - Minimum(X)}{Maximum(X) - Minimum(X)} \tag{3.19}$$

The weights in Equation 3.18 (α_1, α_2, ..., α_6) are constrained as in Equation 3.20. It is feasible that the weights are either identified by the administrator of the mediator or they could be calculated by a neural network based algorithm as has been done for similarity measurement in (Ganjisaffar et al., 2006).

$$\sum_{i=1}^{6} \alpha_i = 1, \quad 0 \leq \alpha_i \leq 1 \tag{3.20}$$

For all the subdigraphs of a composite Web service and all its AFWSs, the OSRM value

is calculated and stored for next use.

Finding Replacement Subdigraphs

Another task which should be done in parallel with the subdigraph calculation is to find the alternative subdigraphs in the repository which can potentially be executed in lieu of each subdigraph. The search is based on the exact compatibility of the subdigraphs, which is formally defined in Definition 7.

Henceforth, we call the alternative subdigraphs as "*Replacement Subdigraphs*". So, we look for "Replacement Subdigraphs" which are possible to be executed rather than "Original Subdigraph", i.e. the primary subdigraph including AFWS.

To attain a better search performance the subdigraphs are indexed based on a multi-value of their functional properties. The discovery method should preferably be a semantic approach to ensure the accuracy and correctness of the replacement. The discovery details are not within the scope of this research. Our discovery method for the experiments is described in Section 5.1.2.

Ranking "Replacement Subdigraphs"

Every composite service contains several atomic Web services. Further, each atomic service may participate in the composition structures of various digraphs of services. Moreover, every digraph or subdigraph of a composite service may have multiple alternative similar digraphs, i.e. "Replacement Subdigraphs". Both relations are a one-to-many relations. Therefore, for every atomic service there exists different related graphs. In other words, if a constituent atomic Web service of a composite Web service fails during an execution, there may exist several replacement subdigraphs.

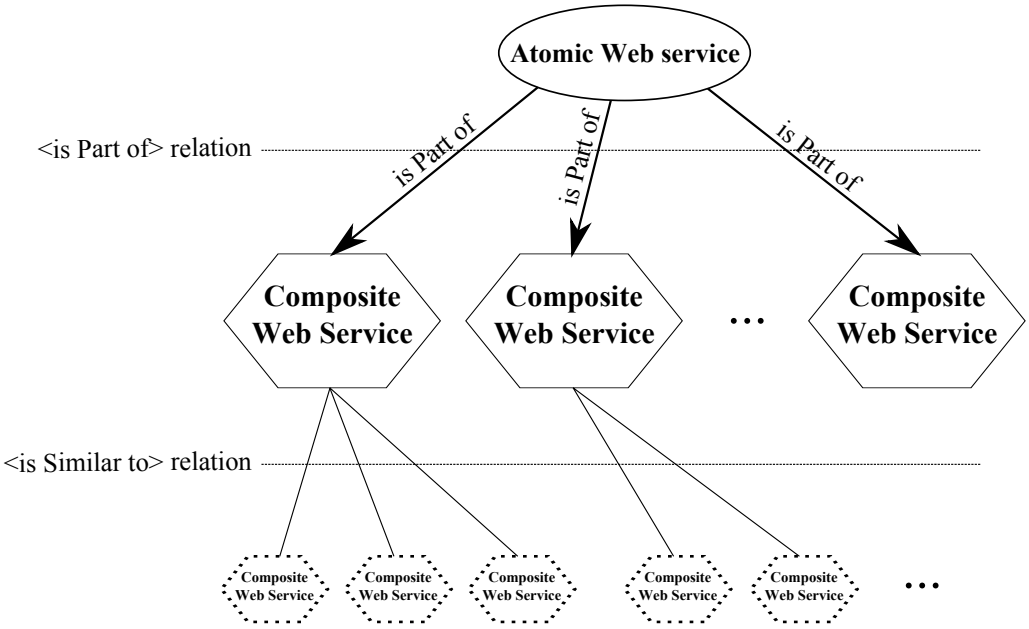


Figure 3.8.: Relation of an “Atomic Web Service” to the available “Replacement Subgraphs”

Figure 3.8 depicts the relation between an atomic service and the alternative “Replacement Subdigraphs”. The structure is a tree in which its root node is an atomic service. The nodes at depth two are the composite services which contain this atomic service (“Original Subdigraphs”). The leaves are alternative graphs of services (“Replacement Subdigraphs”), which are similar to their parent nodes. All the nodes except the root are either original composite services or the subdigraphs of the composite services which are other composite services.

In cases where several “Replacement Subdigraphs” are available for a AFWS, i.e. referring to Figure 3.8, more than one leaf node exists, we have several choices. So, the options are ranked to make the best decision. The aim of ranking is to find the best branch from the second level of the tree to its third level. Hence, if an AFWS fails, the topmost choice is at hand.

The ranking is based on four criteria as follows:

1. The order of the services in the "Replacement Subdigraph" ($|\mathcal{W}|$). The fewer services in "Replacement Subdigraph" the better.
2. The total execution time and cost ($\mathcal{ET}$, and $\mathcal{EC}$ respectively) of the "Replacement Subdigraph". The lower $\mathcal{ET}$ and $\mathcal{EC}$ the better choice it is.
3. The reliability ($\mathcal{R}$) of the "Replacement Subdigraph". The higher quality value of $\mathcal{R}$ is preferable for "Replacement Subdigraph".

Equation 3.21 shows Replacement Subdigraph Ranking Measure (RSRM) based on the four mentioned criteria. Similar to the weights in OSRM, the weights here are constrained as in Equation 3.22. The values of the weights are defined by the administrator of the architecture based on the application and the importance of the number of the services of the digraphs, the total execution time and cost, and the reliability of the services.

$$\begin{aligned} &\text{“}\textit{Replacement Subdigraph}\text{” }\textit{Ranking Measure} \; = \\ &\beta_1(1-|\mathcal{W}|_{normal})+\beta_2(1-\mathcal{ET}_{normal})+ \\ &\beta_3(1-\mathcal{EC}_{normal})+\beta_4\mathcal{R}_{normal} \end{aligned} \tag{3.21}$$

$$\sum_{i=1}^{4}\beta_i = 1, \;\; 0 \leq \beta_i \leq 1 \tag{3.22}$$

Finally, we rank and choose the best "Replacement Subdigraph" for an AFWS by combining two mentioned rank measures, i.e. OSRM, and RSRM. We combine them using Equation 3.23 and the constraint on their weights using Equation 3.24. Accordingly, the weights are either calculated systematically or defined by the administrator.

$$Rank = \gamma_1 \text{OSRM} + \gamma_2 \text{RSRM} \tag{3.23}$$

$$\gamma_1+\gamma_2 = 1, \;\; 0 \leq \gamma_1 \leq 1, \;\; 0 \leq \gamma_2 \leq 1 \tag{3.24}$$

Where OSRM and RSRM are the rankings defined in Equations 3.18 and 3.21 respectively.

At the end of the calculations we are able to find the “Replacement Subdigraph” that represents the best choice to replace the “Original Subdigraph” containing the failed service.

3.2.2 Online Phase

During the execution of a composite Web service, if any of its component services fails, it jeopardizes the completion of the whole composite service. The executor has two choices to survive the whole process, which we call “*forward approach*” and “*backward approach*”. The terms have been used similarly in other works (Tartanoglu et al., 2003; Wiesner et al., 2008; Yuan Yi, 2005).

The Forward Approach

The executor retries to execute the perished service hoping that the jammed service responds in the subsequent attempts. These attempts continues until a maximum threshold count or time is reached. This approach is similar to the *retry* exception handling strategy in (Liu et al., 2010). If this fails the backward approach is chosen.

The Backward Approach

The executor component of the architecture takes three steps to hinder the system from failure. In each step if the conditions are not fulfilled the executor moves on to the next step.

1. **Atomic Replacement**: The executor searches for possible atomic services to be used in place of the failed atomic service (Figure 3.9a). The alternative atomic service should have similar functionality to the failed service.

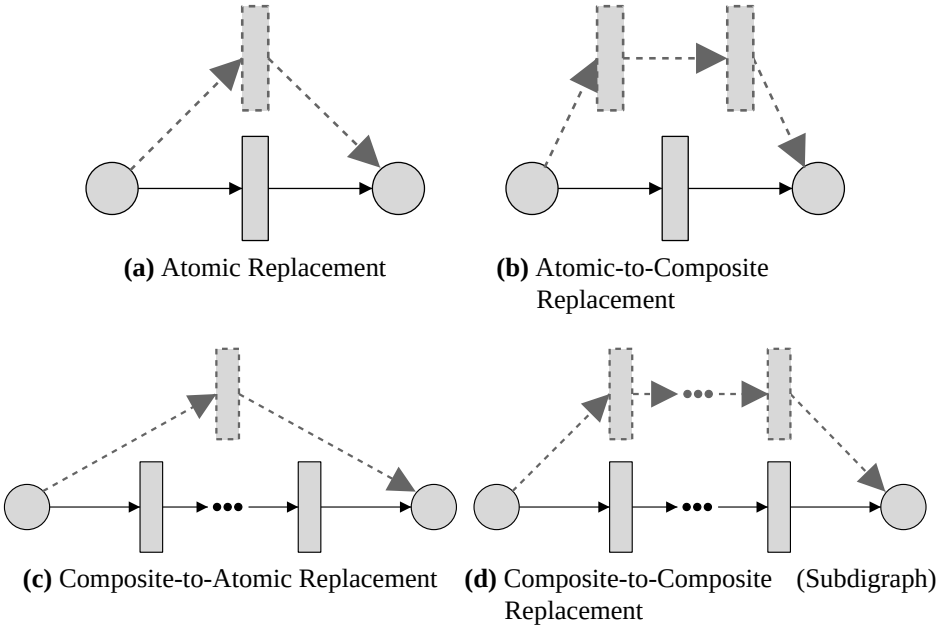


Figure 3.9.: Replacement Types (Adapted from (Möller & Schuldt, 2010))

2. **Atomic-to-Composite Replacement**: The executor searches in the composite service repository to find out which composite service can be executed instead of the perished service and fulfills the functionality of the failed service (Figure 3.9b).
3. **Subdigraph Replacement**: The executor exchanges an "Original Subdigraph" containing the failed service AFWS with a "Replacement Subdigraph" (Figures 3.9c and 3.9d). The best choice is already available from the offline phase's calculations, so in this step there is no calculation delay. The substitution in this step is not as straightforward as the previous two steps. This is due to the possibility of the presence of the *world-altering* services in the chosen "Original Subdigraph" structure preceding AFWS.

Consequently, if any world-altering Web service is executed preceding the failed service, and it is marked as a compensatable service (Bhiri et al., 2011), the undo request is sent to either the provider or a supporting third-party to undo the effects. The way each Web service is undone depends on the provider of the service, or the supporting provider; the role of the mediator is only to issue the required undo command, which is usually a

Algorithm 1. ***UndoUnusedEffects***$(Failed\,Service, OD, RD)$

Input: ***Failed Service*** *is the Web service which has failed during execution.*
Input: OD (***Original Digraph***) *is the Digraph which the Failed Service is one of its vertices.*
Input: RD (***Replacement Digraph***) *is going to be posed in place of the Original Digraph.*

{We only need to undo the well-executed World-altering Web Services}
1: $\mathcal{WES} \leftarrow Executed\,Services(World-altering\,Services(Original\,Digraph))$;
2: $To\,be\,undone \leftarrow null$;
3: ***for all*** $\mathcal{S}$ *in* $\mathcal{WES}$ ***do*** *{*$\mathcal{S}$ *is a member of the set of well-executed Web services}*
4: ***if*** $\mathcal{S} \in Replacement\,Digraph$ ***then***
5: ***if*** ($IOPR$ *of* $\mathcal{S}$) $\neq$ ($IOPR$ *of* ($\mathcal{S} \in Replacement\,Digraph$)) ***then***
6: $To\,be\,undone+ = \mathcal{S}$;
7: ***else***
8: *Flag* $\mathcal{S}$ *in Replacement Digraph as already executed;*
9: ***end if***
10: ***else***
11: $To\,be\,undone+ = \mathcal{S}$;
12: ***end if***
13: ***end for***
14: ***for all*** $\mathcal{S}$ *in* ***To be undone do***
15: *Invoke the Undo Procedure of the service* $\mathcal{S}$;
16: ***end for***

common execution of a Web service. The actual way of undoing is not within the scope of this research.

All the non-required *effects* need to be compensated. Algorithm 1 shows the way we undo the effects of the non-required services and the execution of the new services. We optimize the undo process of the well-executed world-altering Web services. We check the *Replacement Subdigraph* and if the effect will be generated again it is not undone in the first place; hence, it enhances the undo process which is usually not free in terms of cost and time.

We suppose that all the service effects are positive and not negative, i.e., they do not remove the effects from previously executed Web services. We have not considered the inter-relations between the world-altering services, i.e. if an effect of a services changes the effects generated by some other services in that particular structure of composite. The

reason for this assumption is that in current specifications of the effects for world-altering services, these kinds of relations are not given. Therefore, it is not feasible for such an approach to identify the relations. Hence, this assumption is to isolate our proposed approach and its results from this aspect.

Algorithm 2 shows a pseudo code of the online phase algorithm of SRPFR:

- Lines 1-7 show the forward approach, and if the attempt is unsuccessful the backward approach (Lines 8-20) is taken.
- A matching atomic Web service may replace the failed service to hinder the system from failure (Lines 8-10).
- If we have an available replacement for the failed service (Line 12), which was calculated in the offline phase, that particular pre-calculated best ranked subdigraph of the running composite service is removed from its structure, and its relevant replacement is implanted into the proper location of the digraph (Lines 13-16). In line 14, the algorithm calls the UndoUnusedEffects algorithm (Algorithm 1) to effectively undo the non-required services.
- At last, the executor of the architecture is informed to persevere with the execution of the composite service with no delay. To steadily decline the use of the failed service for the future, a specific non-functional property of the service, called its reliability value, is decreased in the registry (Line 21 of Algorithm 2). This ensures that new business processes will gradually stop using the failed service. If the recovery action succeeds, the execution flow is restored as if the failure did not occur.
- If the offline phase could not find an “Original Subdigraph” and its replacement for the failed service, the inability of the approach to adapt and recover the composite service is declared, which is specified in Line 18 of the algorithm.

Algorithm 2. ***FailureRecovery***$(Composite\,Structure, Failed\,Service)$

Input: ***Composite Structure*** *is the structure of the original composite Web service.*
Input: ***Failed Service*** *is the Web service which has failed during execution.*

*{**Forward Approach**}*
1: $Start\,Time \leftarrow Current\,Time;$
2: $Attempts \leftarrow 0;$
3: ***repeat***
4: ***if*** $Execute(Failed\,Service)$ *is successful* ***then***
5: *return;*
6: ***end if***
7: ***until*** $((Current\,Time > (Start\,Time + Max_{Time}))$ *or*
$(++Attempts > Max_{Re-execution}));$

*{**Backward Approach**}*
8: $\mathcal{AR} \leftarrow Atomic\,Replacement(Failed\,Service);$
9: ***if*** $\mathcal{AR}$ *is not empty* ***then***
10: $EnqueueExecution(\mathcal{AR});$
11: ***else***
12: ***if*** $Replacement\,Is\,Available(Composite\,Structure, Failed\,Service)$ ***then***
13: $\mathcal{OG} \leftarrow Get\,Best\,Calculated\,Original\,Digraph($
$Composite\,Structure, Failed\,Service);$
14: $\mathcal{RG} \leftarrow Get\,Best\,Calculated\,Replacement\,Digraph($
$Composite\,Structure, Failed\,Service);$
15: $\mathbf{Undo\,Unused\,Effects}(Failed\,Service, \mathcal{OG}, \mathcal{RG});$
16: $Enqueue\,Execution(\mathcal{RG});$
17: ***else***
18: *Recovery is unsuccessful;*
19: ***end if***
20: ***end if***

{Reliability values of all graphs containing the "Failed Service" is decreased}
21: *DecreaseReliabilityValue(Failed Service);*

- In algorithm 2, steps 2, and 3 of the backward approach are not distinguished because the replacement finding process is similar in both steps and the discrepancy is their graph size of "Original Subdigraph". So, by performing step 3 if an atomic-to-composite replacement is available, it is preferably chosen. Hence, step 2 is implicitly selected.

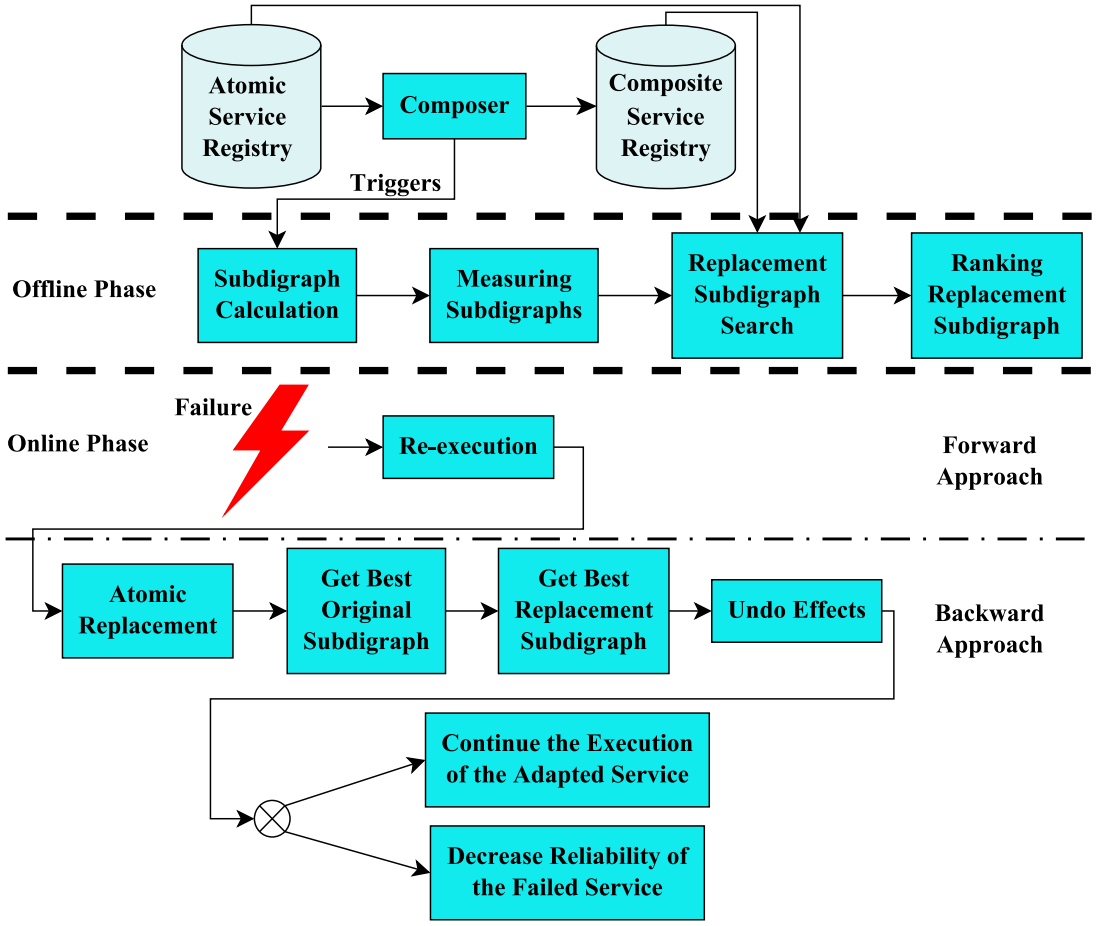


Figure 3.10.: A Detailed Depiction of the Proposed Failure Recovery Approach

3.3 Summary

We presented a failure recovery approach for composite semantic Web services based on a subdigraph renovation plan. The approach has two phases: A design-time phase and a run-time phase. The basic idea is that the design-time phase anticipates the possible fragmentations of a graph representing the behavior of a composite Web service, and the run-time phase, based on some events and gathered information, selects the fragments that should/could be replaced following a failure.

Figure 3.10 sketches the components of SRPFR, its various explained sub-tasks in two offline-online phases, and their flow in the proposed approach.

Chapter 4: A Test Collection of Composite Semantic Web Services

The main obstacle of working on composite Semantic Web Services is the lack of a standard Test Collection (TC), a data set for experiments on composite Web services.

Currently, there are more than 28600 (atomic) Web services on the surface Web, which was collected by a global Web Service Search Engine based on "focused crawler" called Seekda(Lausen & Haselwanter, 2007; Seekda, 2011) from 2007 to 2011. The Web services were offered by thousands of providers. This fairly big test collection of Web services include traditional Web services (Steinmetz et al., 2009).

However, the numbers for semantically described services are not satisfactory. Some research groups created test collections of semantic Web services (Küster & König-Ries, 2008). The test collections are meant to be used for a specific research (Bener et al., 2009) or to be published as a test suite for the research community.

Current available test collections of semantic Web services are SWS Test Collection (SWS-TC) (Ganjisaffar & Saboohi, 2006), OWL-S Service Retrieval Test Collection (OWLS-TC) (OWLS-TC, 2010), SAWSDL Service Retrieval Test Collection (SAWSDL-TC) (SAWSDL-TC, 2010), and WSMO-Lite Test Collection (WSMO-Lite-TC) (Cabral et al., 2012). The first two test collections are created using OWL-S (Martin et al., 2004), SAWSDL-TC is created for the SAWSDL (SAWSDL Working Group, 2007), and the latter is in WSMO-Lite (Fensel et al., 2010). Various characteristics of public test collections have been investigated (Küster & König-Ries, 2008). One major reported problem shared among the test collections is that none of them contain composite Web services, while mediation techniques need composite services for evaluation purposes.

In the literature, most of the researches on the Web services are usually evaluated on a small number of services (Chafle et al., 2006; Lin et al., 2010). There is an online portal for semantic services called OPOSSum (OPOSSum, 2009) which assembles data from SWS-TC, OWLS-TC and some other sources. It creates an assemblage of over 1500 semantic Web services using over 2800 descriptions for different description languages (Küster & König-Ries, 2008). Unfortunately, the portal is not updated any more and newly improved releases of the test collections are not imported into the portal.

There is an ongoing effort (Cabral et al., 2011) in making a new test collection of semantic Web services at the Semantic Evaluation At Large Scale (SEALS) Community (SEALS, 2011). The test collection is called WSMO-Lite-TC which is based on WSMO-Lite (Fensel et al., 2010). WSMO-Lite is the most recent submission to W3C as a semantic Web services' description language. However, according to the primary report, this new test collection does not provide composite services as well (Fensel et al., 2010).

All in all, to the best of our knowledge **there is no effort in making a test collection of composite semantic Web services**. Therefore, the noticeable gap in the field is that none of the investigated test collections include composite semantic Web services.

4.1 Representing Digraphs of Composite Services

As a means of representing the digraphs of composite semantic Web services computationally, i.e. representing which vertices are adjacent to which other vertices, there are some well-known and often-used ways. In order to represent a digraph $D = (V, A)$ for computational purposes an adjacency matrix, a collection of adjacency lists, and an incidence matrix are used.

According to Bang-Jensen & Gutin (2009):

"For the **adjacency matrix representation** of a directed multigraph $D = (V,A)$, we assume that the vertices of D are labelled $v_1, v_2, ..., v_n$ in some arbitrary but fixed manner. The adjacency matrix $M(D) = [m_{ij}]$ of a digraph D is an $n \times n$ matrix such that $m_{ij} = 1$ if $v_i \to v_j$ and $m_{ij} = 0$ otherwise. [...] The adjacency matrix representation is a very convenient and fast tool for checking whether there is an arc from a vertex to another one. [Page 696]

The **adjacency list representation** of a directed pseudograph $D = (V,A)$ consists of a pair of arrays Adj^+ and Adj^-. Each of Adj^+ and Adj^- consists of $|V|$ (linked) lists, one for every vertex in V. For each $x \in V$, the linked list $Adj^+(x)$ ($Adj^-(x)$, respectively) contains all vertices dominated by x (dominating x, respectively) in some fixed order (see Figure 4.1). Using the adjacency list $Adj^+(x)$ ($Adj^-(x)$) one can obtain all out-neighbours (in-neighbours) of a vertex x. [Page 697]

The vertex-arc **incidence matrix** $S = [s_{ij}]$ of a digraph $D = (V,A)$ has rows labelled by the vertices of V and columns labelled by the arcs of A and the entry s_{v_i,a_j} equals 1 if the arc a_j has tail v_i, -1 if a_j has head v_i and 0, otherwise. [Page 145]"

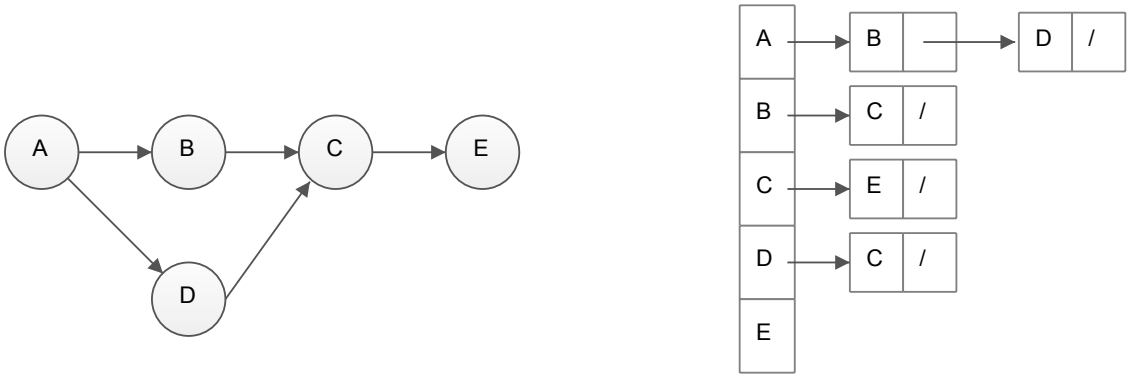
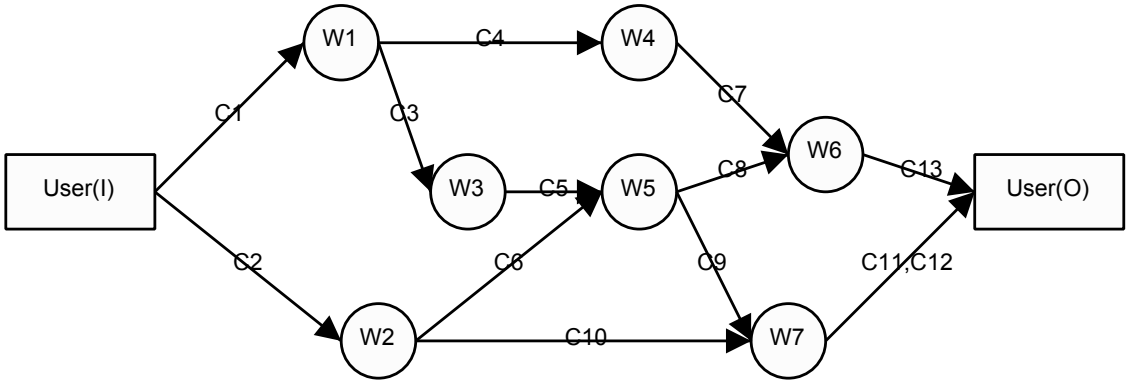


Figure 4.1.: A digraph and a representation by adjacency lists Adj^+



(a) A Digraph of 7 Services: "User(I)" is the *Source* and "User(O)" is the *Sink* of the Digraph

	User(I)	W1	W2	W3	W4	W5	W6	W7	User(O)
User(I)		C1	C2						
W1				C3	C4			C10	
W2						C6			
W3						C5			
W4							C7		
W5							C8	C9	
W6									C13
W7									C11 C12
User(O)									

(b) Adjacency Matrix

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
User(I)													
W1													
W2													
W3													
W4													
W5													
W6													
W7													
User(O)													

(c) Incidence Matrix

Figure 4.2.: A Digraph of Web Services with its Adjacency and Incidence Matrices

We use the adjacency matrix to represent our digraphs of composite semantic Web services (Figure 4.2). The coordinates of an adjacency matrix of our digraphs are $User(I)$ (digraphs' *source*), the constituent Web services' names, and $User(O)$ (*sink*). A non-diagonal entry is the concept used as the label for an edge between two Web services.

We neither use incidence matrix nor adjacency list:

- The incidence matrix (Figure 4.2c) is not appropriate because for a digraph of composite SWS, which typically includes many IO relations, the matrix will be huge with a high memory space consumption. Moreover, if there are more than one concepts for IO relation of two adjacent vertices, such as C_{11} and C_{12} in Figure 4.2c, either the matrix would not be a standard one with columns as edges (arcs) or there

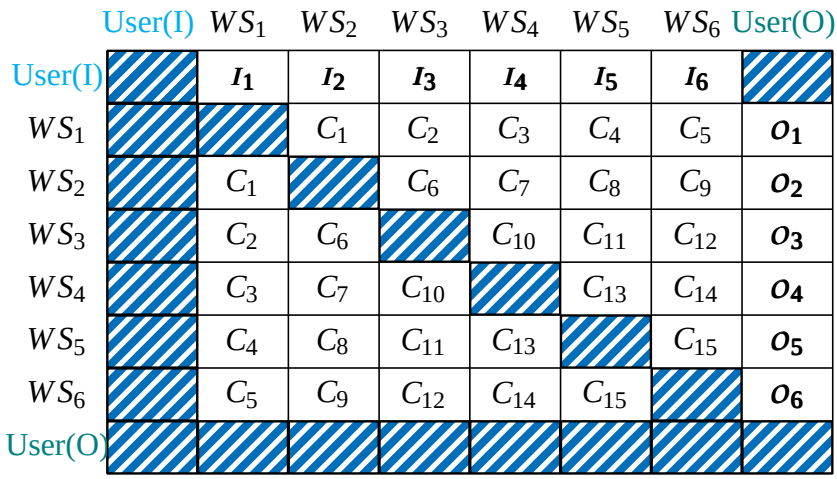


Figure 4.3.: Adjacency Matrix of a Valid Digraph of a Composite Web Service. Coordinates are User Inputs, Constituent Services ($WS_1..WS_6$), and User Outputs

would be complications for storing the concept values.

- The adjacency list needs more than constant time to verify the adjacency of two vertices (Bang-Jensen & Gutin, 2009). However, since mediation, which checks for the existence of an edge between two given vertices, must be done almost instantaneously, we chose to use the adjacency matrix rather than the adjacency list.

Figure 4.3 shows a template of an adjacency matrix for the inputs, and the outputs of a composite Web service with six constitutive Web services ($WS_1...WS_6$). Every cell shows edge(s) from a vertex shown at the left coordinate to a vertex shown at the top coordinate. In the figure the following rules are illustrated:

- The unhatched non-empty cells of the first row are the inputs of the composite service:

$$I = \{I_1, I_2, \ldots, I_n\}$$

- The unhatched non-empty cells of the last column are the outputs of the composite service:

$$O = \{O_1, O_2, \ldots, O_m\}$$

- Cells with the same values are mutually exclusive, i.e. only one of the two can have a value. The mutual exclusivity is because the graph is directed and between every two distinctive edges we allow edge(s) in one direction. The adjacency matrix is asymmetric.

  The numbers and their order do not convey any meaning and they are only given to clarify the figure.

- Hatched cells are the cells that are not allowed to have any value.

  – Diagonal cells are hatched because there is no loop in the graphs.

  – First column is hatched because there is no edge from other vertices to $User(I)$, i.e. $User(I)$ vertex is a *source*.

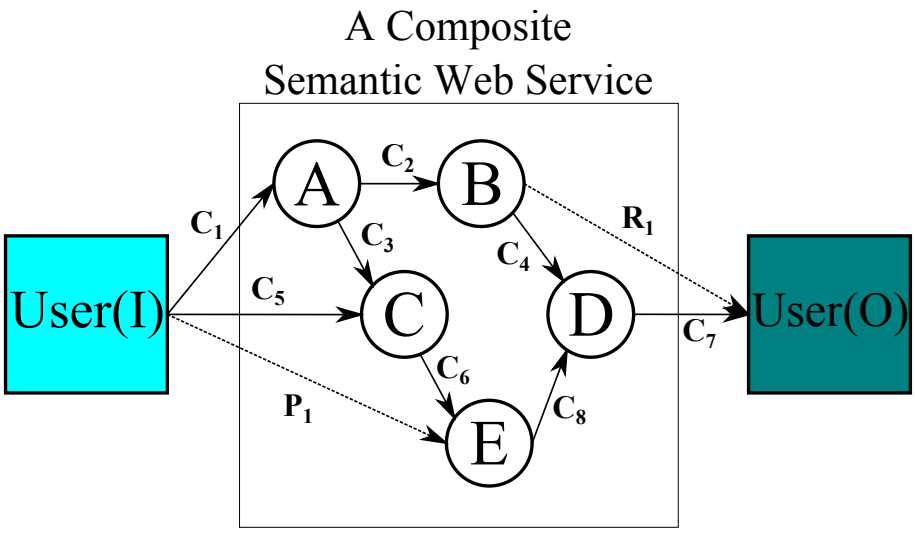
  – Similarly, the last row is hatched because there is no edge from $User(O)$ to other vertices, i.e. $User(O)$ vertex is a *sink*.

  – There is no direct edge from $User(I)$ to $User(O)$ and so the last cell of the first row is hatched.

Example 2. *An example of a composite semantic Web service with five constituent Web services is represented in Figure 4.4 with its IO adjacency matrix:*

- $\mathcal{W} = \{A, B, C, D, E\}$
- $\mathcal{E} = \{C_1, C_2, ..., C_8, P_1, R_1\}$
- $\mathcal{I} = \{C_1, C_5\}$, $\mathcal{O} = \{C_7\}$
- $\mathcal{P} = \{P_1\}$, $\mathcal{R} = \{R_1\}$

In the example, B is a ***world-altering*** *service (A Web service which has an effect on the world (McIlraith et al., 2001)), and A, C, D, and E are* ***information-providing*** *services.*



(a) A Sample Composite Service with 5 services

	User(I)	A	B	C	D	E	User(O)
User(I)		C_1		C_5			
A			C_2	C_3			
B					C_4		
C						C_6	
D							C_7
E					C_8		
User(O)							

(b) Adjacency Matrix of Inputs and Outputs

Figure 4.4.: A Sample Composite Semantic Web Service with Five Constituent Web Services together with its Adjacency Matrix of Inputs and Outputs

4.2 Subdigraph Calculation

We formally defined a subdigraph of a digraph of a composite service in Definition 6 (on page 45). We calculate subdigraphs of a composite semantic Web service by a sequence of vertex- and edge-deletion, shown in Algorithm 3. The algorithm receives a digraph as input whose subdigraphs are to be calculated and returns a list of subdigraphs which represent new semantic Web services.

By deleting a vertex in a digraph of services, there are two strategies to calculate the subdigraphs, i.e. to generate new composite semantic Web services.

Algorithm 3. *GenerateCompositeSWS*($\mathcal{G}$)

Input: *Digraph $\mathcal{G}$ is the digraph whose subdigraphs are to be calculated.*
Output: *Digraph sequence $\mathcal{R}$.*

```
1:  R ← (G); {An ordered list of digraphs to be traversed}
2:  for all G' in R do
3:    if |G'| > 1 then {G' is not an atomic service}
4:      for all v in G' do {v is a vertex in Digraph G'}
5:        H ← G' − v; {vertex-deletion}

6:        {Stage 1}
7:        Convert input edges of v to outputs of H;
8:        Convert output edges of v to inputs of H;
9:        R ← R ∪ H;

10:       {Stage 2}
11:       H' ← H; {1}
12:       Traverse forwards from v in H' and eliminate all vertices;
13:       Convert input edges of v to outputs of H';
14:       R ← R ∪ H';
15:       H'' ← H; {2}
16:       Traverse backwards from v in H'' and eliminate all vertices;
17:       Convert output edges of v to inputs of H'';
18:       R ← R ∪ H'';
19:     end for
20:   end if
21: end for
22: return R
```

First, considering the input(s) of the deleted service as the output(s) of the new composite service, and the output(s) of the deleted service as the input(s) of the new composite service, we achieve a new composite semantic Web service (Lines 7-9 of Algorithm 3). We do not change any other vertices in the digraph.

Second, taking the remaining vertices and following two paths (Stage 2 of Algorithm 3):

1. Deleting the succeeding services of the deleted service and considering the input(s) of the deleted service as the output(s) of the new composite service, we achieve a new semantic service (Lines 11-14).
2. Eliminating the preceding services and considering the output(s) of the deleted service as the input(s) of the new composite service (Lines 15-18).

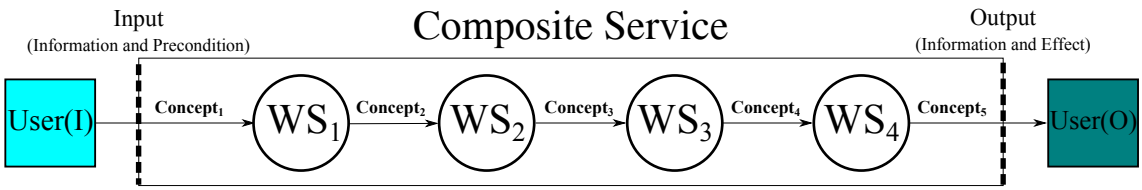


Figure 4.5.: A Sample Sequential Composite Semantic Web Service

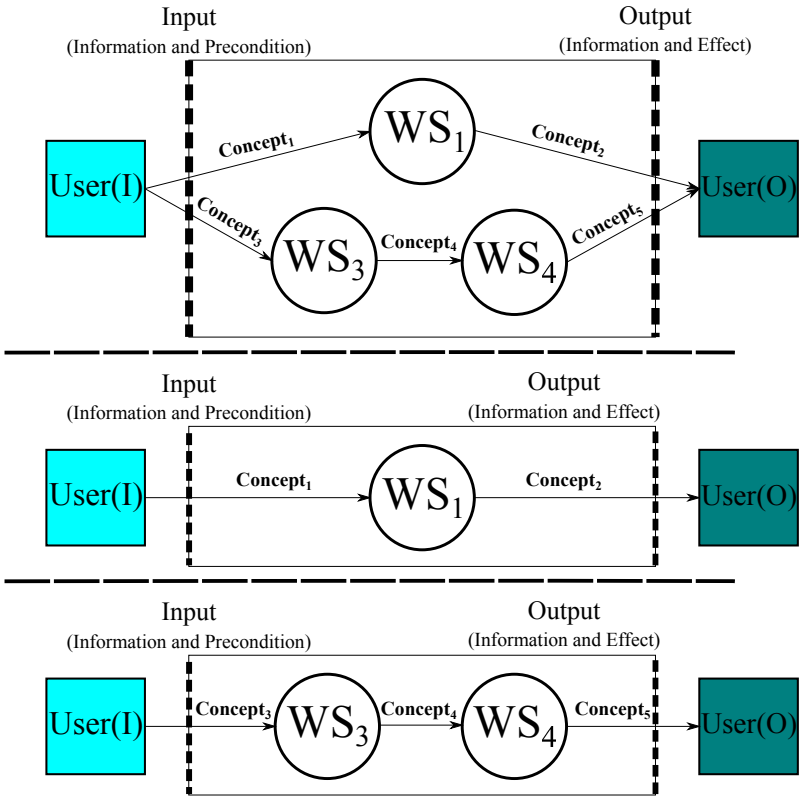


Figure 4.6.: Some of the Generated Semantic Services in Detail

Actually, iterating stages 1, and 2 of the algorithm yields some similar subdigraphs; however, taking into account both strategies for every digraph speeds up the calculations.

Figures 4.6, and 4.7 illustrate the subdigraphs of the sequential composite service which is shown in Figure 4.5. Figure 4.6 elaborates three of the subdigraphs and Figure 4.7 briefly depicts all possible subdigraphs based on our definitions, which are 15 subdigraphs, i.e. 15 semantic Web services.

Through the stages of the algorithm we make some new services and we add them to an ordered list. Then, we process the newly added digraphs to generate new subdigraphs of

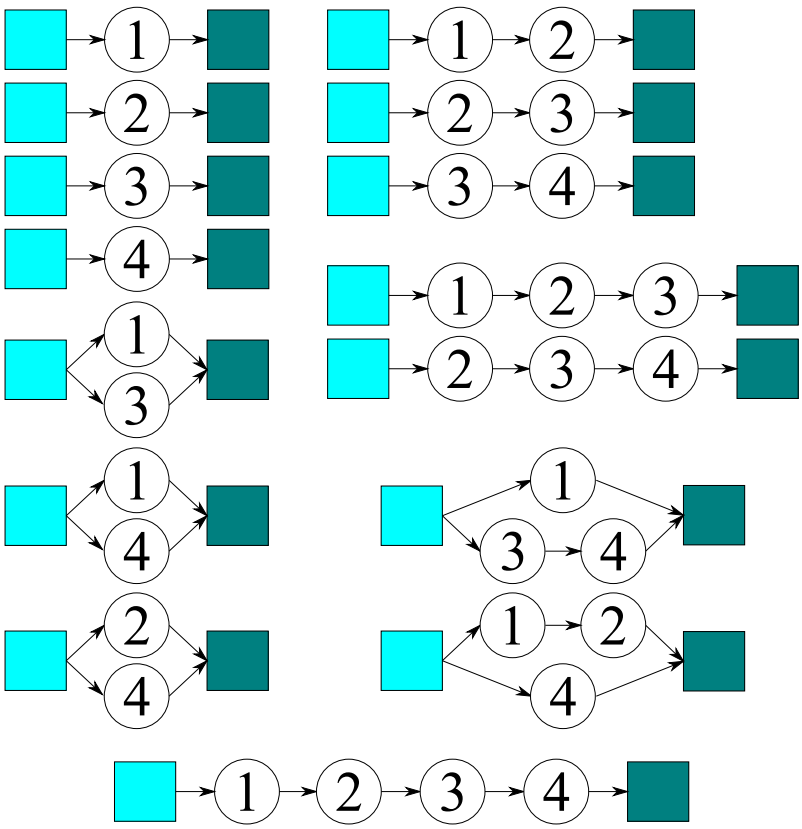


Figure 4.7.: All Possible Subdigraphs

services so as to make more digraphs of composite services. We continue processing the ordered list until no more digraphs are added. The stopping criteria is when we reach the atomic services of the digraphs.

4.2.1 Validation

The connectivity of the resulted subdigraphs is important. We check the connectivity of the underlying undirected graphs by checking the link from $User(I)$ vertex to all other internal vertices, and from all vertices to $User(O)$ vertex.

Furthermore, the validity of the digraphs are investigated in terms of the following conditions:

- There should be no direct connection between $User(I)$ and $User(O)$.
- $\deg^-(User(I)) = 0$, i.e. $User(I)$ is a *source.*
- $\deg^+(User(O)) = 0$, i.e. $User(O)$ is a *sink.*

- Every node should have at least one output (or a result).
- Every node should have at least one input.
- No (zero length) loop in the digraphs.

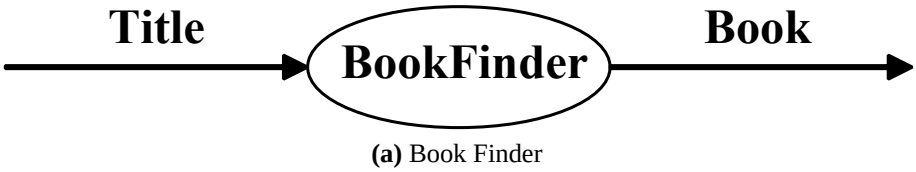
We deliberately keep the semantic Web services with similar input and output concepts. For instance, a neighborhood finding Web service, which returns the nearest city to another one, has the same input and output concept.

4.2.2 Implementation of the Test Collection Generation Algorithm

OWLS-TC is a well-known and often-used test collection of **atomic** semantic Web services (OWLS-TC, 2010). The test collection has been used in S3 (Semantic Service Selection) contests, an annual international contest on semantic service selection with an aim to evaluate retrieval performance of matchmakers for semantic web services. OWLS-TC is a service retrieval test collection in which the atomic services are described in OWL-S language. OWLS-TC Version 4 contains 1083 atomic semantic Web services which are described using their Input and Output, and Precondition and Results (IOPRs). We extracted the properties of atomic Web services from OWLS-TC. The extraction was performed using OWL-S API (cf. Appendix A, Section A.1). The properties include the service names (from human-readable information in ServiceProfile) along with their functional properties (Martin et al., 2004). Moreover, the concepts used in the services are extracted from their related ontologies available in the collection.

A detailed description of OWLS-TC is provided in Appendix A.

Figure 4.8 shows two sample atomic services of OWLS-TC. **BookFinder** service, depicted in Figure 4.8a, receives book's *Title* concept as input and generates the *Book* concept as output, i.e. it finds a book having its title. The service shown in Figure 4.8b, **BookRecommendedPrice**, receives *Book* concept as input and generates *Book*'s recom-



(a) Book Finder

(b) Book Recommended Price

Figure 4.8.: Sample Atomic Services

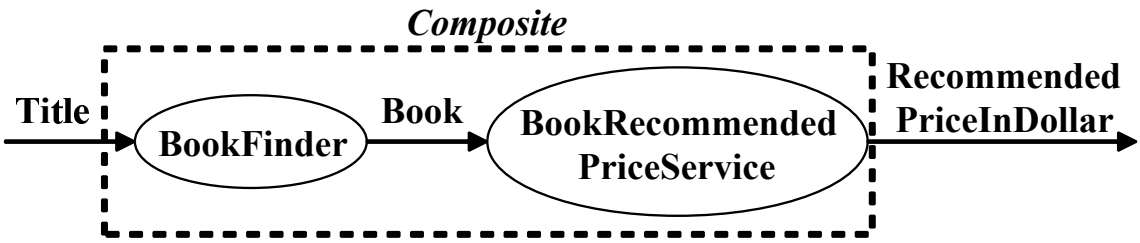


Figure 4.9.: A Sample Composite Service ($order = 2$)

mended *price* as output. The aforementioned concepts are predefined in ontologies available in OWLS-TC. Figure 4.9 depicts a composite service which is created by joining two atomic services in Figure 4.8.

For the purpose of evaluations, a number of composite semantic Web services (a sample is shown in 4.9) were created by IOPR matching, i.e. the IO of the (adjacent) services and their conditions (Precondition and Result (PR)) if available. The composite service generation is motivated by the approach presented for traditional Web services in (Bai et al., 2005). The structure of the composites were mostly sequential digraphs together with other various possible structures such as tree-like digraphs.

Then, all possible subdigraphs of the available composite services were calculated (using Algorithm 3) and non-duplicate composites were stored as other composite services.

The results show that having even a small number of valid composite services, we are able to create several other composites. For example, having one hundred initial composite services with a maximum order of 8, which we created using 1083 atomic services of OWLS-TC (Table A.3 provided in Appendix A), we generated over 2300 new composite digraphs.

The structures can be exported to the needed semantic Web services description languages and their adjacency matrices are exported as well.

4.3 Related Work

In the literature we could not find a similar work on creating a data set of composite semantic Web services to compare with. The works such as by (Chafle et al., 2006; Lin et al., 2010) synthetically made a data set as well for their evaluation purposes (cf. Table 2.3). However, for the purpose of comparison we could not use their data sets because the details were not reported.

A related work for test case generation of traditional Web services based on WSDL is reported in (Bai et al., 2005). It generates test cases of Web services based on service specifications in WSDL. The approach is capable of generating test cases based on the operations defined in WSDL 1.1 (Christensen et al., 2001). The Web services' operations are joined if an output message of an operation equals to an input of its precedent operation. Their experiments show its successful creation of test cases. However, the approach is based on syntactical data types defined in WSDL which do not convey any semantics. Our proposed method of composite Web service (test collection) generation is based on the Input and Output (IO) specifications of semantic Web services (defined in related ontologies) as well as the services' conditions, Precondition and Result (PR).

4.4 Discussion

The number of subdigraphs of the digraphs representing the composite semantic Web services based on our definitions are equal to the cardinality of the power set $\mathcal{P}(\mathcal{S})$ of a set of services (Equation 4.1). This is because we are considering every possible subset of a set of services.

$$|\mathcal{P}(S)| = 2^n \tag{4.1}$$

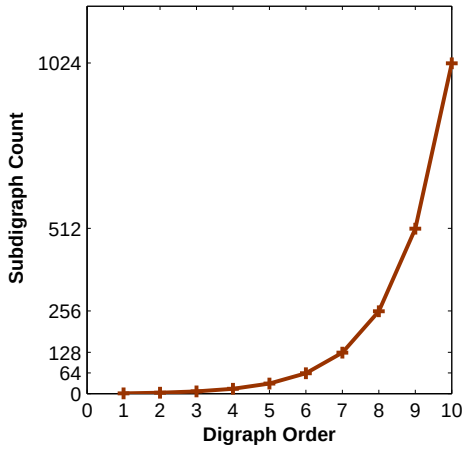
In our digraphs of services we do not admit null graphs, i.e. a composite service with no constituent service. Hence, the number of possible subdigraphs of a digraph of services is one fewer than the usual cardinality of a power set (Equation 4.1). So, for every digraph of services we generate a list of some other atomic and composite services. The length of this list for a digraph with the order of n is formulated in Equation 4.2.

$$SubdigraphCount = 2^n - 1, \quad n \geq 1 \tag{4.2}$$

The actual number of generated composite services from a digraph is fewer than the formula in Equation 4.2 because for every digraph of n services, there are n subdigraphs with the order of 1 which represent the atomic Web services that form the components of the composite services. Furthermore, the output list of Algorithm 3 contains its input digraph and so it is not considered as a newly generated Web service. Therefore, the actual number of generated subdigraphs are as in Equation 4.3.

$$SubdigraphCount_{Composite} = 2^n - n - 2, \quad n > 1 \tag{4.3}$$

Figure 4.10 illustrates the total number of subdigraphs ($2^n - 1$), and the number of subdigraphs which are other composite services ($2^n - n - 2$) for digraphs with the orders of one to ten ($1 \leq |\mathcal{G}| \leq 10$).



(a)

Digraph Order (n)	$SubdigraphCount$	$SubdigraphCount_{Composite}$
	$2^n - 1$	$2^n - n - 2$
1	1	-
2	3	-
3	7	3
4	15	10
5	31	25
6	63	56
7	127	119
8	255	246
9	511	501
10	1023	1012

(b)

Figure 4.10.: The Number of Subdigraphs for a Digraph $\mathcal{G}$ $(1 \leq |\mathcal{G}| \leq 10)$

Literally, by calculating the subdigraphs of m services with n constitutive services, we achieve $m * (2^n - n - 2)$ other composite services. However, because some of the subdigraphs might be duplications of other existing composite services, the final number of generated unique composite services are fewer than that.

As a conclusion, the noticeable lack of a test collection of composite semantic Web services hinders the researchers to evaluate their methods upon a valid collection of services. In this research, we proposed an adaptive approach of representing the digraphs of composite Web services along with its related subdigraph calculation algorithm. Utilizing the proposed algorithm on even a small set of composite semantic Web services generates an enormous set of composite semantic services.

Chapter 5: Recovery Probability

It is crucial that the user must get his request fulfilled even though failures are unavoidable. Therefore, even though a Web service might fail, the recovery method must have a reasonable probability of success so that it can complete the execution.

The objective of this chapter is to prove the applicability of the proposed method, SRPFR, in order to improve the percentage of successful recovery of the services in a registry.

For evaluation purposes, we implemented the proposed method and additionally in order to compare the results with other similar approaches we implemented an atomic replacement method as well as some well-known and highly cited works (Lin et al., 2010; Möller & Schuldt, 2010; Yu & Lin, 2005a). We chose these similar methods because their algorithms' description were clear enough to be implemented. Henceforth, the approach presented by Lin et al. (2010) is identified as "**Lin**" and "**Moller**" is used for the approach presented by Möller & Schuldt (2010). Moreover, "**CSPB**" is presented by Yu & Lin (2005a). We simulated the execution of the Web services and the failures that might happen during these executions in Java.

5.1 Experimental Setup

In order to create a test collection of services, we extracted the properties of atomic Web services of OWLS-TC. The OWLS-TC characteristics and the extraction are described in Appendix A, Section A.1 (on page 148).

5.1.1 OWLS-TC Matches

In terms of IO match, for 246 of the services (out of 1083) there is at least a match. This represents 23% of the services in the collection. Accordingly, in terms of IOPR match, a

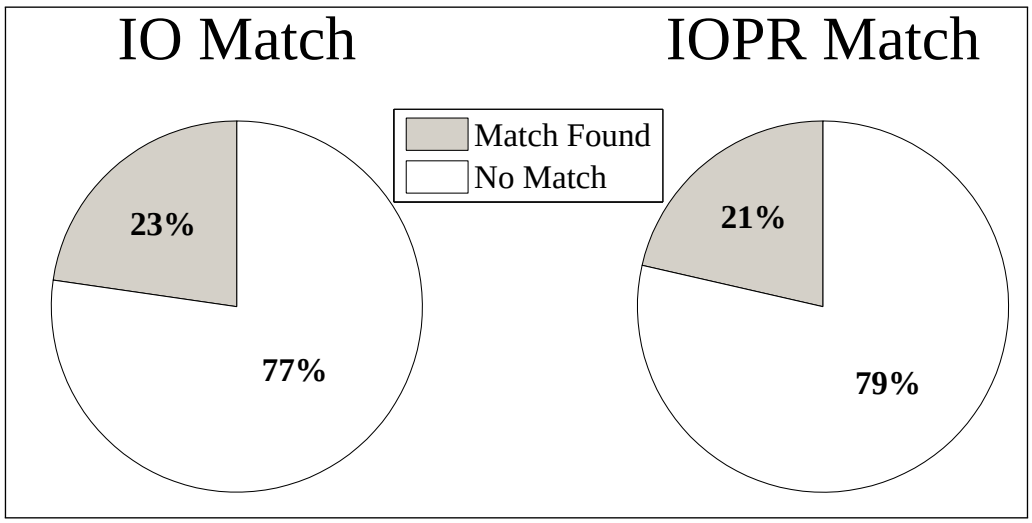


Figure 5.1.: Existence of an Exact Match in OWLS-TC

Table 5.1.: IO (IOR) Matches in OWLS-TC - Some of the Matches

Row	ID 1	Service Name 1	ID 2	Service Name 2
1	1	Kohl Car1PersonBicyclePrice	2	Car1PersonBicyclePrice
2	12	4WheeledCarBicyclePrice	139	Bicycle4Wheeledcar_Price_service
3	32	Academic Book Number booksearch	36	AcademicBookNumberSearch
4	35	AcademicBookNumberOrISBNSearch	780	BookFinderService
5	100	Auto Bicycle Price	140	BicycleAuto_Price_service

match is available for 232 of the services which comprises 21% of the test data. Figure 5.1 depicts the existence of an exact match among the services in OWLS-TC. Table 5.1 lists some of the aforementioned matches among the services in OWLS-TC. Appendix A elaborates various characteristics of the services described in OWLS-TC together with their IO and IOPR matching features. A complete list of matches among all the services in OWLS-TC is provided in Appendix A, Table A.4.

5.1.2 Discovery of Matches

In terms of the implementation of our method as well as other methods we needed a discovery approach. The discovery method is responsible to find a match for a digraph of Web services, i.e. an atomic Web service or a composite one. For example in our method,

the discovery is used to find “Replacement Subdigraphs” for the “Original Subdigraphs”. The task is explained in Section 3.2.1 (“*Finding Replacement Subdigraphs*” on page 54).

We used a search method based on first, the outputs and second, the inputs of the digraphs. For the purpose of making the search time an almost constant time we used a dedicated indexing technique based on IO, i.e. the digraphs of the test collection are indexed based on their outputs and inputs. Hence, in order to search for a match of a specific digraph, we looked for its outputs and inputs in a hash map.

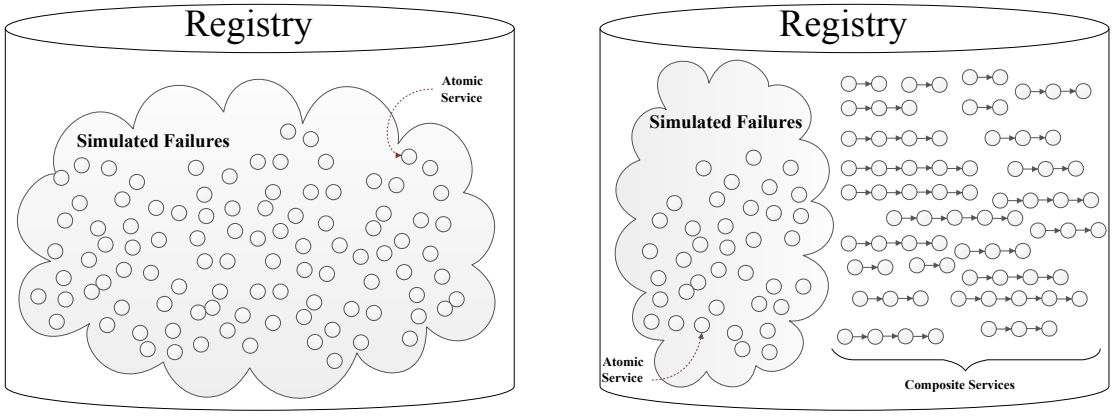
The above indexing and discovery method was used for all the implemented method in a similar manner. Therefore, a single search method isolates the methods from having different results of search, which ensures that the failure recovery probability values are from the method itself and definitely not from the discovery method used.

5.1.3 Experiments

We performed the execution simulation of our method together with other similar solutions both for atomic and composite services. The following sections report the results.

5.2 Experiments on Atomic Services

We tested the approaches on the atomic services available in OWLS-TC. We simulated the execution and failure of all the 1083 atomic Web services. The tests were performed in two different phases. First, we performed the tests on the atomics alone, i.e. the registry includes only the atomic services of OWLS-TC (Figure 5.2a). Second, we added some composite services, but we tested the failure recovery of OWLS-TC atomic services and not the services in the composites (Figure 5.2b). The difference between these two phases is the availability of the composites in the registry, which increases the chances of discovering a composite service that might be an exact match of an atomic service.



(a) Atomic Services Alone

(b) Atomic Services with the Availability of Composites

Figure 5.2.: Experiments on Atomic Services

5.2.1 Atomic Services Alone

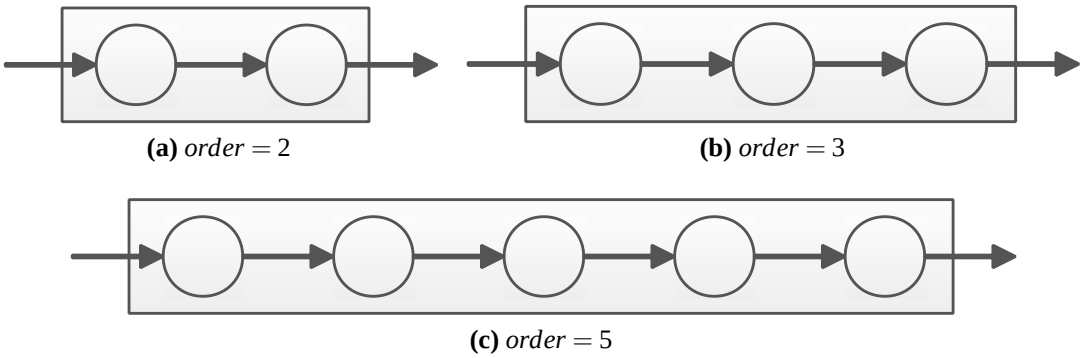
According to the characteristics of the services in OWLS-TC, there are 246 services for which at least an IO match is available. Therefore, if a failure happens for any of these 246 services, the approaches should be able to simply replace them with another similar atomic service. So, in $\frac{246}{1083} = 23\%$ of failures a recovery solution exists. In terms of IOPR matching the number is 232 out of 1083, which is 21% of failures (Figure 5.1).

The **research question** is: Are all the approaches capable of finding the matches for atomic services?

The results revealed that our proposed approach, atomic replacement approach, and all the implemented approaches, i.e. CSPB, Lin and Moller are capable of finding the matches, i.e. 23%, and 21% for IO and IOPR matches respectively, with the same probability (cf. Table 5.2, column “Only Atomics”).

5.2.2 Atomic Services with the Availability of Composites

Next, we created some composite services with the orders of 2 to 5. Figure 5.3 shows some sequential examples. We created more than 6000 composite structures for each



(a) $order = 2$ **(b)** $order = 3$ **(c)** $order = 5$

Figure 5.3.: Sequential Composite Services

digraph order of composite services (2 to 5). We chose a big number ($\geq$ 25000) to increase the chances of discovering matches between the two groups of atomic services and composite ones. Then, we re-executed our proposed approach, SRPFR, and the others.

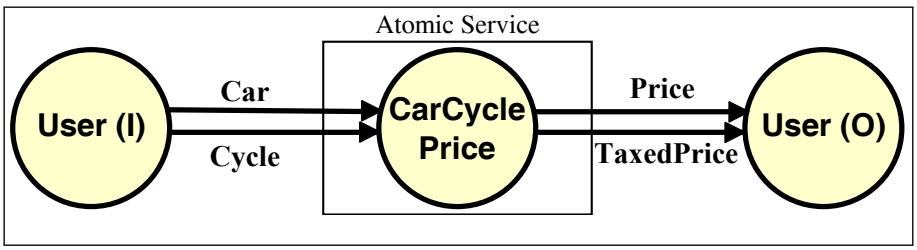
The **research question** here is: Can the availability of composite services beside the atomic services increase the *recovery probability* of atomic services?

Among all, SRPFR and two other methods, Lin and Moller, were able to increase the probability of recovering the services.

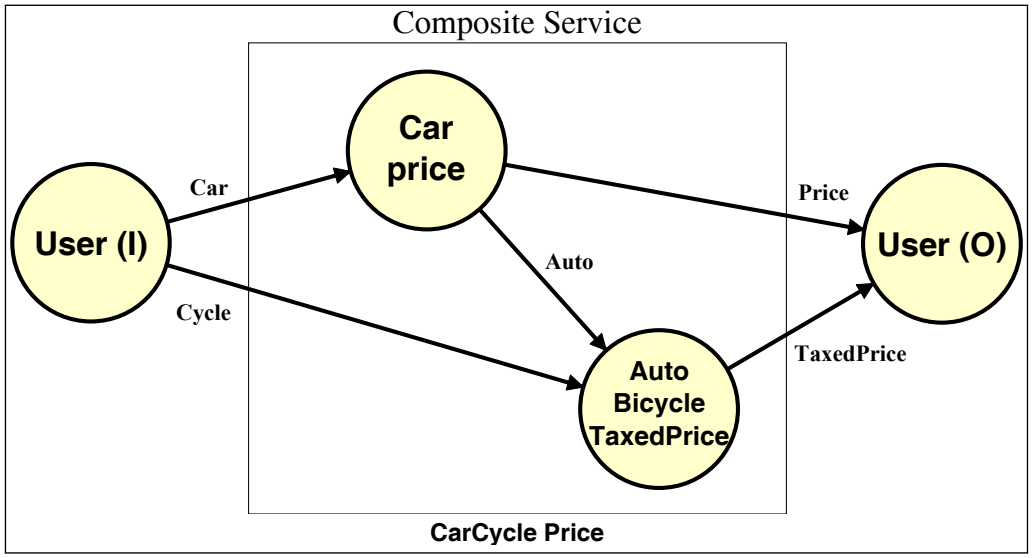
Examples

Two examples of the improvement are described as follows:

1. The service called “CarCyclePrice” (ID: 316) has no match in the test collection (cf. Table A.3, on page 157). Therefore, if it fails during execution, there is no (atomic) replacement available. However, having some composite services in the registry helps the discovery of such a service. For example, among the composite services that we created, three mentioned approaches were able to find a composite service with the order of 2 including two atomic service called “Car price” (ID: 205), and “Auto Bicycle TaxedPrice” (ID:106). Figure 5.4a shows our atomic example which is replaced by its IO (and IOPR) matched composite service (shown in Figure 5.4b).



(a) CarCyclePrice Atomic Service (from OWLS-TC)

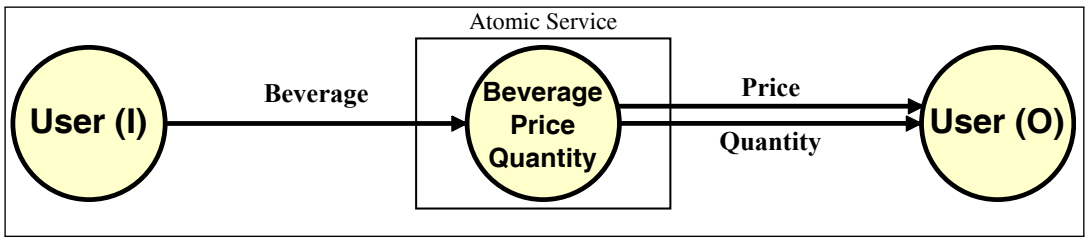


(b) CarCyclePrice Composite Service (Composed using two atomic services of OWLS-TC, "Car price" and "Auto Bicycle TaxedPrice"

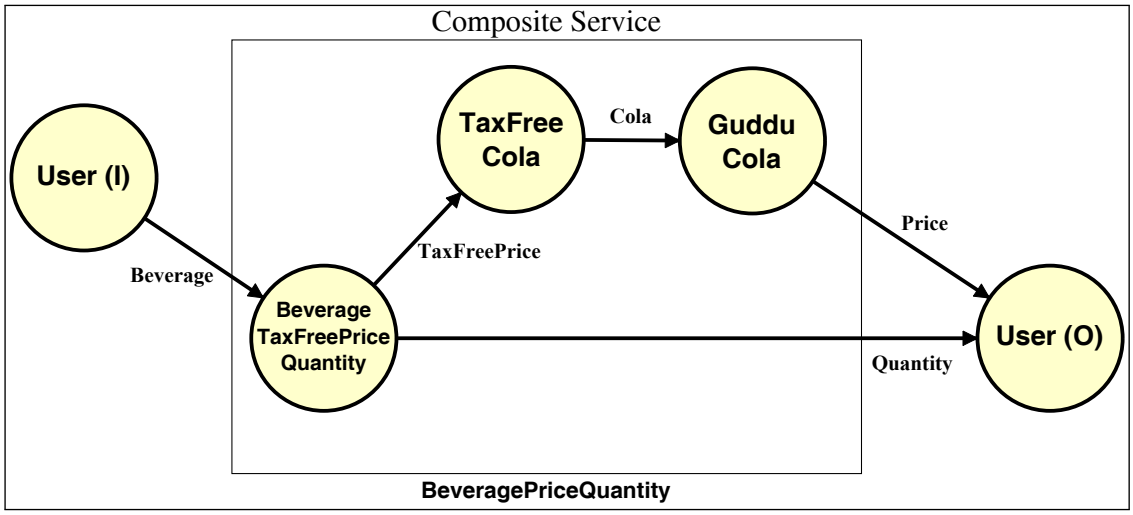
Figure 5.4.: First Example of an Atomic-to-Composite (1-n) Replacement (1-2)

The IO concepts in Figure 5.4, i.e. Car, Cycle, Price, TaxedPrice and Auto, are all pre-defined in multiple ontologies of OWLS-TC.

2. The service called "Beverage Price Quantity" (ID: 138) has no match in the test collection (cf. Table A.3, on page 154) as well. So, its atomic replacement is impossible. Accordingly, having the composite services in the registry enables the approaches to discover a matched service. For example, three mentioned approaches found a composite service with the order of 3 including three atomic services "Beverage TaxFreePrice Quantity" (ID: 138), "TaxFreeCola" (ID: 935), and "Guddu Cola" (ID: 758). Figure 5.5 shows the atomic service (Figure 5.5a) and its matched composite service (Figure 5.5b) based on IO (and IOPR) matching. Similarly, all the IO concepts in the figure are pre-defined in multiple ontologies of OWLS-TC.



(a) BeveragePriceQuantity Atomic Service (from OWLS-TC)



(b) BeveragePriceQuantity Composite Service (Composed using three atomic services of OWLS-TC, "Beverage TaxFreePrice Quantity", "TaxFreeCola", and "Guddu Cola"

Figure 5.5.: Second Example of an Atomic-to-Composite (1-n) Replacement (1-3)

Results

Figure 5.6 shows the changes in the percentage of recovery of the atomic services in OWLS-TC by the availability of some composite services. Additionally, Table 5.2 elaborates the results of various approaches.

Initially, having only atomic services (Digraph Order 1) in the collection (Figure 5.2a), all the approaches are able to find the existing matches with equal probability. However, adding higher digraph orders (composites) (Figure 5.2b) changes the percentage of the recovery of three approaches, Lin, Moller, and ours.

The gap widens by including the services with the higher digraph orders because the greater the number of composite services, the higher the chances of finding a match for services without an IO match in OWLS-TC (like "CarCyclePrice" service in Figure 5.4a).

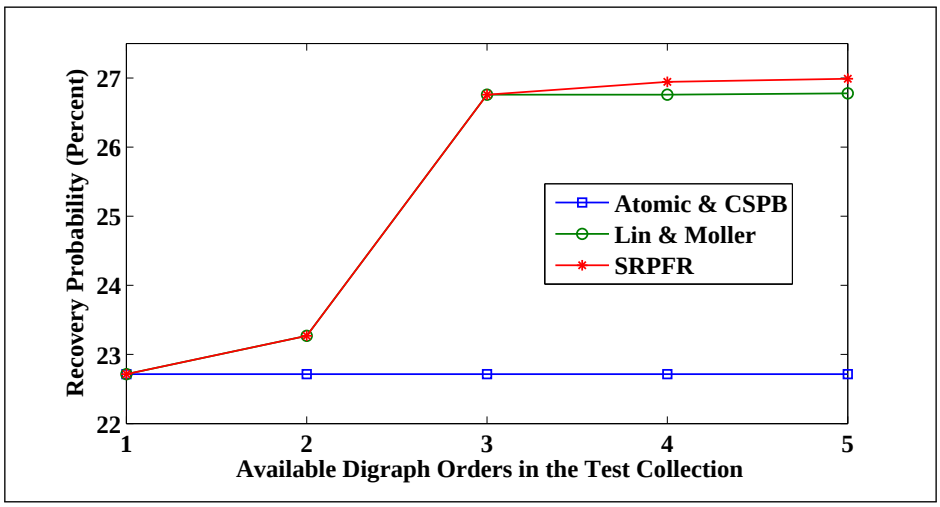


Figure 5.6.: Recovery Probability of Atomic Services with the Availability of Composites

Table 5.2.: Recovery Probability of Atomic Services with the Availability of Composites

Approach	Only Atomics	Composites			
		2	3	4	5
Atomic Replacement	22.71%	22.71%	22.71%	22.71%	22.71%
CSPB (Yu & Lin, 2005a)					
Lin (Lin et al., 2010)	22.71%	23.27%	26.76%	26.76%	26.78%
Moller (Möller & Schuldt, 2010)					
SRPFR	22.71%	23.27%	26.76%	26.94%	26.99%

However, increasing the digraph order (for example from 3 to 4) does not increase the gap any longer. The reason that the percentage does not significantly change is that when the number of components in a composite service increases, the number of Input and Output concepts increases as well. So, it is less probable that we can match an atomic service (of OWLS-TC) with such a composite one because as shown in Table A.1 (provided in Appendix A on page 150) the number of IOs for atomic services (in OWLS-TC) are usually not high.

Discussion

Finally, the difference between the success rate of SRPFR and the other methods, for higher digraph orders is insignificant. Our proposed approach shows a slightly better

percentage of successful recoveries, i.e. less than one percent. The reason behind the minor difference is as follows:

SRPFR calculates the subdigraphs of the composite services in the test collection (cf. Section 4.2) and adds them as other available composite services to its registry. The calculation increases the size of the internal test collection; hence, the opportunity of a successful discovery improves. In the above experiment, by adding the composite services of order 4, the subdigraph calculator first extracts the subdigraphs of the newly added composites, and second adds the smaller composite services (digraph orders between 2 and 3) to its internal registry (if they have not been added before). Therefore, it is more probable that in case of a failure, an atomic-to-composite replacement could be found. However, even for the higher digraph orders, the difference is not noticeable (less than 1% according to the last column of Table 5.2) because in the process of creating the test collection we have already created a big number of composite services with the smaller orders (2 and 3) and it is less likely to create new composite services. We claim that the fundamental difference between our proposed approach and the others slowly emerges from this experiment.

The column labeled “Only Atomics” in Table 5.2 shows the equality of the approaches in finding atomic replacements (one-to-one) among the services (studied in Section 2.3.2) in OWLS-TC (cf. Section 5.2.1). The four right columns with the label “Composites” and their digraph orders show the successful percentage of the approaches in finding an Atomic-to-Composite replacement (Section 2.4). The last row, representing the successful recovery percentage of SRPFR, shows the emerging improvement contributed by our proposed method for digraph orders 4 and 5 onwards (up to atomic-to-composite replacement stage) particularly in comparison with Atomic Replacement approaches and CSPB.

5.3 Experiments on Composite Services

In our experiments we generated the composites based on the atomic services in OWLS-TC. Further, we included the atomic Web services to the registry as well. The availability of the atomics in the registry enhances the chances of finding a match for discovery purposes, i.e. discovery of a potential atomic replacement, and an atomic replacement for a composite.

There are a few **assumptions** regarding our experimental setup for composite services tests as follows:

- For the simulation of the failures that may occur for the composite services, we assume that each and every component of a composite service may fail with equal probability. So, we do not consider any difference between the failure of different atomic Web services of a composite service.
- We deliberately kept only unique composite services in the registry, i.e. for every structure of a composite service including a finite number of atomic services there is at most one instance (and no duplicate) in the registry. This assumption ensures that we do not discover identical replacements which falsely shows recovery percentage improvement. In the next chapter, while we test QoS aspects of replacement we will remove this assumption.
- Even though we had the precondition and the result of the atomic services of OWLS-TC, we performed IO match rather than IOPR. In cases that we use both, we will mention the matching type. We used IO match because some of the approaches like CSPB are for traditional Web services and there is no semantics included, i.e. there is no way of differentiating information-providing and world-altering services. Therefore, using IOPR match for such methods makes the comparison unfair.

In order to evaluate the solutions on composite services we took two approaches:

1. We kept the structure of the composites in the test collection (such as their digraph order) constant and we increased their number. Therefore, we tested the probability of recovery of the approaches for different sizes of test collections with a single order of the graphs.
2. We combined the composites of various digraph orders. We included them in the registry. Hence, for each iteration of the experiments we had different sizes of test collections with a combination of different orders of graphs of composites.

For all the tests we repeated the creation of the test collections and testing the probability of successful executions 10 times, and we report the arithmetic *mean* value of ten iterations. The repetition ensures that the synthetic creation of the test collections does not affect the results.

5.3.1 Single-order Test Collections

We performed different tests on the composite services with the orders of 2, 3, and 5 separately.

The **research question** here is: How the size of the test collection with a single order of digraphs affects the successful *recovery* percentage of composite services?

The experiments and their results are described as follows.

Digraph Order 2

We performed the first experiment on composite services on a set of composites of order 2, i.e. there are two IO related atomic services in a composite structure. Figure 5.3a shows an overall view of such services. Moreover, Figure 5.4b (on page 83) and Figure 4.9 (on page 74) depict two real example composite services. It is clear, but worth mentioning,

that the composites' order are 2 before the adaptation. However, after the adaptation the graph order may change because an atomic service might be replaced by a composite, which results in having a composite graph of a different order.

We added the set of composite services to the registry (which includes atomic services) and we simulated their execution. For each execution we chose an atomic of the service that is being executed (a component of the execution of a composite service) and we declared it as a failed service. Then, we triggered all the failure recovery approaches and examined whether the approaches are capable of recovery. All successful recoveries were counted. Finally, we iterated the tests for several sizes of the test collection.

We repeated the experiments 20 times and each time we increased the number of composites by 200. Therefore, the first test was done on a registry containing 1083 atomic services (Table A.3) and 200 composites each with two atomics. Hence, the number of simulated failures were 400. Accordingly, the number of simulated failures for the last test was 8000 (Equations 5.1, and 5.2). The *Iteration* is 20 times, and the *Size* is 200.

$$Simulated\,Failures = Iteration * Digraph_{Order} * Size \tag{5.1}$$

$$Simulated\,Failures = 20 * 2 * 200 = 8000 \tag{5.2}$$

Equation 5.1 shows the number of simulated failures for a set of accumulated test collections of composite services with a single order.

Results:

Figure 5.7 depicts all the simulated failures with their percentage of successful recoveries for all the approaches.

The approaches' behaviors for the test collection of digraphs of composite services with order 2 are similar. Increasing the size of the test collections, increases the successful

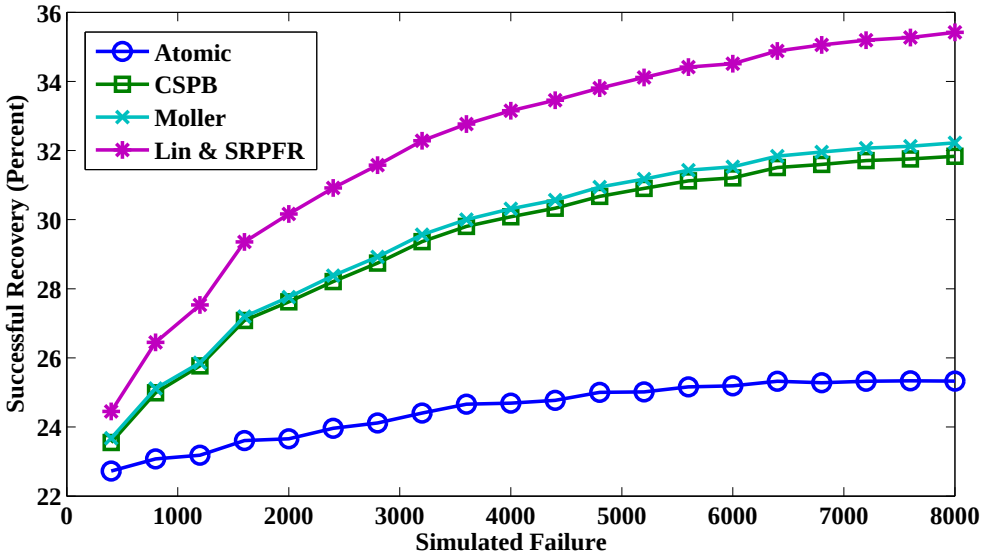


Figure 5.7.: Recovery Probability for Composite Services of $order = 2$ (a sample is shown in Figure 5.3a, and Figure 5.4b)

recovery percentage of the approaches. It is because when the data set is larger it is more probable that the approach is able to find a match.

Figure 5.7 shows that for a test collection with single order 2, SRPFR, and the approach presented by Lin et al. (2010) perform better and the percentage of successful recovery, i.e. probability of failure recovery, is higher for all sizes. Moreover, increasing the size of the test collections widens the gap between their performances. The growth of the gap shows that the two mentioned approaches perform better for larger test collections.

Digraph Order 3 and 5

We repeated the experiment on composite services of orders 3 and 5. The test collections were made similarly and the services which were composed included 3 and 5 IO related atomic services respectively (Figures 5.3b, 5.3c, and 5.5b).

The execution and failure simulations were carried out on a registry of atomic services and composites with either 3 or 5 constituents. The tests were replicated separately for registry sizes of 200 to 4000 composites increasing by 200. Hence, there were 600 to

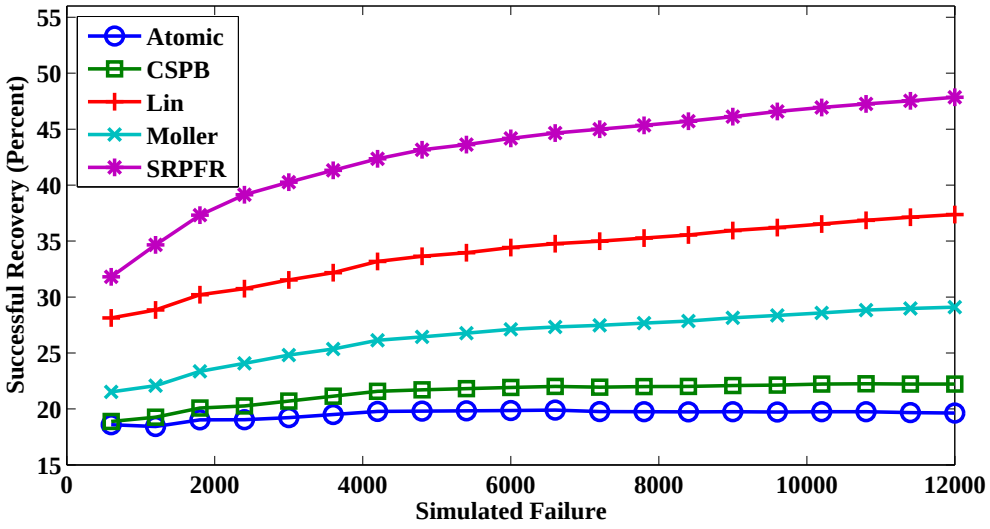


Figure 5.8.: Recovery Probability for Composite Services of *order* = 3 (a sample is shown in Figure 5.3b, and Figure 5.5b)

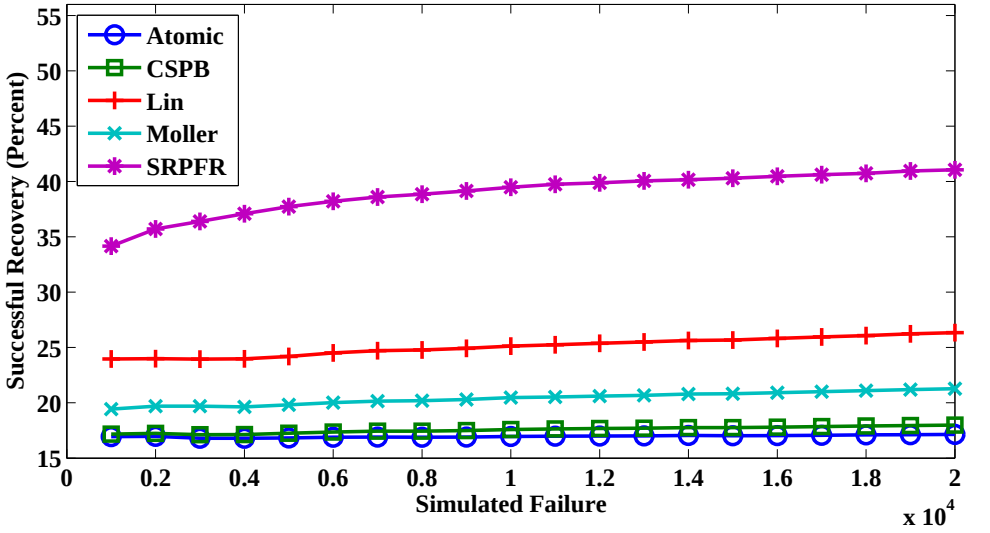


Figure 5.9.: Recovery Probability for Composite Services of *order* = 5 (a sample is shown in Figure 5.3c)

12000 simulated failures (Equation 5.3) for the test collections of the graph order 3.

$$Simulated\ Failures = 20 * 3 * 200 = 12000 \tag{5.3}$$

The number of simulated failures for the test collections of the graphs of order 5 were between 1000 and 20000 (Equation 5.4).

$$Simulated\ Failures = 20 * 5 * 200 = 20000 \tag{5.4}$$

Figures 5.8 and 5.9 show the recovery probability results for composite services of orders 3 and 5 respectively.

As shown in Figures 5.8 and 5.9, similar to the tests carried out for the graph of orders 2, all the approaches behave in a similar manner. Increasing the size of the registry, which allows discovery of more matches, the percentage of the successful recoveries is increased.

The gap between the successful recovery rates of SRPFR and other approaches remarkably grows from the first towards the end. The considerable difference for large number of composites is twofold:

1. SRPFR generates all possible subgraphs of the composites in the registry and hence the availability of more composite services increases the chances of finding a match for a subgraph of services.
2. SRPFR may go backwards through the structure of a composite to find a match.

These are the major reasons of the higher probability of successful recoveries of SRPFR.

Comparing Different Single-orders

In terms of single order test collections, we compared the successful recovery percentage of our proposed method, SRPFR, for different sizes of single order test collections.

The **research question** is: How does the digraph order of the test collection with a single order of digraphs affect the successful *recovery* percentage of composite services?

We designed the experiments, i.e. the number of iterations and the size of test collections such that we could have similar number of simulated failures. We created several test collections with the following sizes for three digraph of orders 2, 5, and 10:

- Digraph Order 2: Size: 200 to 6600:

$$Simulated\,Failures = 2 * 6600 = 13200$$

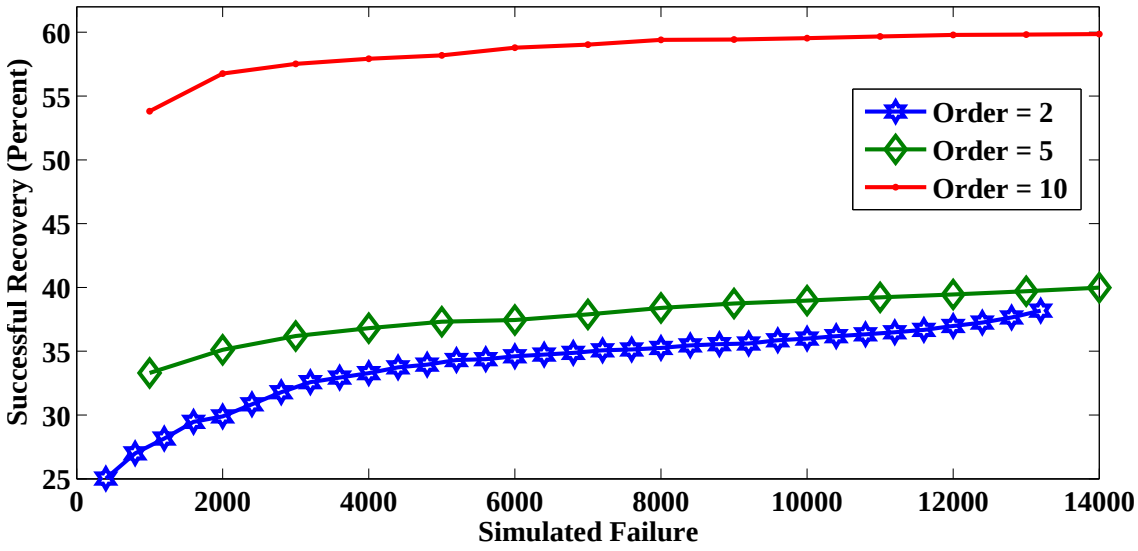


Figure 5.10.: Comparing Recovery Probability of SRPFR, orders 2, 5, and 10

- Digraph Order 5: Size: 200 to 2800:

$$Simulated\,Failures = 5 * 2800 = 14000$$

- Digraph Order 10: Size: 10 to 1400:

$$Simulated\,Failures = 10 * 1400 = 14000$$

Figure 5.10 shows that SRPFR's probability of recovery increases by higher orders of test collections because the higher sizes of the larger digraphs allow the method to create a bigger internal registry, which results in higher successful discoveries.

5.3.2 Multiple-order Test Collections

Likewise, we made test collections of composite service with a combination of different graph orders.

The **research question**: How does the size of the test collection with multiple orders of digraphs affect the successful *recovery* percentage of composite services?

For the experiments of the multiple order test collections we generated various orders of composites together with the atomic services.

The number of simulated failures (on the components on composite services) within a multiple order test collection is calculated using Equation 5.5. $Digraph_{Order}$ varied between two *minimum* and *maximum* values. The experiments were repeated for *Iteration* times and each time a set of services with a specific *Size* was added to the collection. The added set included an equal number of services from each $Digraph_{Order}$, i.e. we added $\frac{Size}{Max-Min+1}$ composite services for every $Digraph_{Order}$.

$$Simulated\,Failures = Iteration * \sum_{Digraph_{Order}=Min}^{Max} (Digraph_{Order} * \frac{Size}{Max-Min+1}) \quad (5.5)$$

The following sections describe the experiments and their results.

Test Collections: $2 \leq Digraph_{Order} \leq 6$

The first multiple-order test was carried out by assigning the following numbers to Equation 5.5:

- *Minimum* $Digraph_{Order}$ = 2
- *Maximum* $Digraph_{Order}$ = 6
- *Iteration* = 20
- *Size* = 200

So, the simulated failures were between 800 and 16000:

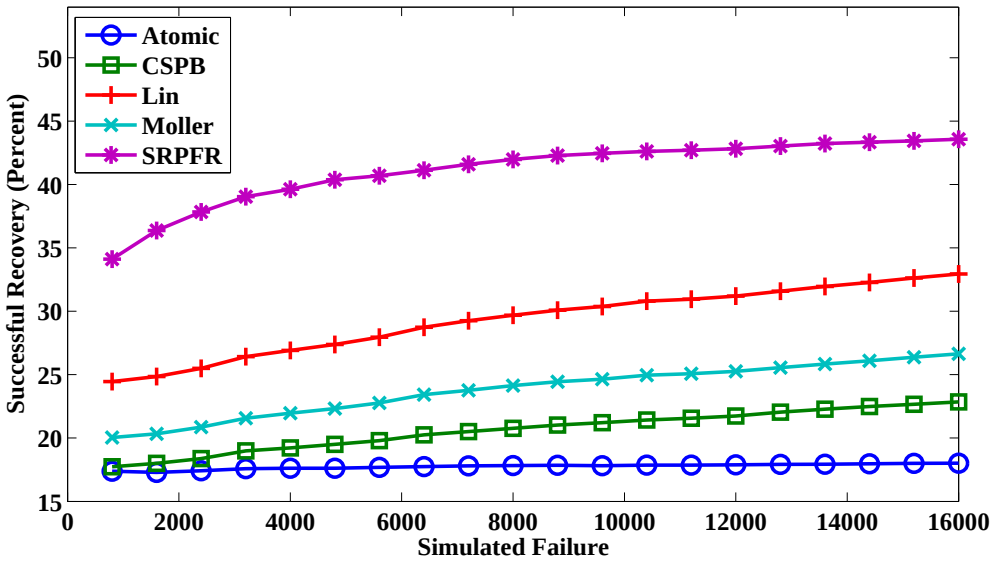
$$Simulated\,Failures = 20 * \sum_{Digraph_{Order}=2}^{6} (Digraph_{Order} * \frac{200}{5}) = 16000$$

Figure 5.11a shows the result for the composite services' test collection.

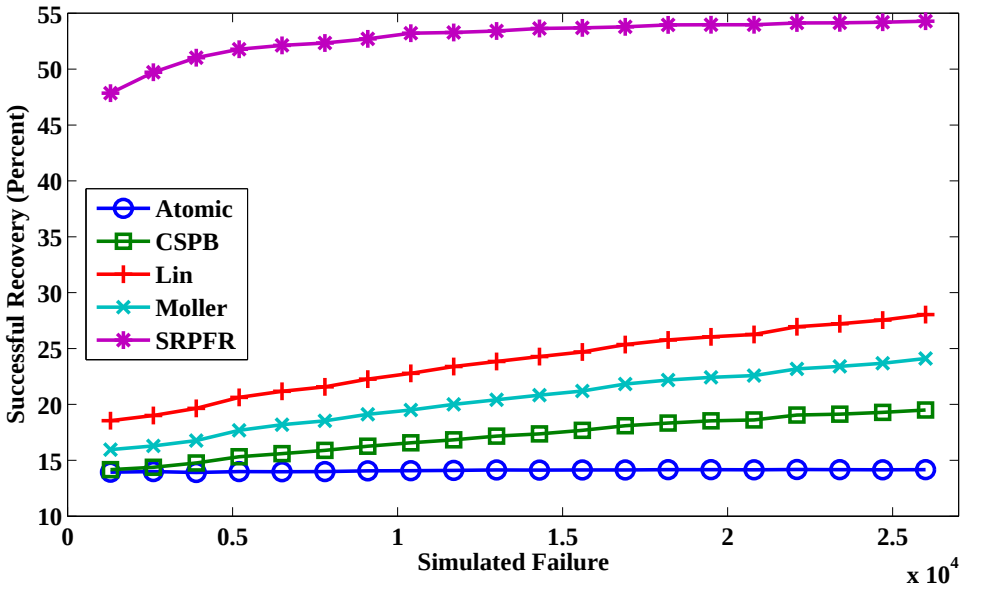
Test Collections: $2 \leq Digraph_{Order} \leq 11$

We assigned the numbers to the formula (Equation 5.5) for the second multiple order test as follows:

- *Minimum* $Digraph_{Order}$ = 2



(a) Composite services with a combination of orders between 2 and 6 ($2 \leq order \leq 6$)



(b) Composite services with a combination of orders between 2 and 11 ($2 \leq order \leq 11$)

Figure 5.11.: Recovery Probability for Composite Services of Multiple Order

- $Maximum\ Digraph_{Order}$ = 11
- $Iteration$ = 20
- $Size$ = 200

As a result, there were between 1300 and 26000 simulated failures:

$$Simulated\ Failures = 20 * \sum_{Digraph_{Order}=2}^{11} (Digraph_{Order} * \frac{200}{10}) = 26000$$

Accordingly, Figure 5.11b is for orders between two and eleven inclusive. The successful recovery percentage is significantly higher when the order of the digraphs increases.

The reason is that by having bigger digraphs of services, the internal test collection, i.e. the registry created by adding the subdigraphs of services, will have higher number of composites. Hence, the probability of discovering “Replacement Digraphs” is higher.

Results

The results reveal similar outcome for all the approaches: When the test collection’s size grows, the probability is increased and the gap between the probability of the approaches expands.

A Large Multiple-order Test Collection

For a final test, we created a large test collection with a combination of different orders of composites. The test collection’s size was 5000 (composite services) with the orders of two to eleven. The number of simulated failures were 32500:

$$Simulated\,Failures = \sum_{Digraph_{Order}=2}^{11} (Digraph_{Order} * \frac{5000}{10}) = 32500$$

Figure 5.12 illustrates the result. It is a noteworthy achievement in which our proposed approach has almost twice the recovery percentage of even the best of the other approaches.

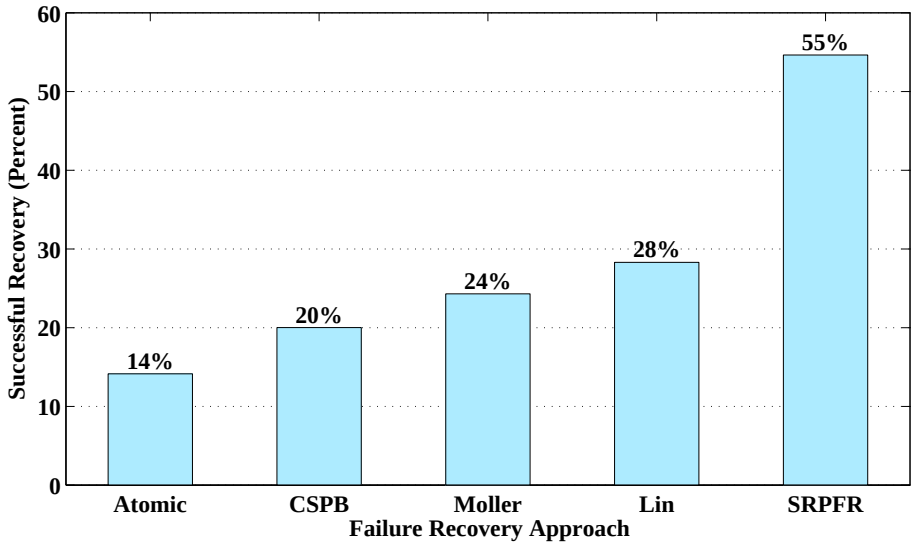


Figure 5.12.: Recovery Probability for a Large Test Collection of Composite Services, size = 5000, ($2 \leq order \leq 11$)

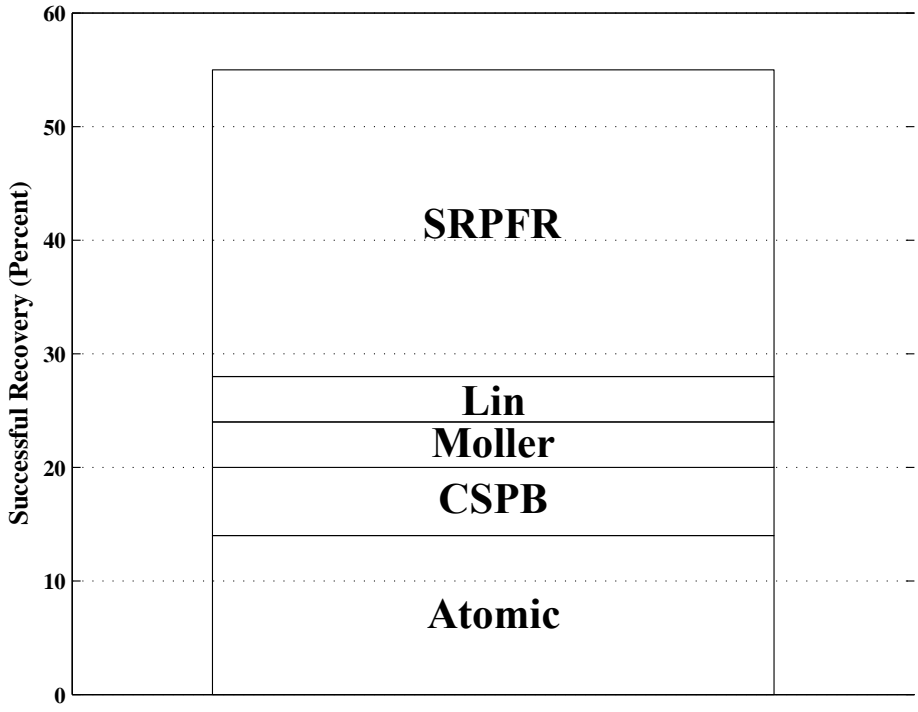


Figure 5.13.: The Successful Recovery Percentage Increase by the Approaches

5.4 Discussion and Conclusion

In this chapter we reported the results of multiple experiments. The experiments carried out to test the successful recovery percentage of the studied approaches on the Web processes including atomic and composite services.

Figure 5.13 shows a summary of the successful recovery percentage of the approaches for the large test collection (illustrated in Figure 5.12). The figure shows the improvement contributed by each method in terms of successful recovery percentage. It is obvious that our method increased the percentage by 27 percents. The methods are discussed as follows.

The methods are sorted in terms of percentage of successful recovery as shown in Figure 5.13 in the following:

1. The lowest probability is for an approach performing an **atomic replacement** on the constituting components. The reason is that the atomic replacement relies on the existence of a matching atomic service, which is constant and not high.
2. The approach presented by Yu & Lin (2005a), called CSPB has the second rank. It shows better success percentage than an atomic replacement approach because it looks for replacement paths containing more than one service ($Digraph_{Order} \geq 1$). So, the probability of discovering a matched service is higher. However, the method uses the abstract services, i.e. it switches the services based on their membership to the service classes. Therefore, every service in the potential digraph "to be replaced" can be replaced by other members of the service class. In other words, the order of the graph of the so-called "backup path" and its initial path is equal. This constraint does not allow the approach to find backup paths of different orders, so, the probability does not improve significantly.
3. The next rank belongs to the approach represented by Möller & Schuldt (2010). The success percentage is higher than two previous approaches because the method checks for many-to-many (n:m) replacements. So, the subgraph of the original composite service "to be replaced" and the new subgraph may have different orders, which allows the discovery of other matching services. However, the "to be replaced" subgraph merely includes the services which has not been executed yet. In other words, to find a subgraph of service to be removed it does not go backwards to the well-executed services.
4. The runner-up, which is the most related approach and closest one to ours, is the method described by Lin et al. (2010). The method has the highest recovery probability than the others because it replaces a region of the digraph including even the well-executed services. Moreover, the replacement subdigraph may include a

different number of services as compared to the region “to be replaced”.

5. The **highest probability** of failure recovery belongs to our proposed method, **SRPFR**. The proposed method has all the advantages of the studied approaches, i.e. it looks for an atomic replacement, the “Original Subdigraph” and “Replacement Subdigraph” can have different orders, and it can go backwards through the well-executed services considering information-providing and world-altering services. Moreover, due to pre-calculations which is designed to be done offline and in an ongoing manner, SRPFR finds more matching replacements. Therefore, having higher number of replacements allows the method 1) to be successful in recovery of the failed service in more cases, hence higher percentage of successful failure recovery, and 2) to be able to choose a better replacement in terms of digraphs’ order as well as QoS attributes.

Consequently, the experiment results reveal that SRPFR approach performs better in terms of the probability of recovery for failures that may occur for composite Web services. Further, having larger test collections enables SRPFR to find more matches and so the probability is higher.

Yu & Lin (2005a) claimed that if for a particular atomic Web service there is no backup path, it is a **critical point** for the business process. Comparing the results of the the solutions we find that if the mediator goes backwards through the structure even to the well-executed services and discovers the matches, it is more probable to find a replacement. Hence, the number of so-called *critical points* will be minimal.

Chapter 6: Quality of Service (QoS)

A failure recovery method with high probability of success is indispensable for a reliable system. However, there are some other critical factors which are essential for the user. For example the whole system may face a serious danger if its response time is high. The part of the "response time" that is of our concern is the required time for the service-based system to recover after a failure. The other factors are the deviation of the system from its promising non-functional properties, such as the cost of the execution, service execution time, and the reliability of the chosen components.

Our major objective in this chapter is to show that the delay of the proposed recovery approach at execution time, and specifically at the moment of failure is minimum, i.e. since the calculations are done in the offline phase; the delay of recovery is almost zero. Moreover, we will prove that the deviation of Quality of Service (QoS) caused by a successful failure recovery (using SRPFR approach) is minimum. That is to say, among the choices for the replacement, the best one with similar QoS values is chosen.

In the following experiments we created the test collections similar to Chapter 5.

6.1 Failure Recovery Delay Time

A failure recovery process requires that a failed composite service has a high percentage of success. Moreover, the time needed to recover the system from a failure, i.e. the interval commencing from the failure onset until the moment that the system can continue to respond, must be minimized.

In order to measure the required time of recovery we separated the different time intervals of various sub-tasks of a recovery method. The intervals are in three major categories:

1. **Initialization Time**: This is the time needed for the approach to arrange its required internal structures. The initializer process is called after the system's registry of the services is made ready and before commencing any execution, i.e. system's startup time. For example, in our implementation we perform the indexing on the inputs and the outputs of the services at the initialization stage.
2. **Preparation Time**: Before commencing the execution of every composite Web service, the failure recovery method of the system needs to prepare for the service's failures. The preparation is triggered before starting the execution of the service. In our experiments, we search for the matches at this stage.
3. **Recovery Time**: The recovery time is the interval from the moment of failure to the moment that the system can continue the execution. In other words, it is the adaptation time of the composite service. The recovery time is further divided into the time for its two smaller sub processes:

    a) **Recovery Search Time**: The recovery approaches need time to discover a similar service. The similar service is a single service in Atomic Replacement approaches and a set of services in others.

    b) **Recovery Actual Replacement Time**: Finally, the recovery approach needs to replace the failed service with a single or multiple services. We name this time as *Actual Replacement time*.

During the execution of a composite Web service if any failure occurs, the recovery approach needs a particular duration of time, which we call "Recovery Time", to adapt its primary structure. In our proposed method we transferred the time-consuming calculations and processes to the initialization, and preparation sections (Offline phase, cf. Section 3.2.1 on page 50) aiming to minimize the actual delay time of replacements. There-

fore, at run-time (Online phase, cf. Section 3.2.2 on page 57) the approach recovers the failure without any processing delay.

6.1.1 Experiments on a Big Test Collection

For the purpose of simulating the delay time of the SRPFR, we created a test collection of composite services with a combination of different digraph orders, which is similar to a practical test collection. Next, we simulated the execution of the services and a failure of a component of the executing Web service. We measured the time interval of the SRPFR to perform the calculations and the replacement.

The experiments were performed on a machine with the following specifications:

- **CPU**: Intel Core 2 Duo, 3.0 GHz CPU
- **Memory**: 8GB
- **Operating System**: Windows Vista Business, Service Pack 2, 32-bit
- **Java**: Version 7

The **Research Question** is: How long does SRPFR take to adapt a failed composite service?

The test collection, which we created for this purpose, included 3000 composite services, together with the atomic services of the OWLS-TC (1083 services). The digraph orders were between 2 and 11 inclusive. We simulated the execution of the 3000 composite services and the failures of all their components separately. The time intervals of the sub-tasks of SRPFR were measured. The *mean* values of repeating the experiment 10 times are reported as follows:

SRPFR needed an interval of 3.7 seconds to initialize its internal structures. Then, we started the simulation of the execution and their failures. There were 19500 simulated

failures (for 3000 composite service tests). The average preparation time of each composite service was 35 milliseconds (*ms*). Most importantly, the recovery time for 19500 failures were approximately less than one microsecond. The Actual Replacement time for recovery was almost zero because the "Original Subdigraph" and its best "Replacement Subdigraph" were calculated during the preparation time and no time-consuming calculation was required at the failure moment.

In other words, for the above test collection, SRPFR approach adds an average of 35*ms* delay before commencing the execution of each composite service, and less than a millisecond at the failure time.

In order to precisely identify the preparation time of the composite services in this experiment, we measured their preparation time according to the composites' digraph order. Figure 6.1 shows the average preparation time of the composite services based on their digraph order.

It is noteworthy that the successful recovery probability of the SRPFR for this test collection was 55%, which is a significant achievement as discussed earlier in Chapter 5.

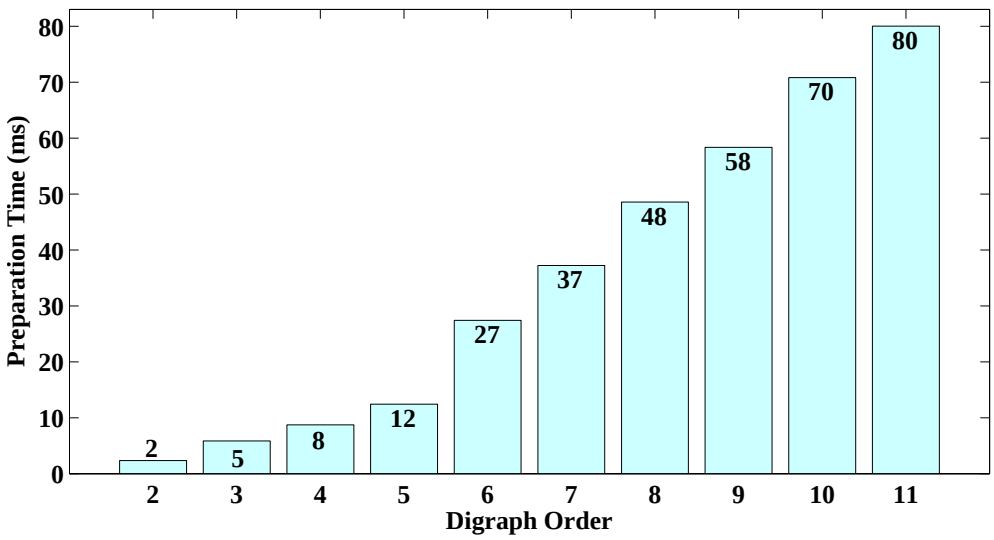


Figure 6.1.: Preparation Time (milliseconds) of Different Digraph Orders in a Test Collection of 3000 Services (Digraph Orders between 2 to 11)

6.1.2 Experiments on the Size Changes of the Test Collections

The **Research Question** is: How do the initialization time, the preparation time, and the recovery time change by including more composite services into the registry? Is the preparation time taken by SRPFR for composite services with different digraph orders the same? Is the recovery time a constant value for different sizes of the registry?

For evaluation purposes, we synthetically created a set of composite services with digraph orders of 2 to 11. The composite services were generated similar to our other experiments, which are based on our proposed method in Chapter 4. The initial set included 150 composite services (15 composite services for each digraph order). We simulated the execution of the services and the failure of their constituent components. We measured the time needed to complete each step of our proposed method, i.e. the initialization time, the preparation time to execute the composite service, and the recovery time needed at the failure onset. Then we generated another set of composite services. The new set included 150 unique composite services, which we added to the registry. We repeated the simulations to measure the various steps' interval time separately. The composite services' generation and their inclusion in the registry are iterated so that the registry reached a size of 3000 composite services.

The experiments were performed on the same machine as described in Section 6.1.1. The reported values are the arithmetic mean values of repeating the experiments 10 times to eliminate the effect of a synthetic creation of the test collections.

The initialization time required for every step of this experiments were linearly increased. The first registry (including only 150 composite services) needed around 104*ms* to be initialized (so that SRPFR could use it). The last registry with 3000 composite services

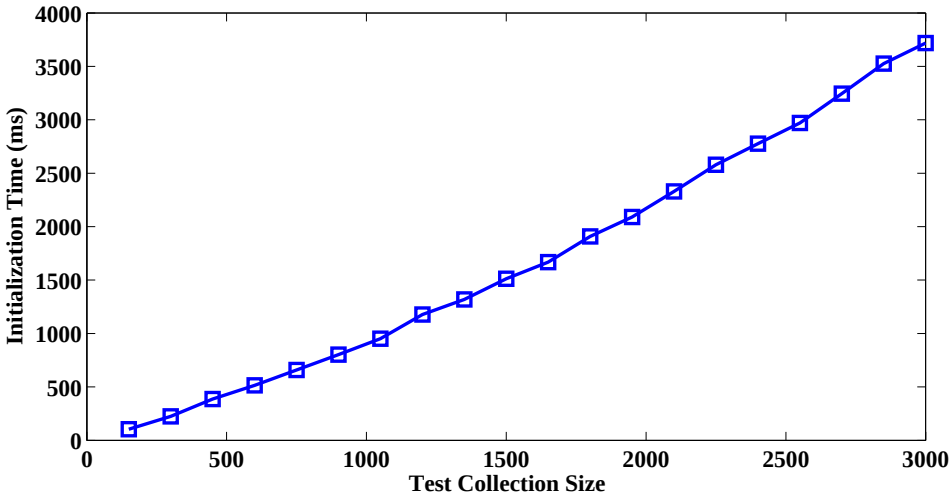


Figure 6.2.: Initialization Time (milliseconds) for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 11)

Table 6.1.: Initialization Times (milliseconds) for Test Collections of 150 to 3000 Composite Services including Digraph Orders between 2 to 11

Test Collection Size	150	300	450	600	750	900	1050	1200	1350	1500	1650	1800	1950	2100	2250	2400	2550	2700	2850	3000
Initialization Time (*ms*)	104	225	386	514	658	802	951	1177	1318	1512	1667	1908	2091	2329	2579	2777	2970	3245	3526	3718

needed around 3.7 seconds to be initialized. The initialization time for a registry of 3000 composite services is not too short; however, because this is a one time process, that is, at the system's startup time, and will not be repeated, its delay time will not cause the system to respond slower. Figure 6.2 shows the increment of the initialization time for a registry with a test collection size from 150 to 3000. Table 6.1 displays the initialization time (milliseconds) values for the experiment.

In order to measure the "preparation time" required for the execution of the composite services we performed the preparation for all the composite services in the registry in each iteration (adding 150 composite services to the registry). The time measurement has been done separately for the composite services of different graph orders. Hence, we could figure out the time interval for the preparation of composite services with 2 to

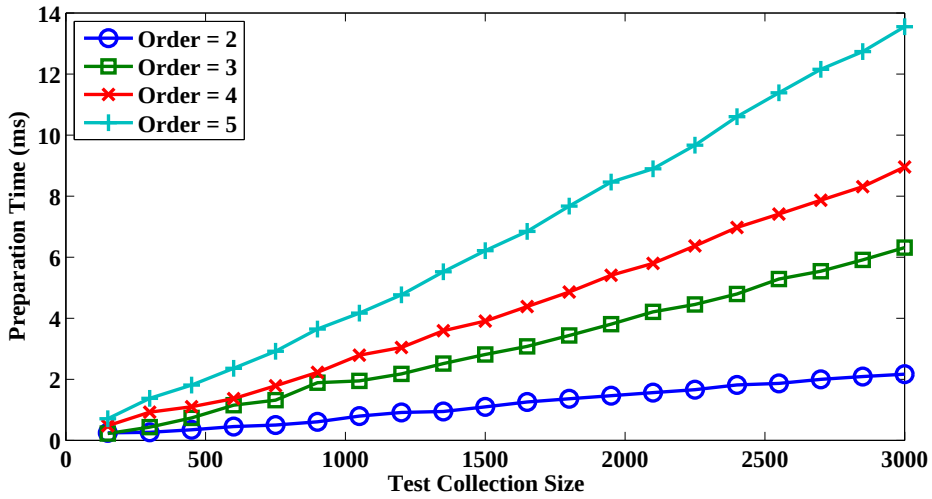


Figure 6.3.: Preparation Time (milliseconds) for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 5)

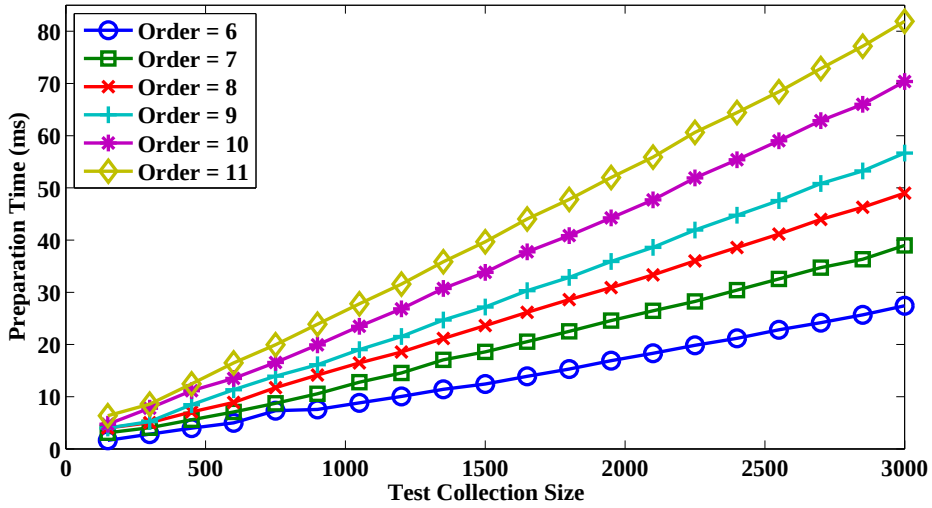


Figure 6.4.: Preparation Time (milliseconds) for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 6 to 11)

11 atomic services distinctively. Figures 6.3 and 6.4 depict the preparation time of the composite services with 2 to 5, and 6 to 11 components respectively.

Additionally, the time measurement has also been done for all the digraph order in average. So, the average time of preparing a composite service for a failure recovery has been figured out. Figure 6.5 illustrates the average value of preparation time for all the digraphs of orders 2 to 11 altogether.

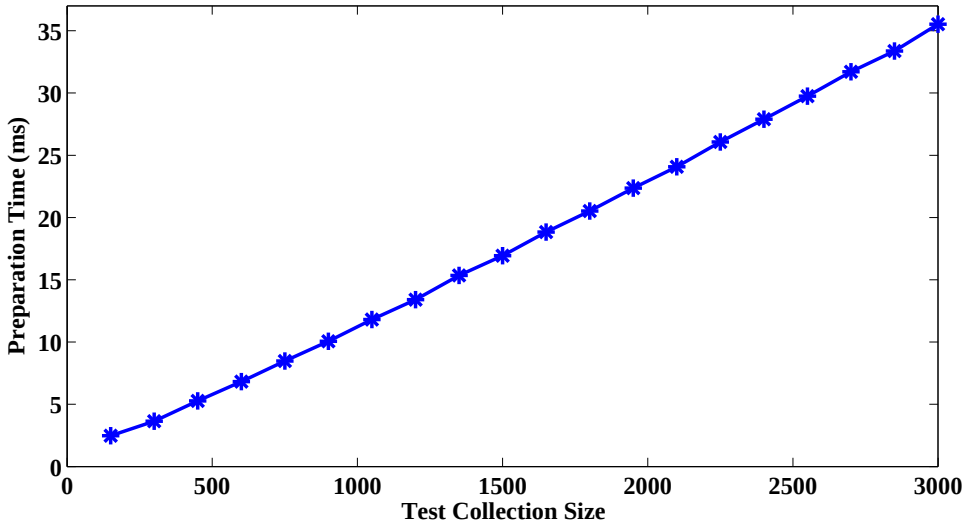


Figure 6.5.: Average Preparation Time (milliseconds) for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 11)

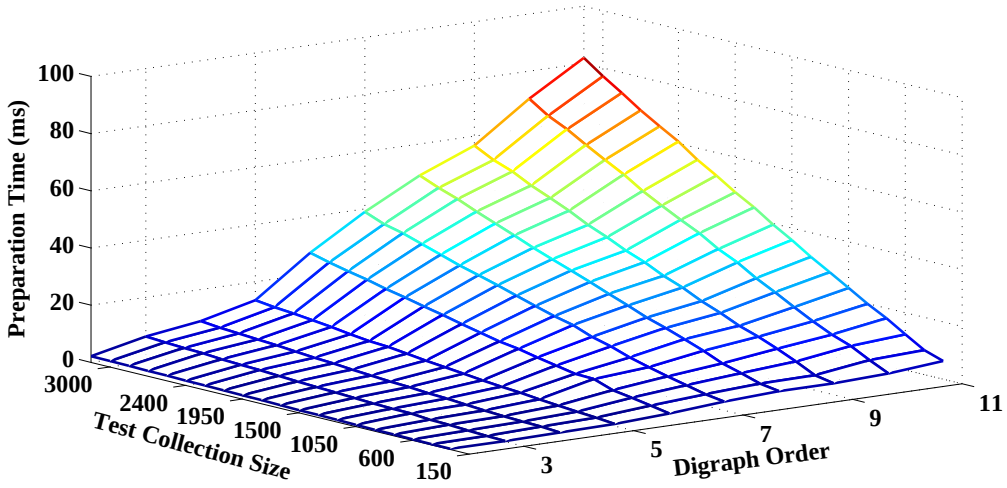


Figure 6.6.: Preparation Time (milliseconds) for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 11)

The preparation time increases as the registry's size grows. The SRPFR searches the whole registry to find suitable replacements and to rank them. The higher the digraph order of the composite service (being prepared) the higher the number of their subdigraphs, and apparently the higher their matching replacements. Table 6.2 elaborates the preparation time of the composite services in this experiment and Figure 6.6 illustrates a 3D graph of these preparation times. The values and the figure show the growth of the required time for higher digraph orders as well as bigger test collections.

Table 6.2.: Preparation Times (milliseconds) for Test Collections of 150 to 3000 Composite Services including Digraph Orders between 2 to 11

		Digraph Order									
		2	3	4	5	6	7	8	9	10	11
Test Collection Size	150	0.2	0.2	0.5	1	2	3	4	4	5	6
	300	0.3	0.4	1	1	3	4	5	5	8	9
	450	0.3	1	1	2	4	6	7	8	11	12
	600	0.5	1	1	3	5	7	9	11	13	16
	750	0.5	1	2	3	7	9	12	14	17	20
	900	1	2	2	4	8	11	14	16	20	24
	1050	1	2	3	4	9	13	16	19	23	28
	1200	1	2	3	5	10	15	19	22	27	32
	1350	1	3	4	6	11	17	21	25	31	36
	1500	1	3	4	6	12	19	24	27	34	40
	1650	1	3	4	7	14	21	26	30	38	44
	1800	1	3	5	8	15	23	29	33	41	48
	1950	1	4	5	8	17	25	31	36	44	52
	2100	2	4	6	9	18	26	33	39	48	56
	2250	2	4	6	10	20	28	36	42	52	61
	2400	2	5	7	11	21	30	39	45	55	64
	2550	2	5	7	11	23	33	41	48	59	68
	2700	2	6	8	12	24	35	44	51	63	73
	2850	2	6	8	13	26	36	46	53	66	77
	3000	2	6	9	14	27	39	49	57	70	82

The preparation times for the composite services of smaller digraph orders are relatively low. However, this (calculation) time is not so short for higher digraph orders. This extra (calculation) time is required to find the best “Original Subdigraph” and its related best “Replacement Subdigraph”, which significantly increases the recovery probability of the failed composite services. In our experiments, this preparation was scheduled to be done before commencing the (simulated) execution of the composite services. However, we proposed (in SRPFR) to transfer these time-consuming calculations to an offline phase. These offline processes should be done in an ongoing manner, ideally when the system is in idle mode. Hence, there would be no delay even before invocation of the composite services.

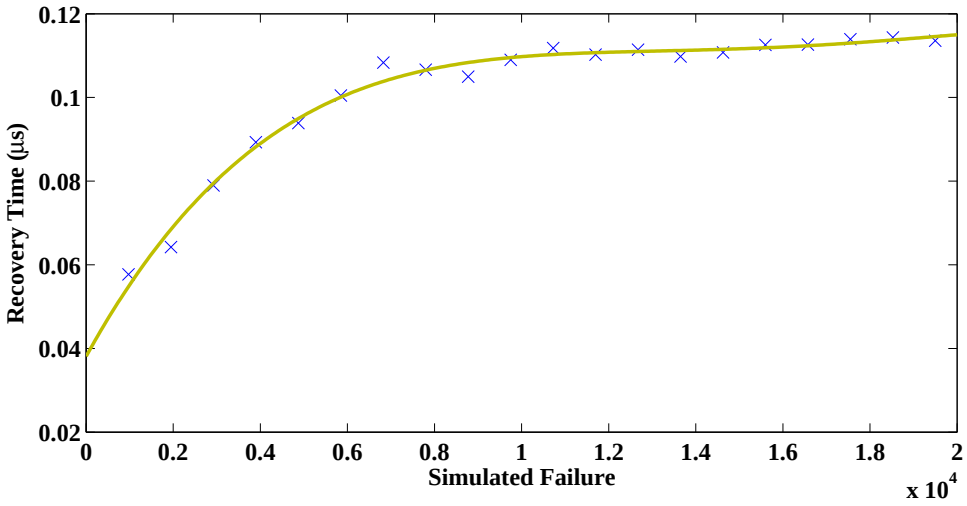


Figure 6.7.: Recovery Time for Failure Simulations in Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 11)

Finally, for the purpose of evaluating the method's response time at the failure moment, which we call Recovery Time, we calculated the average time required to perform the actual replacement. The calculations (and their interval time) at this stage is minimal due to the availability of the best choices from the previous calculations. We calculated the average time of this step for all the simulated failures separately for each iteration (increasing the size of the test collections). The results show that the recovery time is maintained at a constant value (of less than one microseconds) even for large test collections. Figure 6.7 shows the recovery time maintained at a constant value for all test collections.

The percentage of the successful failure recovery of these test collections were graphed out as well, and illustrated in Figure 6.8. We do not elaborate the successful percentage of recovery here because it was discussed earlier in Chapter 5.

6.1.3 Discussion

The execution simulations of SRPFR on a big test collection as well as on a growing set of test collections of various digraph orders reveal that the method is applicable for big test collections of composite services (from the failure recovery delay time's point of view).

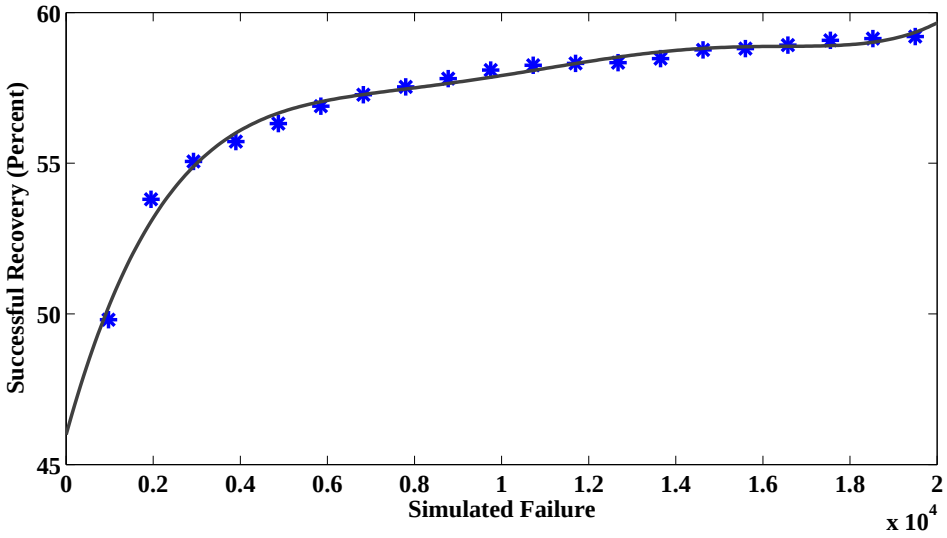


Figure 6.8.: Recovery Probability for Test Collection Sizes between 150 to 3000 Services (Digraph Orders between 2 to 11)

However, there are some conditions which need to be satisfied.

- Firstly, the method needs to initialize its registry at startup time (preceding any composite service's execution). The initialization takes a long time (i.e. a few seconds) if the test collection is not small. Hence, for a system in which the initialization must be done more than once, the initialization delay is not desirable. An example is a system which is not always on and runs on demand.
- Secondly, the preparation time for (the execution of) composite services of high order particularly in large test collections can be more than 80*ms*. The delay is definitely not desirable for real-time systems. However, if the number and the structure of the composite services in the registry are either static or change less frequently, it is possible to store the preparation results, i.e. the list of "Original Subdigraphs" and their "Replacement Subdigraph" for each component of the composite services separately and use them at a failure time. If a change occurs to the registry, the list can be updated. Moreover, in more frequent changing systems, if there is the ability of performing concurrent tasks, which ensures the preparation of the lists long before commencing the execution of the composites, the method can be used.

- Finally, at failure time there is no delay for the time-consuming calculations of finding the best set of services (the ones to be removed and to be implanted). This excludes the time which is common for all the adaptation approaches. For example every adaptation process needs to enqueue the new set of services (adapted structure) for the execution. Furthermore, we explicitly considered the availability of world-altering category of Web services in the structures of the composite services. So, we need to request for a compensation of a well-executed (world-altering) service in case that it is in a chosen "Original Subdigraph". The compensation needs its own execution time. In cases that this extra delay time is not desirable we can eliminate this feature from the system by resetting its weight in the "Original Subdigraph" Ranking Measure formula (α_4 and α_5 in Equation 3.18 on page 53).

6.2 Quality of Service (QoS) Deviation

Organizations tend to provide a reliable online service to users through Web services. Several providers may compete to offer the users a similar service. By the term similar we mean the services with the same functionality. However, the non-functional properties of the provided services may differ. The users are usually inclined to use a service with better Quality of Service (QoS). Consequently, a composite process is generated with the aim of not only offering the requested functionality but also ensuring the best available quality. A failure recovery approach is required to maintain the promised QoS even after the adaptation of the composite service's structure.

For the purpose of evaluating the QoS aspects of our method we needed various services with their identified QoS values. However, even for the atomic services available in OWLS-TC there is no QoS value. Hence, for evaluation purposes we had to generate synthetic QoS values.

6.2.1 QoS Deviation for Atomic Replacement (one-to-one)

An ideal replacement approach is to replace a failed atomic service with a similar service. The replacement service must have both functional and non-functional properties matching with the original failed service. Our method, SRPFR, replaces a failed atomic service with a matched service, if their IOPR specifications match, and their QoS attributes have a minimum difference. The matchmaking of IOPR specifications is achieved through the use of semantic annotations of the Web services. The minimization of the QoS changes is attained through the alternatives' ranking which is performed on the list of available (functionally matched) replacements. The ranking details were discussed in Section 3.2.1.

The **Research Question** is: Is SRPFR capable of choosing the best replacement (for a failed atomic service) in terms of the minimum changes to various QoS attributes?

In order to evaluate and to prove that our approach would select a replacement atomic service for a failed atomic service (during an execution) with a minimum deviation of the QoS, we created several test cases. We chose an atomic service from OWLS-TC, for which there is no matching service in the test collection, and this selection is because we purposely did not want SRPFR to find any other alternative service. We have already shown that SRPFR is able to find a replacement with a high percentage of success (Chapter 5), and our objective here is to show that among the potential alternatives, our method would choose the best.

Based on the mentioned requirement, the first service, which is not in any service class, is called "3wheeledcar year price service" (Service ID: 5 in Table A.3 provided in Appendix A on page 153). We replicated the chosen service by duplicating its functional properties (IO only because service with ID 5 has only two inputs, one output, and no precondi-

Table 6.3.: RSRM for Atomic Replacement (based on QoS Values)

Test Case	Original Atomic Service			Available Replacement Atomic Services												RSRM Weights			Selected Replacement
	ET	EC	R	1				2				3				β_2	β_3	β_4	
				ET	EC	R	RSRM	ET	EC	R	RSRM	ET	EC	R	RSRM				
1	5	10	0.8	6 ✓	11 ✓	0.7 ✓	**1.0**	7	12	0.6	0.5	8	13	0.5	0.0	$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$	1
2	5	10	0.8	7	11 ✓	0.7 ✓	**0.8**	6 ✓	12	0.6	0.7	8	13	0.5	0.0	$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$	1
3	5	10	0.8	7	11 ✓	0.7 ✓	**0.7**	6	12	0.6	0.5	5 ✓	13	0.5	0.3	$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$	1
4	5	10	0.8	7	11	0.7 ✓	0.4	6	12	0.6	0.2	2 ✓	3 ✓	0.5	**0.7**	$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$	3
5	5	10	0.8	5	10	0.85	0.3	3 ✓	8 ✓	0.9 ✓	**1**	4	11	0.8	0.2	$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$	2

ET=Execution Time (milliseconds), EC=Execution Cost (RM), R=Reliability

tion and result). Then, we assigned some different set of values for the QoS attributes, Execution Time (ET), Execution Cost (EC), and Reliability (R), to the replicated services.

For every test case, we created *three* replicas as explained above and assigned different QoS values. The number "3" for the replicas was chosen for the sake of brevity and no other purpose. Moreover, for illustration purposes, we designed most of the test cases such that they would be justifiable: "Why that particular replacement has been chosen?". However, two last cases (particularly the last one) have been designed in a way that the best alternatives are not easily detectable by looking at the QoS values and some calculations are required to select the best.

We executed SRPFR on the created set of services to identify which atomic service would be selected (to be executed) in lieu of the (simulated) failed atomic service. The experiment was repeated with different sets of QoS values. The test cases are summarized in Table 6.3. In the table, ET values are assumed to be in milliseconds (*ms*), EC values in Ringgit Malaysia (RM), and the R values in the range of $[0,1]$.

In Table 6.3, for every test case the following items and their values are identified:

- **“Original Atomic Service”**: The atomic service which is assumed to have failed, and its synthetically assigned QoS values.
- **Three separately shown “Available Replacement Atomic Services”**: The alternatives with similar functionality and different non-functional properties, and their purposely different QoS values.
  - The calculated “**RSRM**” value of every replacement is identified (Equation 3.21 on page 56).
- **“RSRM weights”**: The β values for RSRM.
- **“Selected Replacement”**: The selected replacement for every original service in a column titled “Selected Replacement”. Further, the related cells of the selected replacement are highlighted and its calculated Replacement Subdigraph Ranking Measure (RSRM) is emphasized.

  Additionally, the best QoS value (of the replacements) is ticked for Execution Time (ET), Execution Cost (EC), and Reliability (R) values separately.

The following sections elaborate the experiments.

Test Case 1

The first test case was made using a triple replication of the mentioned atomic service. The primary atomic service was assigned with the following QoS values:

- $\mathcal{ET} = 5ms$
- $\mathcal{EC} = 10$ RM
- $\mathcal{R} = 0.8$

Then, the service was replicated to three atomic services with different QoS values as follows:

1. $\mathcal{ET} = 6ms$, $\mathcal{EC} = 11$ RM, $\mathcal{R} = 0.7$
2. $\mathcal{ET} = 7ms$, $\mathcal{EC} = 12$ RM, $\mathcal{R} = 0.6$
3. $\mathcal{ET} = 8ms$, $\mathcal{EC} = 13$ RM, $\mathcal{R} = 0.5$

Henceforth, we call these three replicated atomic services as Replica 1, 2 and 3.

The values are chosen and assigned with a specific purpose and was given three different set of values for which the execution time and cost were higher than the original service, and the reliability was lower. The ET and EC values increase for the subsequent replicas and R values decrease.

Prior to the execution of the original atomic service (we call it during the preparation time), a topmost ranked replacement would be chosen. For this special case, in which the service to be executed is an atomic service, there is only one "Original Subdigraph". The only "Original Subdigraph" here is the atomic service itself. So, there is no need for measuring and ranking the original subdigraphs (Section 3.2.1). Next, for the "Finding Replacement Subdigraphs" sub-task, we tested SRPFR and it was able to find Replica 1-3 services. Finally, as in *Ranking "Replacement Subdigraphs"* (cf. Section 3.2.1) the RSRM values for Replica 1-3 was calculated. In RSRM equation (Equation 3.21), β_1 has been assigned 0, because for this test case we already know that all the replicas have one service. In order to equalize the effect of ET, EC, and R for the rank result, we assigned their weights the same number. Equation 3.22 (on page 56) constrains the sum of the weight values to be one. So, we assigned the weights as $\frac{1}{3} = 0.\overline{3}$:

$$\beta_1 = 0$$

$$\beta_2 = \beta_3 = \beta_4 = \frac{1}{3} = 0.\overline{3}$$

$$\sum_{i=1}^{4} \beta_i = 1$$

The normal values for execution time of three replicas are calculated as follows:

$$Minimum(\mathcal{ET}) = Minimum(ET_1, ET_2, ET_3) = Minimum(6,7,8) = 6$$

$$Maximum(\mathcal{ET}) = Maximum(ET_1, ET_2, ET_3) = Maximum(6,7,8) = 8$$

$$ET_1 normal = \frac{ET_1 - Minimum(\mathcal{ET})}{Maximum(\mathcal{ET}) - Minimum(\mathcal{ET})} = \frac{6-6}{8-6} = 0$$
$$ET_2 normal = \frac{ET_2 - Minimum(\mathcal{ET})}{Maximum(\mathcal{ET}) - Minimum(\mathcal{ET})} = \frac{7-6}{8-6} = 0.5$$
$$ET_3 normal = \frac{ET_2 - Minimum(\mathcal{ET})}{Maximum(\mathcal{ET}) - Minimum(\mathcal{ET})} = \frac{8-6}{8-6} = 1$$

The normal values for execution costs:

$$Minimum(\mathcal{EC}) = Minimum(EC_1, EC_2, EC_3) = Minimum(11,12,13) = 11$$

$$Maximum(\mathcal{EC}) = Maximum(EC_1, EC_2, EC_3) = Maximum(11,12,13) = 13$$

$$EC_1 normal = \frac{EC_1 - Minimum(\mathcal{EC})}{Maximum(\mathcal{EC}) - Minimum(\mathcal{EC})} = \frac{11-11}{13-11} = 0$$
$$EC_2 normal = \frac{EC_2 - Minimum(\mathcal{EC})}{Maximum(\mathcal{EC}) - Minimum(\mathcal{EC})} = \frac{12-11}{13-11} = 0.5$$
$$EC_3 normal = \frac{EC_2 - Minimum(\mathcal{EC})}{Maximum(\mathcal{EC}) - Minimum(\mathcal{EC})} = \frac{13-11}{13-11} = 1$$

The normal reliability values:

$$Minimum(\mathcal{R}) = Minimum(R_1, R_2, R_3) = Minimum(0.7, 0.6, 0.5) = 0.5$$

$$Maximum(\mathcal{R}) = Maximum(R_1, R_2, R_3) = Maximum(0.7, 0.6, 0.5) = 0.7$$

$$R_1 normal = \frac{R_1 - Minimum(\mathcal{R})}{Maximum(\mathcal{R}) - Minimum(\mathcal{R})} = \frac{0.7-0.5}{0.7-0.5} = 1$$
$$R_2 normal = \frac{R_2 - Minimum(\mathcal{R})}{Maximum(\mathcal{R}) - Minimum(\mathcal{R})} = \frac{0.6-0.5}{0.7-0.5} = 0.5$$
$$R_3 normal = \frac{R_3 - Minimum(\mathcal{R})}{Maximum(\mathcal{R}) - Minimum(\mathcal{R})} = \frac{0.5-0.5}{0.7-0.5} = 0$$

The RSRM values of all the Replicas has been calculated as follows:

$$RSRM_1 = 0 + \beta_2(1 - ET_1 normal) + \beta_3(1 - EC_1 normal) + \beta_4 R_1 normal$$

$$= 0 + 0.\overline{3} * 1 + 0.\overline{3} * 1 + 0.\overline{3} * 1 = 1$$

$$RSRM_2 = 0 + \beta_2(1 - ET_2 normal) + \beta_3(1 - EC_2 normal) + \beta_4 R_2 normal$$

$$= 0 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0.5 = 0.5$$

$$RSRM_3 = 0 + \beta_2(1 - ET_3 normal) + \beta_3(1 - EC_3 normal) + \beta_4 R_3 normal$$

$$= 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0 = 0$$

Among the three RSRM values, $RSRM_1$ has the greatest value. So, the atomic service, number 1, is chosen as the selected replacement (which we called "Replacement Subdigraph" in general) to be switched with the original atomic service (in case of a failure at the execution time). The first row (Test Case 1) of Table 6.3 summarizes the calculated values for this test case.

For Test Case 1, the reason that SRPFR selected the first replica is that this replica has the lowest deviation of the three QoS attributes among the others. So, obviously it is the best replacement choice.

$$\boldsymbol{ET_1} < ET_2 < ET_3\,, (\mathbf{6} < 7 < 8) \quad \checkmark$$

$$\boldsymbol{EC_1} < EC_2 < EC_3\,, (\mathbf{11} < 12 < 13) \quad \checkmark$$

$$\boldsymbol{R_1} > R_2 > R_3\,, (\mathbf{0.7} > 0.6 > 0.5) \quad \checkmark$$

Test Case 2

The second test case has been made similar to the first one. The primary atomic service's QoS attributes are the same as the first test. The three replicated services' QoS values are as follows:

1. $\mathcal{ET} = \mathbf{7}ms$, $\mathcal{EC} = 11$ RM, $\mathcal{R} = 0.7$

2. $\mathcal{ET} = \mathbf{6}ms$, $\mathcal{EC} = 12$ RM, $\mathcal{R} = 0.6$

3. $\mathcal{ET} = 8ms$, $\mathcal{EC} = 13$ RM, $\mathcal{R} = 0.5$

The only difference between Test Cases 1 and 2 is the execution time of the first and second replicas. We switched their ET values. The changed formulas are shown as follows:

$$ET_1 normal = \frac{ET_1 - Minimum(\mathcal{ET})}{Maximum(\mathcal{ET}) - Minimum(\mathcal{ET})} = \frac{7-6}{8-6} = 0.5$$

$$ET_2 normal = \frac{ET_2 - Minimum(\mathcal{ET})}{Maximum(\mathcal{ET}) - Minimum(\mathcal{ET})} = \frac{6-6}{8-6} = 0$$

Other values, including $ET_3 normal$, EC normal values, and R normal values are equal to the values in Test Case 1. The RSRM values are:

$$RSRM_1 = 0 + 0.\overline{3} * 0.5 + 0.\overline{3} * 1 + 0.\overline{3} * 1 = 0.8$$

$$RSRM_2 = 0 + 0.\overline{3} * 1 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0.5 = 0.7$$

$$RSRM_3 = 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0 = 0$$

Therefore, in this test case $RSRM_1$ has the greatest value as well. So, similarly it will be chosen. The second row (Test Case 2) of Table 6.3 summarizes the calculated values for this test case.

For this test case, the reason of selecting the first replica is that this replica has the lowest deviation of two out of three QoS attributes. So, it is still the best replacement choice.

$$ET_2 < \boldsymbol{ET_1} < ET_3\,, (6 < \mathbf{7} < 8) \quad \text{✗}$$

$$\boldsymbol{EC_1} < EC_2 < EC_3\,, (\mathbf{11} < 12 < 13) \quad \text{✓}$$

$$\boldsymbol{R_1} > R_2 > R_3\,, (\mathbf{0.7} > 0.6 > 0.5) \quad \text{✓}$$

Test Case 3

The third test case has been made similar to the second one with the only change of decreasing its third service's ET value from 8 to 5. The QoS values are as follows:

1. $\mathcal{ET} = 7ms$, $\mathcal{EC} = 11$ RM, $\mathcal{R} = 0.7$

2. $\mathcal{ET} = 6ms$, $\mathcal{EC} = 12$ RM, $\mathcal{R} = 0.6$

3. $\mathcal{ET} = \mathbf{5}ms$, $\mathcal{EC} = 13$ RM, $\mathcal{R} = 0.5$

The normal values for execution times:

$$Minimum(\mathcal{ET}) = Minimum(7,6,5) = 5$$

$$Maximum(\mathcal{ET}) = Maximum(7,6,5) = 7$$

$$ET_1 normal = \frac{7-5}{7-5} = 1$$

$$ET_2 normal = \frac{6-5}{7-5} = 0.5$$

$$ET_3 normal = \frac{5-5}{7-5} = 0$$

The other values have no change. The RSRM values:

$$RSRM_1 = 0 + 0.\overline{3} * 0 + 0.\overline{3} * 1 + 0.\overline{3} * 1 = 0.7$$

$$RSRM_2 = 0 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0.5 = 0.5$$

$$RSRM_3 = 0 + 0.\overline{3} * 1 + 0.\overline{3} * 0 + 0.\overline{3} * 0 = 0.3$$

Hence, also in this test case $RSRM_1$ has the greatest value and its atomic service will be chosen. The third row (Test Case 3) of Table 6.3 summarizes these calculated values.

The reason for selecting the first replica here is that again this replica has the lowest deviation of two out of three QoS attributes. So, it is again the best replacement choice.

$$ET_3 < ET_2 < \boldsymbol{ET_1}, (5 < 6 < \mathbf{7}) \quad ✗$$

$$\boldsymbol{EC_1} < EC_2 < EC_3, (\mathbf{11} < 12 < 13) \quad ✓$$

$$\boldsymbol{R_1} > R_2 > R_3, (\mathbf{0.7} > 0.6 > 0.5) \quad ✓$$

Test Case 4

The fourth test case has been made similar to the third one with two changes on ET and EC of the third replica. The QoS values are as follows:

1. $\mathcal{ET} = 7ms$, $\mathcal{EC} = 11$ RM, $\mathcal{R} = 0.7$

2. $\mathcal{ET} = 6ms$, $\mathcal{EC} = 12$ RM, $\mathcal{R} = 0.6$

3. $\mathcal{ET} = \mathbf{2}ms$, $\mathcal{EC} = \mathbf{3}$ RM, $\mathcal{R} = 0.5$

The normal values for execution times and costs:

$$Minimum(\mathcal{ET}) = Minimum(7,6,2) = 2$$

$$Maximum(\mathcal{ET}) = Maximum(7,6,2) = 7$$

$$ET_1 normal = \frac{7-2}{7-2} = 1$$

$$ET_2 normal = \frac{6-2}{7-2} = 0.8$$

$$ET_3 normal = \frac{2-2}{7-2} = 0$$

$$Minimum(\mathcal{EC}) = Minimum(11,12,3) = 3$$

$$Maximum(\mathcal{EC}) = Maximum(11,12,3) = 12$$

$$EC_1 normal = \frac{11-3}{12-3} = 0.9$$

$$EC_2 normal = \frac{12-3}{12-3} = 1$$

$$EC_3 normal = \frac{3-3}{12-3} = 0$$

The R values and their normal values have no change. The RSRM values:

$$RSRM_1 = 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0.1 + 0.\overline{3} * 1 = 0.4$$

$$RSRM_2 = 0 + 0.\overline{3} * 0.2 + 0.\overline{3} * 0 + 0.\overline{3} * 0.5 = 0.2$$

$$RSRM_3 = 0 + 0.\overline{3} * 1 + 0.\overline{3} * 1 + 0.\overline{3} * 0 = 0.7$$

Therefore, in this test case $RSRM_3$ has the greatest value and its atomic service will be chosen. The fourth row (Test Case 4) of Table 6.3 summarizes these calculated values.

In this case, selecting the third replica is because its ET and EC values are lowest even though its R value is not the highest.

$$\boldsymbol{ET_3} < ET_2 < ET_1\,, (\mathbf{2} < 6 < 7) \quad \checkmark$$

$$\boldsymbol{EC_3} < EC_1 < EC_2\,, (\mathbf{3} < 11 < 12) \quad \checkmark$$

$$R_1 > R_2 > \boldsymbol{R_3}\,, (0.7 > 0.6 > \mathbf{0.5}) \quad \boldsymbol{\times}$$

Test Case 5

The RSRM values:

$$RSRM_1 = 0 + 0.\overline{3} * 0 + 0.\overline{3} * 0.34 + 0.\overline{3} * 0.5 = 0.3$$

$$RSRM_2 = 0 + 0.\overline{3} * 1 + 0.\overline{3} * 1 + 0.\overline{3} * 1 = 1$$

$$RSRM_3 = 0 + 0.\overline{3} * 0.5 + 0.\overline{3} * 0 + 0.\overline{3} * 0 = 0.2$$

So, in this test case $RSRM_2$ has the greatest value and its atomic service will be chosen. The fifth row (Test Case 5) of Table 6.3 summarizes these calculated values.

The selection of the second replica is due to its lowest QoS values among the others.

$$\boldsymbol{ET_2} < ET_3 < ET_1\,, (\mathbf{3} < 4 < 5) \quad \checkmark$$

$$\boldsymbol{EC_2} < EC_1 < EC_3\,, (\mathbf{8} < 10 < 11) \quad \checkmark$$

$$\boldsymbol{R_2} > R_1 > R_3\,, (\mathbf{0.9} > 0.85 > 0.8) \quad \checkmark$$

Discussion

For the above 5 test cases, we executed SRPFR and we got the same results as shown in the last column of Table 6.3. For all the cases SRPFR could choose a replacement (atomic service) such that its QoS deviation is the lowest. Test Case 5 is a special case because its selected replacement does not show the **minimum QoS change** but the maximum change. The reason for this difference is because in SRPFR we assume that the original service in which one of its components failed, has the best QoS values, i.e. if we compare and rank the original service together with its replacements, we would see that the original service

stands at the top of the list. However, if there is a case in which a replacement service has better QoS values, this **better** service will be chosen even though the QoS deviation is high (but in a positive direction). Therefore, the replacement service has always the best QoS among the available replacements (Test Cases 1-5) and sometimes (Test Case 5) even a better QoS in comparison with the original service.

In summary, in order to answer the research question for this section (provided on page 112): Yes, SRPFR assures the selection of the best replacement in terms of QoS.

6.2.2 QoS Deviation for Composite Replacement (many-to-many)

The actual contribution of SRPFR in terms of minimum QoS deviation is for composite services. We examined our approach and the results show that SRPFR is able to adapt the structure of a composite so that its QoS attributes do not have a significant change. This is only possible if there are multiple alternatives of replacement for a particular failure. However, the existance of the choices are highly feasible because of SRPFR's specific features such as:

1. Calculating the subdigraphs of the available composite services in the registry and storing them as separate composites.
2. Going backwards through the structure of the composite service to find the so-called "Original Subdigraph" which may even contain some well-executed services.

SRPFR ranks the available choices based on the multiple criteria to find the best "Original Subdigraph" to be removed and the best "Replacement Subdigraph" to be placed in the structure for each and every "AFWS".

For evaluation purposes, we chose one of the composite services in a synthetically generated test collection. Then, we performed our approach on this composite service. The

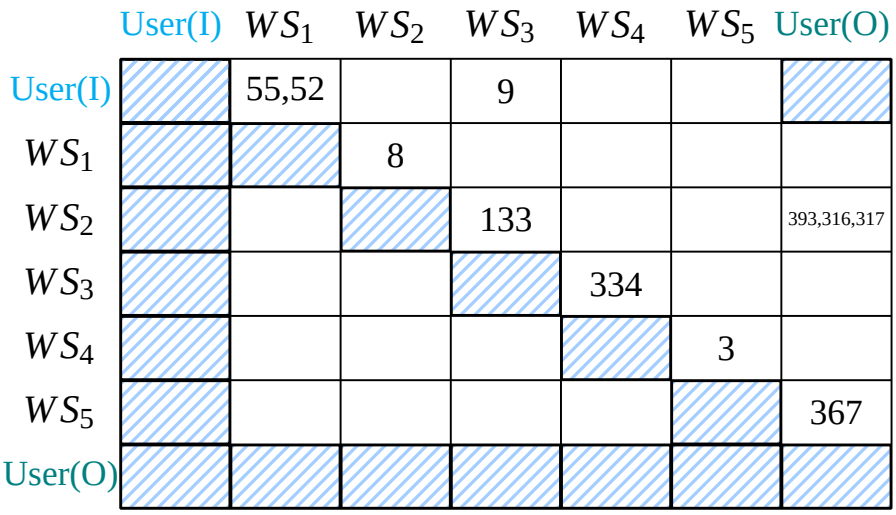


Figure 6.9.: Adjacency Matrix of the Composite Service. Coordinates are User Inputs, Constituent Services (WS_1..WS_6), and User Outputs

results are shown in Tables 6.4, 6.5 and 6.6.

The composite service, which its adjacency matrix is depicted in Figure 6.9, includes 5 atomic services. The following list shows the atomic services with their inputs and outputs. The input and output numbers are the IDs that we assigned to the concepts available in the ontologies of OWLS-TC.

- Atomic 1 (WS_1): Name="Expensive Car Recommended Price",

  Inputs= [55, 52], Output= [8], QoS=[ET=16, EC=5, R=0.810]

- Atomic 2 (WS_2): Name="RPCW brand",

  Input= [8], Outputs= [393, 133, 316, 317], QoS=[ET=16, EC=5, R=0.844]

- Atomic 3 (WS_3): Name="ColaProvider",

  Inputs= [9, 133], Output= [334], QoS=[ET=30, EC=7, R=0.849]

- Atomic 4 (WS_4): Name="Guddu Cola",

  Input= [334], Output= [3], QoS=[ET=20, EC=0, R=0.964]

- Atomic 5 (WS_5): Name="CoffeewithWhiskeyPrice",

  Input= [3], Output= [367], QoS=[ET=29, EC=7, R=0.909]

Table 6.4.: Ranking for Composite Replacement (based on QoS values), Atomic Services 1 and 2

AFWS=WS_1, Code=115														
Original Subdigraph					Replacement Subdigraph						Changes from Original to Replacement			
Service List	Digraph Order	ET	EC	R	Service List	Digraph Order	ET	EC	R	Rank	Digraph Order	ET	EC	R
[115.1606]	2	32	10	0.684	[114.1605]	2	24	8	0.184	**0.708**	0	-8	-2	-0.500

AFWS=WS_2, Code=1606														
Original Subdigraph					Replacement Subdigraph						Changes from Original to Replacement			
Service List	Digraph Order	ET	EC	R	Service List	Digraph Order	ET	EC	R	Rank	Digraph Order	ET	EC	R
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1521]	3	20	8	0.299	0.823	0	-46	-4	-0.392
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1519]	3	42	9	0.166	0.674	0	-24	-3	-0.525
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1525]	4	39	16	0.230	0.526	2	-7	4	-0.487
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1523]	4	54	9	0.100	0.510	2	8	-3	-0.617
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1524]	4	60	12	0.074	0.437	2	14	0	-0.643
[1606.1591]	2	46	12	0.717	[1605.1590]	2	15	8	0.300	**0.868**	0	-31	-4	-0.417
[1606.1591]	2	46	12	0.717	[1605.1590.1521.1523]	4	39	9	0.104	0.554	2	-7	-3	-0.613
[1606.1591]	2	46	12	0.717	[1605.1590.1519.1525]	4	46	17	0.132	0.438	2	0	5	-0.585
[115.1606]	2	32	10	0.684	[114.1605]	2	24	8	0.184	0.744	0	-8	-2	-0.500

Table 6.5.: Ranking for Composite Replacement (based on QoS values), Atomic Service 3

AFWS=WS_3, Code=1591														
Original Subdigraph					Replacement Subdigraph						Changes from Original to Replacement			
Service List	Digraph Order	ET	EC	R	Service List	Digraph Order	ET	EC	R	Rank	Digraph Order	ET	EC	R
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1519]	3	42	9	0.166	0.653	0	-24	-3	-0.525
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1521]	3	20	8	0.299	0.774	0	-46	-4	-0.392
[1591.1520]	2	50	7	0.819	[1590.1519]	2	32	7	0.210	0.631	0	-18	0	-0.609
[1591.1520]	2	50	7	0.819	[1590.1521.1523.1519]	4	56	8	0.073	0.379	2	6	1	-0.746
[1591.1520]	2	50	7	0.819	[1590.1521]	2	10	6	0.379	0.767	0	-40	-1	-0.439
[1591.1520]	2	50	7	0.819	[1590.1519.1522.1521]	4	45	14	0.194	0.388	2	-5	7	-0.624
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1524]	4	60	12	0.074	0.457	2	14	0	-0.643
[1606.1591]	2	46	12	0.717	[1605.1590.1519.1525]	4	46	17	0.132	0.459	2	0	5	-0.585
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1523]	4	54	9	0.100	0.517	2	8	-3	-0.617
[1606.1591]	2	46	12	0.717	[1605.1590.1520.1525]	4	39	16	0.230	0.528	2	-7	4	-0.487
[1606.1591]	2	46	12	0.717	[1605.1590.1521.1523]	4	39	9	0.104	0.556	2	-7	-3	-0.613
[1606.1591]	2	46	12	0.717	[1605.1590]	2	15	8	0.300	**0.832**	0	-31	-4	-0.417

Table 6.6.: Ranking for Composite Replacement (based on QoS values), Atomic Services 4, and 5

AFWS=WS_4, Code=1520														
Original Subdigraph					Replacement Subdigraph						Changes from Original to Replacement			
Service List	Digraph Order	ET	EC	R	Service List	Digraph Order	ET	EC	R	Rank	Digraph Order	ET	EC	R
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1519]	3	42	9	0.166	0.633	0	-24	-3	-0.525
[1606.1591.1520]	3	66	12	0.691	[1605.1590.1521]	3	20	8	0.299	**0.763**	0	-46	-4	-0.392
[1591.1520]	2	50	7	0.819	[1590.1519]	2	32	7	0.210	0.606	0	-18	0	-0.609
[1591.1520]	2	50	7	0.819	[1590.1521.1523.1519]	4	56	8	0.073	0.344	2	6	1	-0.746
[1591.1520]	2	50	7	0.819	[1590.1521]	2	10	6	0.379	0.750	0	-40	-1	-0.439
[1591.1520]	2	50	7	0.819	[1590.1519.1522.1521]	4	45	14	0.194	0.329	2	-5	7	-0.624

AFWS=WS_5, Code=1518
No Replacement Available

Based on Equations 3.10, 3.11, and 3.13 provided in Definition 5 (Chapter 3 on page 43) the QoS values of the composite service are as follows:

$$\mathcal{ET}_{Composite} = \sum_{i=1}^{|\mathcal{W}|} \mathcal{ET}_{w_i} = 111$$

$$\mathcal{EC}_{Composite} = \sum_{i=1}^{|\mathcal{W}|} \mathcal{EC}_{w_i} = 24$$

$$\mathcal{R}_{Composite} = \prod_{i=1}^{|\mathcal{W}|} \mathcal{R}_{w_i} = 0.51$$

The weights for the Original Subdigraph Ranking Measure (OSRM) (Equation 3.18 on page 53) have been assigned equally as $\frac{1}{6} = 0.1\overline{6}$ in order to equalize their effects.

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = \frac{1}{6} = 0.1\overline{6}, \quad \sum_{i=1}^{6} \alpha_i = 1$$

Similarly, to make the effects of various QoS attributes the same, the weights for the Replacement Subdigraph Ranking Measure (RSRM) (Equation 3.21 on page 56) have been assigned equally as $\frac{1}{4} = 0.25$.

$$\beta_1 = \beta_2 = \beta_3 = \beta_4 = \frac{1}{4} = 0.25, \quad \sum_{i=1}^{4} \beta_i = 1$$

The weights for the final Rank (Equation 3.23 on page 56) have been assigned equally as $\frac{1}{2} = 0.5$.

$$\gamma_1 = \gamma_2 = \frac{1}{2} = 0.5, \quad \gamma_1 + \gamma_2 = 1$$

For every atomic service of the composite service the replacements were ranked as follows.

For Atomic 1 (WS_1) there was only one “Replacement Subdigraph”. Hence, SRPFR was not required to rank the alternatives. The “Original Subdigraph” included two atomics (WS_1 and WS_2). The only “Replacement Subdigraph” had two atomics as well. The final

rank values is calculated:

$$\text{“}\textit{Original Subdigraph}\text{” }\textit{Ranking Measure} =$$

$$\frac{1}{6}(1-0)+\frac{1}{6}*0+\frac{1}{6}*0+\frac{1}{6}(1-0)+\frac{1}{6}(1-0)+\frac{1}{6}(1-0)=0.\overline{6}$$

$$\text{“}\textit{Replacement Subdigraph}\text{” }\textit{Ranking Measure} =$$

$$\frac{1}{4}(1-0)+\frac{1}{4}(1-0)+\frac{1}{4}(1-0)+\frac{1}{4}*0=0.75$$

$$\textit{Rank}=\frac{1}{2}*0.\overline{6}+\frac{1}{2}*0.75=0.708$$

The Atomic 2 (WS_2) had 9 alternatives with different “Original Subdigraphs” and “Replacement Subdigraphs”. These subdigraphs have different orders and various QoS attributes. The highest rank belongs to the original subdigraph which has WS_2 and WS_3 as their components, together with its replacement subdigraphs with service numbers 1605 and 1590 respectively. Its rank value equals to 0.868 which is calculated as follows.

$$\text{“}\textit{Original Subdigraph}\text{” }\textit{Ranking Measure} =$$

$$\frac{1}{6}(1-0)+\frac{1}{6}*0.412+\frac{1}{6}*1+\frac{1}{6}(1-0)+\frac{1}{6}(1-0)+\frac{1}{6}(-0.031)=0.730$$

$$\text{“}\textit{Replacement Subdigraph}\text{” }\textit{Ranking Measure} =$$

$$\frac{1}{4}(1-0)+\frac{1}{4}(1-0)+\frac{1}{4}(1-0)+\frac{1}{4}*1=1$$

$$\textit{Rank}=\frac{1}{2}*0.730+\frac{1}{2}*1=0.868$$

The replacements for the other AFWSs were calculated similarly. The rankings are shown in Table 6.5 (for WS_3) and Table 6.6 (for WS_4 and WS_5). For WS_5 there was no replacement to be ranked. Hence, if the Web service, WS_5, fails during an execution, SRPFR is not able to recover it, i.e. either by replacing it with another similar service or by any other recovery type.

6.2.3 Discussion

We have shown that SRPFR ensures the recovery of a failure of a composite service to be performed such that the QoS value is maintained close to the QoS values of the original service. Additionally, if there is an alternative set of services with a better QoS (lower execution time for instance) this beneficial replacement would be chosen.

The weights for the ranking formula significantly change the final decision. For example in cases that the execution time of the services has a higher priority, its respective weight should be assigned with a higher value. Therefore, by adjusting the weights, the different QoS attributes can be prioritized such that the adaptation process can choose the best alternative. In order to find an optimized set of values for the weights, there should be a test case with valid judgments which shows the best “Original Subdigraph” and “Replacement Subdigraph” for every set of alternatives of the AFWSs. Unfortunately, because there is no such test data available we could not propose the exact values.

Chapter 7: Conclusion

In this chapter, we summarize the work done and conclude its results. Then, we illuminate future work on failure recovery of composite semantic Web services.

7.1 Summary of Research Work

We started this research with the main objective of increasing the probability of recovery of composite Web services' execution from a failure. Hence, we proposed a method, SRPFR, that could adapt the structure of composite Web services. Besides, the similarity of the original composite and the adapted one must be high enough, so the user would get the requested goal. However, we had to keep the adaptation's delay time very low so that the end user would not feel the interruption.

Along the way, we found that we had to develop an application which is able to generate a test collection of composite Web services with particular customizable features. We needed the data set to evaluate our work and compare the result with the works of others.

A summary of the study and the **contributions** are as follows:

- We proposed a two-phase method, SRPFR, for the failure recovery of composite Web services. The phases are divided into design-time, offline, and run-time, online. The preparations and the time-consuming calculations are scheduled to be executed before commencing the invocation of the composite service. The actual replacement is triggered at the time of failure using the available list which was prepared in the offline phase.

  In order to get the best choices for the replacement, we rank the candidates based on various features to find both the best set to be removed and the best one to be implanted.

- In the real world, we could not find a data set of composite semantic Web services to test our approach. Therefore, we developed a method **to generate a big set of composite semantic Web services** with customizable features. The method is flexible enough to generate a desired number of composite Web services.

  The composite services are generated using either a pre-defined set of atomic services and their concepts defined in some ontologies or a synthesized set of atomic services with a hypothetical hierarchy of concepts. For the purpose of evaluating our method, we generated the composite services based on a standard test collection of atomic services called OWLS-TC.

  The composites are created using IO-relation (which is for traditional and semantic Web services) as well as condition-relation (Precondition and Effect (PE)-related), which is specifically for semantic Web services, of the atomic services.

- We implemented our proposed method, and examined it on a number of test collections. The results show that we achieved **a significant positive change in the probability of recovery** of composite Web services from an execution failure.

  We could recover the services from a failure with a probability of higher than **55%** for a big set of composite services (including various digraph orders). We compared our method with others from the probability of recovery aspects. The comparison shows a similar probability of recovery for atomic services, a slightly better percentage when composite services are available along with the atomic services, and finally a higher probability of recovery of SRPFR as compared to the others for composite services (of larger digraph orders).

  A major reason that our method has a higher probability of recovery is our searching allows us to go backwards through the structure of the composite Web services. The other methods do not allow this backwards movement and they mostly adapt the

structure of the composite service starting from the failed service.

The backwards movement requires the compensation of the well-executed Web services, hence the explicit consideration of the **world-altering** category of semantic Web services. We explicitly considered the availability of the world-altering services and we undo their effects if they are required to be removed from the structure.

Another reason for a high percentage of recovery as compared to the other methods is that we calculate the subdigraphs of the available composite services in the registry and we store these smaller composite services as well. Hence, through our searching we may find these smaller composites as the replacements for a set of services.

- A critical requirement for a failure recovery method is the delay time to adapt the composite service's structure at the time of failure. We examined our method to measure the adaptation delay. The result shows that **the recovery delay time is almost zero**. This minimum required time is because we transferred the time-consuming calculations to a phase prior to the execution of the composite service.
- Our final contribution is keeping the QoS of the adapted service similar to the original composite service which had faced a failure. We chose the replacement set of services based on the similarity of its non-functional properties other than their functional properties. Therefore, there is **minimum deviation on the QoS**, i.e. the best set of services has been chosen which have the most similar QoS values.
- The method has a feature to enable the administrator of the system to prioritize the required non-functional properties. This can be done by adjusting the weights of the ranking formula for both the set of services to be removed and their replacement set. The ranking is based on the number of services in the structure, their undo needs, execution time and cost, and reliability values of the components.

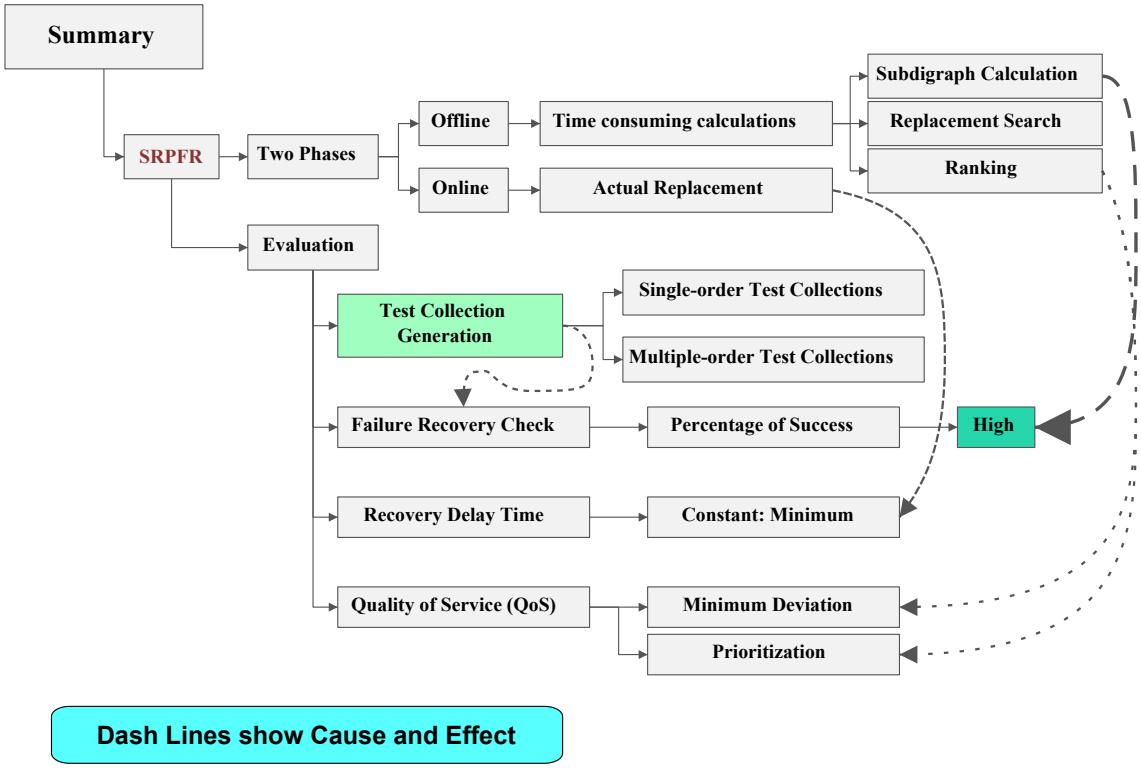


Figure 7.1.: Summary and Contributions

Figure 7.1 depicts a **summary** of the work and its contributions: We developed a two-phase method, **SRPFR**, for failure recovery of composite semantic Web services (by adapting their structure) which is capable of handling world-altering services. We also created a method which is able **to generate a customizable test collection** of composite services, which enabled us to prove the ability of our method in increasing the percentage of success of the recovery to a **high** degree (as compared to the other well-known methods). The high probability is because the method adds the subdigraphs of the composites (as new composite services) to the registry and goes backwards through the structure of the failed composite service. The delay of recovery at the moment of failure is maintained at **a minimum constant value** because we only perform the actual replacement at the onset of failure and its required calculations at the offline phase. Finally, through the use of a ranking function we could **prioritize** the needed QoS attributes and we were able to assert their **minimum deviation** from the initial values (even in case of an adaptation).

7.2 Limitations and Future Work

- The service matchings of our composites, IO, IOR, and IOPR, have been chosen to be exact matches. Other levels of matching have been proposed by others. A future work is to consider other levels of matching in order to discover more alternatives (for replacement purposes).

  For example considering sub-concept, and super-concept for matching between the inputs and outputs of to-be-joined services enables the discovery process to match more services.

- The test collections, which we generated for our evaluation purposes, are solely based on concepts defined in shared ontologies. However, it has been shown that for matchmaking purposes, other similarity measures such as text based similarity are important as well (Bravo & Alvarado, 2010; Ganjisaffar et al., 2006).

  For example, among the synthetically generated composite services we found some services which are technically possible to be created in terms of their joining concepts. However, looking at the name of the services (which are composed together) reveals that they are meaningless to be generated. The text based similarity, for example, can eliminate such undesired services.

- The failure recovery algorithm proposed in this research and some others such as CSPB (Yu & Lin, 2005a) can handle a single point of failure (at most one Web service is disabled at a time) efficiently. Actually, if the other points of failures are out of the chosen set of services to be removed (which we called “Original Subdigraph”), SRPFR can find a solution for it.

  However, a mechanism is needed to ensure that if there are multiple points of failures they are recoverable in general.

- The vertices of our composite services are the atomic Web services, i.e. the points of functionality. However, the control flow gateways such as OR, AND, and XOR are not considered. A service composition can contain more complex control flows such as choice and concurrent behaviors. A future work is to consider them and revise the representation of the composite structures to cover the aforementioned structures.
- The full applicability of our proposed method in terms of its delay time would be best shown when there is a complete implementation of the mediator (Figure 1.1 on page 3). The mediator has to support the ongoing execution of the offline phase of SRPFR. By this enhancement, the time-consuming calculations would be performed much earlier than commencing the execution of the composite services. So, even the delay before the execution of the composite services would be lessened.

Bibliography

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., & Verma, K. (2005). *Web Service Semantics - WSDL-S*. W3C member submission, WWW Consortium.

Alves, A., Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Sterling, König, D., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., & Yiu, A. (2007). Web Services Business Process Execution Language Version 2.0. OASIS Committee Draft.

Angarita, R., Cardinale, Y., & Rukoz, M. (2012). FaCETa: Backward and forward recovery for execution of transactional composite ws. In *International Workshop on REsource Discovery (RED)* (pp. 89–103). Heraklion, Greece.

Bai, X., Dong, W., Tsai, W.-T., & Chen, Y. (2005). WSDL-based automatic test case generation for web services testing. In *IEEE International Workshop on Service-Oriented System Engineering (SOSE)* (pp. 215–220). Washington, DC, USA.

Bang-Jensen, J. & Gutin, G. Z. (2009). *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics. Springer London, 2nd edition.

Bener, A. B., Ozadali, V., & İlhan, E. S. (2009). Semantic matchmaker with precondition and effect matching using SWRL. *Expert Systems with Applications*, 36(5), 9371–9377.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web.

Bhiri, S., Gaaloul, W., & Godart, C. (2006). Discovering and improving recovery mechanisms of composite web services. In *International Conference on Web Services (ICWS)* (pp. 99–110).

Bhiri, S., Gaaloul, W., Godart, C., Perrin, O., Zaremba, M., & Derguech, W. (2011). Ensuring customised transactional reliability of composite services. *Journal of Database Management*, 22(2), 64–92.

Bieberstein, N., Laird, R. G., Jones, K., & Mitra, T. (2008). *Executing SOA: A Practical Guide for the Service-Oriented Architect*. IBM Press, 1st edition.

Bondy, J. A. & Murty, U. S. R. (2008). *Graph theory*, volume 244 of *Graduate texts in Mathematics*. New York: Springer.

Booth, D., Haas, H., Mccabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. (2004). *Web Services Architecture*. Technical report, World Wide Web Consortium.

Bravo, M. & Alvarado, M. (2010). Similarity measures for substituting web services. *International Journal of Web Services Research (IJWSR)*, 7(3), 1–29.

Bruijn, J. d., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., & Stollberg, M. (2005). Web Service Modeling Ontology (WSMO). Website.

Cabral, L., Li, N., & Kopecký, J. (2012). Building the WSMO-Lite test collection on the SEALS platform. In *Second International Workshop on Evaluation of Semantic Technologies (IWEST)* (pp. 37–48). Heraklion, Greece.

Cabral, L., Li, N., Tymaniuk, S., & Winkler, D. (2011). *D14.4 Evaluation Design and Collection of Test data for Semantic Web Services Tools v2*. Technical report, Semantic Evaluation at Large Scale (SEALS).

Canfora, G., Di Penta, M., Esposito, R., & Villani, M. L. (2005). QoS-aware replanning of composite web services. In *IEEE International Conference on Web Services (ICWS)* (pp. 121–129).

Canfora, G., Di Penta, M., Esposito, R., & Villani, M. L. (2008). A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software*, 81, 1754–1769.

Cardoso, J., Sheth, A. P., Miller, J. A., Arnold, J., & Kochut, K. (2004). Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3), 281–308.

Chafle, G., Dasgupta, K., Kumar, A., Mittal, S., & Srivastava, B. (2006). Adaptation in web service composition and execution. In *IEEE International Conference on Web Services (ICWS)* (pp. 549–557). Chicago, USA: IEEE Computer Society.

Chafle, G., Doshi, P., Harney, J., Mittal, S., & Srivastava, B. (2007). Improved adaptation of web service compositions using value of changed information. In *IEEE International Conference on Web Services (ICWS)* (pp. 784–791). Los Alamitos, CA, USA.

Chinnici, R., Moreau, J.-J., Ryman, A., & Weerawarana, S. (2007). *Web Services Description Language (WSDL) Version 2.0*. W3C recommendation, WWW Consortium.

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1*. W3C note, World Wide Web Consortium.

Dai, Y., Yang, L., & Zhang, B. (2009). QoS-driven self-healing web service composition based on performance prediction. *Journal of Computer Science and Technology*, 24, 250–261.

Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., & Pohl, K. (2008). A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3-4), 313–341.

Erl, T. (2007). *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Upper Saddle River, NJ, USA: Prentice Hall.

Feng, X., Wang, H., Wu, Q., & Zhou, B. (2007a). An adaptive algorithm for failure recovery during dynamic service composition. In A. Ghosh, R. De, & S. Pal (Eds.), *Pattern Recognition and Machine Intelligence*, volume 4815 of *Lecture Notes in Computer Science* (pp. 41–48). Springer Berlin / Heidelberg.

Feng, X., Wu, Q., Wang, H., Ren, Y., & Guo, C. (2007b). ZebraX: A model for service composition with multiple QoS constraints. In C. Cérin & K.-C. Li (Eds.), *Advances in Grid and Pervasive Computing*, volume 4459 of *Lecture Notes in Computer Science* (pp. 614–626). Springer Berlin / Heidelberg.

Fensel, D., Fischer, F., Kopecký, J., Krummenacher, R., Lambert, D., & Vitvar, T. (2010). WSMO-Lite: Lightweight semantic descriptions for services on the web. Website.

Ganek, A. G. & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1), 5–18.

Ganjisaffar, Y., Abolhassani, H., Neshati, M., & Jamali, M. (2006). A similarity measure for OWL-S annotated web services. In *IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 621–624). Los Alamitos, CA, USA: IEEE Computer Society.

Ganjisaffar, Y. & Saboohi, H. (2006). Semantic web services' test collection SWS-TC. Available online at: http://www.semwebcentral.org/projects/sws-tc/.

Ghallab, M., Isi, C. K., Penberthy, S., Smith, D. E., Sun, Y., & Weld, D. (1998). *PDDL - The Planning Domain Definition Language*. Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Greenfield, P., Fekete, A., Jang, J., & Kuo, D. (2003). Compensation is not enough. In *Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC)* (pp. 232–239).: IEEE Computer Society.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., & Lafon, Y. (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C recommendation, W3C.

Guo, H., Huai, J., Li, H., Deng, T., Li, Y., & Du, Z. (2007). Angel: Optimal configuration for high available service composition. In *IEEE International Conference on Web Services (ICWS)* (pp. 280–287).

Harney, J. & Doshi, P. (2006). Adaptive web processes using value of changed information. In *4th International Conference on Service-Oriented Computing (ICSOC)* (pp. 179–190).

Hashemian, S. V. & Mavaddat, F. (2005). A graph-based approach to web services composition. In *The Symposium on Applications and the Internet* (pp. 183–189). Washington, DC, USA: IEEE Computer Society.

Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent Systems*, 16, 30–37.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C member submission, World Wide Web Consortium.

IEEE (1990). IEEE standard glossary of software engineering terminology. *IEEE Std 610.12-1990*.

Jaeger, M. C., Rojec-Goldmann, G., & Muhl, G. (2005). Qos aggregation in web service compositions. In *The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE)* (pp. 181–185).

Kazhamiakin, R., Benbernou, S., Baresi, L., Plebani, P., Uhlig, M., & Barais, O. (2010). Adaptation of service-based systems. In *Service Research Challenges and Solutions for the Future Internet*, volume 6500 of *Lecture Notes in Computer Science* (pp. 117–156). Springer.

Krill, P. (2005). Microsoft, IBM, SAP discontinue UDDI registry effort. Website. http://www.infoworld.com/d/architecture/microsoft-ibm-sap-discontinue-uddi-registry-effort-777.

Kuropka, D., Tröger, P., Staab, S., & Weske, M., Eds. (2008). *Semantic Service Provisioning*. Berlin: Springer.

Küster, U. & König-Ries, B. (2008). Towards standard test collections for the empirical evaluation of semantic web service approaches. *International Journal of Semantic Computing*, 2(3), 381–402.

Lausen, H. & Haselwanter, T. (2007). Finding web services. In *In Proceedings of the 1st European Semantic Technology Conference (ESTC)*.

Li, Y., Zhang, X., Yin, Y., & Lu, Y. (2011). Towards functional dynamic reconfiguration for service-based applications. In *IEEE World Congress on Services (SERVICES)* (pp. 467–473).

Lin, K.-J., Zhang, J., & Zhai, Y. (2009). An efficient approach for service process reconfiguration in SOA with end-to-end QoS constraints. In *IEEE Conference on Commerce and Enterprise Computing (CEC)* (pp. 146–153). Washington, DC, USA: IEEE Computer Society.

Lin, K.-J., Zhang, J., Zhai, Y., & Xu, B. (2010). The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA. *Service Oriented Computing and Applications (SOCA)*, 4(3), 157–168.

Liotta, A., Yew, A., Bohoris, C., & Pavlou, G. (2002). Delivering service adaptation with 3g technology. In M. Feridun, P. Kropf, & G. Babin (Eds.), *Management Technologies for E-Commerce and E-Business Applications*, volume 2506 of *Lecture Notes in Computer Science* (pp. 108–120). Springer Berlin Heidelberg.

Liu, A., Li, Q., Huang, L., & Xiao, M. (2010). FACTS: A framework for fault-tolerant composition of transactional web services. *IEEE Transactions on Services Computing*, 3(1), 46–59.

Lyu, M. R., Ed. (1996). *Handbook of software reliability engineering*. McGraw-Hill, Inc.

Mahbub, K. & Zisman, A. (2009). Replacement policies for service-based systems. In *The International Conference on Service-Oriented Computing (ICSOC)*, ICSOC/ServiceWave'09 (pp. 345–357). Berlin, Heidelberg: Springer-Verlag.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. (2004). OWL-S: Semantic Markup for Web Services. Website. W3C Member Submission.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. (2006). OWL-S: Semantic markup for web services - Pre-Release 1.2.

Martin, D., Burstein, M., Mcdermott, D., McIlraith, S. A., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., & Srinivasan, N. (2007). Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3), 243–277.

McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, 16(2), 46–53.

Medjahed, B. (2004). *Semantic web enabled composition of web services*. PhD thesis, Virginia Polytechnic Institute and State University. Chair-Bouguettaya, Athman.

Möller, T. & Schuldt, H. (2010). OSIRIS Next: Flexible semantic failure handling for composite web service execution. In *Fourth International Conference on Semantic Computing (ICSC)* (pp. 212–217). Los Alamitos, CA, USA.

OPOSSum (2009). Online portal for semantic services.

OWLS-TC (2010). OWL-S service retrieval test collection. Available online at: http://www.semwebcentral.org/projects/owls-tc/.

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. P. (2002). Semantic matching of web services capabilities. In *First International Semantic Web Conference on The Semantic Web (ISWC)* (pp. 333–347). London, UK: Springer-Verlag.

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10), 46–52.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., & Fensel, D. (2005). Web Service Modeling Ontology. *Appl. Ontol.*, 1(1), 77–106.

Salas, J., Perez-Sorrosal, F., Patiño Martínez, M., & Jiménez-Peris, R. (2006). Ws-replication: a framework for highly available web services. In *Proceedings of the 15th international conference on World Wide Web (WWW)* (pp. 357–366).: ACM.

SAWSDL-TC (2010). SAWSDL service retrieval test collection. Available online at: http://www.semwebcentral.org/projects/sawsdl-tc/.

SAWSDL Working Group (2007). Semantic Annotations for WSDL and XML Schema. Website. http://www.w3.org/TR/sawsdl/.

Sayal, M., Breitbart, Y., Scheuermann, P., & Vingralek, R. (1998). Selection algorithms for replicated web servers. *SIGMETRICS Perform. Eval. Rev.*, 26(3), 44–50.

SEALS (2011). SEALS - Semantic Evaluation at Large Scale. Available online at: http://www.seals-project.eu/.

Seekda (2011). Web Services Search Engine @ seekda.com. Website. http://webservices.seekda.com/.

Steinmetz, N., Lausen, H., & Brunner, M. (2009). Web service search on large scale. In L. Baresi, C.-H. Chi, & J. Suzuki (Eds.), *Service-Oriented Computing*, volume 5900 of *Lecture Notes in Computer Science* (pp. 437–444). Springer Berlin Heidelberg.

Subramanian, S., Thiran, P., Narendra, N. C., Mostefaoui, G. K., & Maamar, Z. (2008). On the enhancement of BPEL engines for self-healing composite web services. In *International Symposium on Applications and the Internet* (pp. 33–39). Washington, DC, USA: IEEE Computer Society.

Taher, Y., Benslimane, D., Fauvet, M.-C., & Maamar, Z. (2006). Towards an approach for web services substitution. In *10th International Database Engineering and Applications Symposium, (IDEAS)* (pp. 166–173).

Tartanoglu, F., Issarny, V., Romanovsky, A., & Levy, N. (2003). Dependability in the web services architecture. In *Architecting Dependable Systems*, volume 2677 of *Lecture Notes in Computer Science* (pp. 90–109). Springer Berlin / Heidelberg.

Vaculín, R. & Sycara, K. (2008). Semantic web services monitoring: An OWL-S based approach. In *41st Annual Hawaii International Conference on System Sciences (HICSS)* (pp. 313–322). Washington, DC, USA: IEEE Computer Society.

Vaculín, R., Wiesner, K., & Sycara, K. (2008). Exception handling and recovery of semantic web services. In *Fourth International Conference on Networking and Services (ICNS)* (pp. 217–222).

Van Riemsdijk, M. B. & Wirsing, M. (2007). Using goals for flexible service orchestration: a first step. In *AAMAS international workshop on service-oriented computing: agents, semantics, and engineering*, AAMAS (pp. 31–48).: Springer.

W3C Working Group (2004). Web Services Glossary. Website. http://www.w3.org/TR/ws-gloss/.

Wiesner, K., Vaculín, R., Kollingbaum, M., & Sycara, K. (2008). Recovery mechanisms for semantic web services. In *The 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)*, volume 5053 of *Lecture Notes in Computer Science* (pp. 100–105).: Springer.

Wu, D., Parsia, B., Sirin, E., Hendler, J., & Nau, D. (2003). Automating DAML-S Web Services Composition Using SHOP2. In D. Fensel, K. Sycara, & J. Mylopoulos (Eds.), *The Semantic Web - International Semantic Web Conference (ISWC)*, volume 2870 of *Lecture Notes in Computer Science* (pp. 195–210). Springer.

Yan, Y., Poizat, P., & Zhao, L. (2010). Repair vs. recomposition for broken service compositions. In *8th International Conferenece on Service-Oriented Computing (ICSOC)*, volume 6470 of *Lecture Notes in Computer Science* (pp. 152–166). San Francisco, CA, USA: Springer Berlin / Heidelberg.

Yu, Q. & Bouguettaya, A. (2008). Framework for web service query algebra and optimization. *ACM Transactions on the Web*, 2, 1–35.

Yu, T. & Lin, K.-J. (2005a). Adaptive algorithms for finding replacement services in autonomic distributed business processes. In *The 7th International Symposium on Autonomous Decentralized Systems (ISADS)* (pp. 427–434).

Yu, T. & Lin, K.-J. (2005b). Service selection algorithms for Web services with end-to-end QoS constraints. *Information Systems and E-Business Management*, 3, 103–126.

Yuan Yi, C. (2005). Team-oriented model for composite web services failure recovery. Honours Programme of the School of Computer Science and Software Engineering, The University of Western Australia.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., & Sheng, Q. Z. (2003). Quality driven web services composition. In *12th International Conference on World Wide Web (WWW)* (pp. 411–421). New York, NY, USA: ACM.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5), 311–327.

Zhai, Y., Zhang, J., & Lin, K.-J. (2009). SOA middleware support for service process reconfiguration with end-to-end QoS constraints. In *IEEE International Conference on Web Services (ICWS)* (pp. 815–822).

Zheng, Z. & Lyu, M. R. (2010). An adaptive QoS-aware fault tolerance strategy for Web services. *Empirical Software Engineering*, 15, 323–345.

Appendix A: OWLS-TC Characteristics

OWL-S Service Retrieval Test Collection (OWLS-TC)is an OWL-S service retrieval test collection (OWLS-TC, 2010). The services are all described in OWL-S version 1.1 and some in OWL-S version 1.0 as well.

This test collection contains service advertisements, sample requests, and relevance judgments of the advertisements regarding to the requests. Klusch and Kapahnke et al. developed it in 2005 to support the evaluation of the performance of OWL-S semantic Web service matchmaking algorithms. The developers actively improved OWLS-TC. For example, regarding the world-altering category of services, there was no service, including precondition and result (effect), in the first three versions of this test collection. Some informal conditions written in services' comments could not convey practical world-altering service definitions.

They have released the fourth version of this collection in September 2010. The latest release has service condition and result (effect) specifications available both in SWRL (Horrocks et al., 2004) and PDDL (Ghallab et al., 1998). The test collection is bundled separately based on the mentioned languages which are used to specify the services' precondition and result.

A.1 OWLS-TC Details

OWLS-TC Version 4 contains 1083 atomic semantic Web services. They are described using their Input and Output, and Precondition and Results (IOPRs).

We extracted the properties of atomic Web services from OWLS-TC. The properties include the service names (from human-readable information in ServiceProfile) along with their functional properties (Martin et al., 2004). The extraction was performed using

OWL-S API[1], a Java library for programmatic access to OWL-S service descriptions. Every service has several inputs and outputs which are specified as Uniform Resource Identifiers (URIs) of the concepts in multiple defined ontologies. Moreover, some services have precondition and result which are identified as other URIs as well.

All the 1083 services in OWLS-TC are listed in Table A.3. The table identifies service ID, the service name, its Filename (with .owls suffix as their filename extensions), number of inputs and outputs (in IO) column, number of preconditions and results (in PR) column, and service class ID. The (service) ID is a sequential number that we assigned for the services. The service class ID is another value which we assigned for the service classes, and is described in Section A.2.

Among 1083 services in OWLS-TC, 1022 have at least one input, 1057 have at least one output, and 996 have both. The average number of inputs and outputs are 1.42 and 1.49 respectively. The services have different number of inputs and outputs. The frequency of the services based on their inputs and outputs are shown in Table A.1. In the table every cell identifies the number of services with a particular input and output count. For example, there are 286 services with one input concept and two output concepts.

Among all, 120 of services have either precondition or result, 119 have precondition, 22 have result, and 21 have both precondition and result. Hence, only 1 has a result but no precondition (Service ID 945 in Table A.3). Consequently, 22 world-altering services are described in this test collection. The frequency of the services based on their preconditions and results are shown in Table A.2. In the table every cell identifies the number of services with a particular precondition and result count. For example, there are 963

[1] http://on.cs.unibas.ch/owls-api/index.html

Table A.1.: Frequency of Services in OWLS-TC with a Particular Input and Output Count. The cells show the number of services with that particular IO count.

		Output Count							
		0	1	2	3	4	5	6	Grand Total
Input Count	0		45	12	3	1			61
	1	9	363	286	50	3		2	713
	2	8	161	37	6	4	2		218
	3	9	26	7	2	1			45
	4		11	5		1	1		18
	5		3	1	1	1		2	8
	6		5	2		2			9
	7		1		2				3
	8		3	1		1		1	6
	9		1						1
	11		1						1
	Grand Total	26	620	351	64	14	3	5	1083

Table A.2.: Frequency of Services in OWLS-TC with a Particular Precondition and Result Count. The cells show the number of services with that particular PR count.

		Result Count		
		0	1	Grand Total
Precondition Count	0	963	1	964
	1	98	21	119
	Grand Total	1061	22	1083

services with neither precondition nor result.

In total, 465 concepts, which are predefined in multiple ontologies, have been used in the test collection as inputs or outputs. In other words, the inputs and outputs of the services in this test collection are defined using 465 ontology concepts. In addition, there are 28 conditions used as precondition and result for the services. None of the conditions have been used both for the precondition and for the result of the services. Therefore, the atomic services in this test collection cannot be joined based on PR.

A.2 Service Functionality Matching in OWLS-TC

There are similarities based on the semantics of the services in OWLS-TC. These similarities are based on the IO concepts of the services. There is no PR similarity because none of the conditions has been used as both precondition and result.

For the IO similarity we used the exact match of the concepts in this research. That is to say, two services are exactly matched if their input and output concepts exactly match (Definition 7 on page 47).

In terms of IO match, for 246 of the services there is at least a matched service in the collection. These 246 services are classified in 99 service classes. The service classes are called "abstract services". The services in each service class is called a "concrete service" (Canfora et al., 2008). The service classes have various number of members (atomic services) as follows: 1 service class has 7 similar services, 1 service class has 5 services, 9 service classes have 4 services each, 22 service classes have 3 services each, and 66 service classes have 2 services each.

$$1*7+1*5+9*4+22*3+66*2=246 \quad \textit{Services in Service Classes} \tag{A.1}$$

837 services are not in any class, i.e. there are no available matches in the test collection.

$$246+837=1083 \quad \textit{Total Services in OWLS-TC} \tag{A.2}$$

Additionally, there are 217 pairs of exact IO matches among the services. In terms of Input and Output, and Result (IOR) matches which matches the services based on their *result* in addition to IO, number of pairs are 217 as well. In other words, all the matches

based on IO have either no result condition or similar result conditions. However, going one step further for matching, IOPR, which matches all four functional properties of a semantic Web services, there are 206 pairs of match among the services in the collection. Table A.4 shows all 217 pairs of matches of the collection. In the table, IOR and IOPR matches are identified in the last column. Moreover, the service class ID of which the pair belongs to is distinguished as well. Table A.5 separately lists the 11 pairs of IOR matches which are not IOPR match, i.e. their preconditions do not match. Finally, Table A.6 shows all 99 service classes and their concrete services.

As a conclusion, for the approaches that use late binding (abstract services) and atomic replacement for failure recovery, for 837 out 1083 services there is no replacement and for only 246 services there is at least an IO matched atomic replacement.

Table A.3.: Services in OWLS-TC

ID	Name	Filename (.owls)	IO	PR	Class
0	4WheeledCar1PersonBicyclePrice	1personbicycle4wheeledcar_price_service	2,1	0,0	0
1	Kohl Car1PersonBicyclePrice	1personbicyclecar_price_Kohlservice	2,1	0,0	1
2	Car1PersonBicyclePrice	1personbicyclecar_price_service	2,1	0,0	1
3	Car1PersonBicyclePrice	1personbicyclecar_price_TheBestservice	2,1	0,0	1
4	Bicycle4Wheeledcar_Price_service	2personbicycle4wheeledcar_price_service	2,1	0,0	4
5	3wheeledcar year price	3wheeledcaryear_price_service	2,1	0,0	
6	3wheeledcar year recommended price	3wheeledcaryear_recommendedprice_service	2,1	0,0	
7	3Wheeled Car price	3wheeledcar_price_service	1,1	0,0	
8	4WheeledCar 1PersonBicycle MaxPrice	4wheeledcar1personbicycle_maxprice_service	2,1	0,0	
9	4WheeledCar 1PersonBicycle Price	4wheeledcar1personbicycle_price_service	2,1	0,0	0
10	4WheeledCar 2PersonBicycle MaxPrice	4wheeledcar2personbicycle_maxprice_service	2,1	0,0	
11	4WheeledCar 2PersonBicyclePrice	4wheeledcar2personbicycle_price_service	2,1	0,0	4
12	4WheeledCarBicyclePrice	4wheeledcarbicycle_price_service	2,1	0,0	12
13	4wheeledcar year price report	4wheeledcaryear_pricereport_service	2,2	0,0	
14	4wheeledcar year price	4wheeledcaryear_price_service	2,1	0,0	
15	4WheeledCar price	4wheeledcar_price_service	1,1	0,0	
16	4wheeledCarTechnology	4wheeledcar_technology_service	1,1	0,0	
17	4WheeledCar Year Price	4wheeledcar_yearprice_service	1,2	0,0	
18	GovernmentOrganization Academic Degree Scholarship	academic-degreegovernmentorganization_funding_service	2,1	0,0	
19	Academic Degree GovernmentOrganization Lending	academic-degreegovernmentorganization_lending_service	2,1	0,0	
20	GovernmentOrganization Academic Degree Unilateral Giving	academic-degreegovernmentorganization_unilateralgiving_service	2,1	0,0	
21	Government Academic Degree Scholarship	academic-degreegovernment_funding_service	2,1	0,0	
22	Academic Degree Government Lending	academic-degreegovernment_lending_service	2,1	0,0	
23	GovernmentAcademicDegreeScholarshipService	academic-degreegovernment_scholarship_service	2,1	0,0	
24	Government Academic Degree Unilateral Giving	academic-degreegovernment_unilateralgiving_service	2,1	0,0	
25	GermanGovernment Academic-Degree Funding Duration	academic-degree_fundingduration_GermanGovservice	1,2	0,0	
26	Academic-Degree Funding GermangovService	academic-degree_funding_GermanGovservice	1,1	0,0	
27	GermanGovernment Academic-Degree Lending Duration	academic-degree_lendingduration_GermanGovservice	1,2	0,0	
28	Academic-Degree Lending GermanGovernment	academic-degree_lending_GermanGovservice	1,1	0,0	
29	GermanGovernment Academic-Degree Scholarship Duration	academic-degree_scholarshipduration_GermanGovservice	1,2	0,0	
30	Academic-Degree Scholarship GermangovService	academic-degree_scholarship_GermanGovservice	1,1	0,0	
31	Academic-Number Book Author booksearch	academic-item-number_bookauthor_service	1,2	0,0	
32	Academic Book Number booksearch	academic-item-number_book_service	1,1	0,0	32
33	Academic-Number Publication Author	academic-item-number_publicationauthor_service	1,2	0,0	
34	Academic-Number Publication	academic-item-number_publication_service	1,1	0,0	
35	AcademicBookNumberOrISBNSearch	AcademicBookNumberOrISBNSearch	1,1	0,0	35
36	AcademicBookNumberSearch	AcademicBookNumberSearch	1,1	0,0	32
37	Academic Address	academic_address_service	1,1	0,0	
38	Academic Postal-Address	academic_postal-address_service	1,1	0,0	
39	AcceptCostAndHealingPlan	AcceptCostAndHealingPlan_service	1,1	0,0	
40	Activity Beach	activity_beach_service	1,1	0,0	
41	Activity City	activity_city_service	1,1	0,0	
42	Activity Destination	activity_destination_service	1,1	0,0	
43	Activity FamilyDestination	activity_familydestination_service	1,1	0,0	
44	Activity Farmland	activity_farmland_service	1,1	0,0	
45	Activity Nationalpark	activity_nationalpark_service	1,1	0,0	
46	Activity RuralArea	activity_ruralarea_service	1,1	0,0	
47	Activity Town	activity_town_service	1,1	0,0	
48	Activity UrbanArea	activity_urbanarea_service	1,1	0,0	
49	TranslocationService	AddLinks__service	2,1	1,1	
50	StrikeIron Address Distance Calculator between two addresses.	addressDistanceCalculator	6,1	1,0	
51	Address location Geocoder service.	addressGeocoder	1,3	0,0	
52	Adventure RuralArea	adventure_ruralarea_service	1,1	0,0	
53	Adventure UrbanArea	adventure_urbanarea_service	1,1	0,0	
54	Move Agent	agent_movement	2,1	0,0	
55	MianMarkt ShoppingService	agent_price_MianMarktservice	1,1	0,0	
56	Car Price	amount-of-money3wheeledcar_price_service	2,1	0,0	
57	Threewheeled Car Recommended Price	amount-of-money3wheeledcar_recommendedprice_service	2,1	0,0	
58	Car Price	amount-of-money4wheeledcar_price_service	2,1	0,0	
59	Car Price	amount-of-money4wheeledcar_recommendedprice_service	2,1	0,0	
60	Car Price	amount-of-moneycar_pricecompany_service	2,2	0,0	
61	Car Price	amount-of-moneycar_price_service	2,1	0,0	
62	Car Price	amount-of-moneycheapcar_price_service	2,1	0,0	
63	Cheap Car Recommended Price	amount-of-moneycheapcar_recommendedprice_service	2,1	0,0	
64	Car Price	amount-of-moneyexpensivecar_price_service	2,1	0,0	
65	Expensive Car Recommended Price	amount-of-moneyexpensivecar_recommendedprice_service	2,1	0,0	
66	GetDrugInformation	Apothecary_service	1,1	0,0	
67	ApplePriceService	apple_price_service	1,1	0,0	
68	AuthorizePhysician	AuthorizePhysician_service	3,3	0,0	
69	Author Book MaxPrice	author_bookmaxprice_service	1,2	0,0	
70	Ziku BookFinderPriceService	author_bookprice_service	1,2	0,0	
71	Author Book RecommendedPrice	author_bookrecommendedprice_service	1,2	0,0	
72	Author Book TaxedPrice	author_booktaxedprice_service	1,2	0,0	
73	Author Book TaxFreePrice	author_booktaxfreeprice_service	1,2	0,0	
74	Author Monograph MaxPrice	author_monographmaxprice_service	1,2	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
75	Author Monograph Price	author_monographprice_service	1,2	0,0	
76	Author Monograph RecommendedPrice	author_monographrecommendedprice_service	1,2	0,0	
77	Author Monograph TaxedPrice	author_monographtaxedprice_service	1,2	0,0	
78	Author Monograph TaxFreePrice	author_monographtaxfreeprice_service	1,2	0,0	
79	Author Novel MaxPrice	author_novelmaxprice_service	1,2	0,0	
80	Author Novel Price	author_novelprice_service	1,2	0,0	
81	Author Novel RecommendedPrice	author_novelrecommendedprice_service	1,2	0,0	
82	Author Novel TaxedPrice	author_noveltaxedprice_service	1,2	0,0	
83	Author Novel TaxFreePrice	author_noveltaxfreeprice_service	1,2	0,0	
84	Author Publication MaxPrice	author_publicationmaxprice_service	1,2	0,0	
85	Author Publication Price	author_publicationprice_service	1,2	0,0	
86	Author Publication RecommendedPrice	author_publicationrecommendedprice_service	1,2	0,0	
87	Author Publication TaxedPrice	author_publicationtaxedprice_service	1,2	0,0	
88	Author Publication TaxFreePrice	author_publicationtaxfreeprice_service	1,2	0,0	
89	Author ScienceFictionBook MaxPrice	author_sciencefictionbookmaxprice_service	1,2	0,0	
90	Author ScienceFictionBook Price	author_sciencefictionbookprice_service	1,2	0,0	
91	Author ScienceFictionBook RecommendedPrice	author_sciencefictionbookrecommendedprice_service	1,2	0,0	
92	Author ScienceFictionBook TaxedPrice	author_sciencefictionbooktaxedprice_service	1,2	0,0	
93	Author ScienceFictionBook TaxFreePrice	author_sciencefictionbooktaxfreeprice_service	1,2	0,0	
94	Auto1PersonBicyclePrice	auto1personbicycle_price_service	2,1	0,0	
95	Auto 2PersonBicycle MaxPrice	auto2personbicycle_maxprice_service	2,1	0,0	
96	Auto 2PersonBicycle Price	auto2personbicycle_price_service	2,1	0,0	
97	Auto 2PersonBicycle RecommendedPrice	auto2personbicycle_recommendedprice_service	2,1	0,0	
98	Auto 2PersonBicycle TaxedPrice	auto2personbicycle_taxedprice_service	2,1	0,0	
99	Auto Bicycle MaxPrice	autobicycle_maxprice_service	2,1	0,0	
100	Auto Bicycle Price	autobicycle_price_service	2,1	0,0	100
101	Auto Bicycle RecommendedPrice	autobicycle_recommendedprice_service	2,1	0,0	
102	Auto Bicycle TaxedPrice	autobicycle_taxedprice_service	2,1	0,0	
103	Auto Bicycle MaxPrice	autocycle_maxprice_service	2,1	0,0	
104	Auto Cycle Price	autocycle_price_service	2,1	0,0	
105	Auto Bicycle RecommendedPrice	autocycle_recommendedprice_service	2,1	0,0	
106	Auto Bicycle TaxedPrice	autocycle_taxedprice_service	2,1	0,0	
107	Auto Price Color	auto_pricecolor_service	1,2	0,0	
108	AutoPrice	auto_price_service	1,1	0,0	
109	Auto RecommendedPrice Color	auto_recommendedpricecolor_service	1,2	0,0	
110	AutoTechnology	auto_technology_service	1,1	0,0	
111	car price	auto_yearprice_service	1,2	0,0	
112	AvailablePreparedFoodService	Available_preparedfoodquantity_service	1,2	0,0	112
113	Avocado Price	avocado_price_service	1,1	0,0	
114	Government Academic Degree Scholarship	awardgovernment_funding_service	2,1	0,0	
115	GermanGovernment Award Funding Duration	award_fundingduration_GermanGovservice	1,2	0,0	
116	Award Funding Germangoverment	award_funding_GermanGovservice	1,1	0,0	
117	GermanGovernment Award Lending Duration	award_lendingduration_GermanGovservice	1,2	0,0	
118	Award Lending Germangoverment	award_lending_GermanGovservice	1,1	0,0	
119	GermanGovernmentAwardScholarshipService	award_scholarshipduration_GermanGovservice	1,2	0,0	119
120	GermanGovernmentAwardScholarshipService	award_scholarshipduration_SwissGovservice	1,2	0,0	119
121	GermanGovernmentAwardScholarshipService	award_scholarship_GermanGovservice	1,1	0,0	
122	GovernmentOrganization BallisticMissile Financing Range	ballisticmissilegovernmentorganization_financingrange_service	2,2	0,0	
123	GovernmentOrganization BallisticMissile Funding Range	ballisticmissilegovernmentorganization_fundingrange_service	2,2	0,0	
124	GovernmentOrganization BallisticMissile Giving Range	ballisticmissilegovernmentorganization_givingrange_service	2,2	0,0	
125	GovernmentOrganization BallisticMissile Lending Range	ballisticmissilegovernmentorganization_lendingrange_service	2,2	0,0	
126	Government BallisticMissile Financing	ballisticmissilegovernment_financingrange_service	2,2	0,0	
127	GovBllisticMissileFundingService	ballisticmissilegovernment_fundingrange_service	2,2	0,0	
128	Government BallisticMissile Giving	ballisticmissilegovernment_givingrange_service	2,2	0,0	
129	Government BallisticMissile Lending	ballisticmissilegovernment_lendingrange_service	2,2	0,0	
130	Ben	Ben_service	1,1	0,0	
131	Beverage MaxPrice Physical-Quantity	beverage_maxpricephysical-quantity_service	1,2	0,0	
132	Beverage MaxPrice Quantity	beverage_maxpricequantity_service	1,2	0,0	
133	Beverage Price Physical-Quantity	beverage_pricephysical-quantity_Aldiservice	1,2	0,0	
134	Beverage Price Quantity	beverage_pricequantity_Aldiservice	1,2	0,0	
135	Beverage TaxedPrice Physical-Quantity	beverage_taxedpricephysical-quantity_service	1,2	0,0	
136	Beverage TaxedPrice Quantity	beverage_taxedpricequantity_service	1,2	0,0	
137	Beverage TaxFreePrice Physical-Quantity	beverage_taxfreepricephysical-quantity_service	1,2	0,0	
138	Beverage TaxFreePrice Quantity	beverage_taxfreepricequantity_service	1,2	0,0	
139	Bicycle4Wheeledcar_Price_service	bicycle4wheeledcar_price_service	2,1	0,0	12
140	BicycleAuto_Price_service	bicycleauto_price_service	2,1	0,0	100
141	Car1PersonBicyclePrice Year	bicyclecar_priceyear_service	2,2	0,0	
142	BicycleCar_Price_service	bicyclecar_price_service	2,1	0,0	142
143	BookFinder	BookFinder	1,1	0,0	143
144	BookMedicalFlight	BookMedicalFlight_service	2,3	0,0	
145	BookMedicalTransport	BookMedicalTransport_service	3,2	0,0	
146	BookNonMedicalFlight	BookNonMedicalFlight_service	2,3	0,0	
147	BookNonMedicalTransport	BookNonMedicalTransport_service	2,2	0,0	
148	Bea Book Shopping	bookpersoncreditaccount__Beaservice	3,1	1,1	148
149	Book Shopping	bookpersoncreditaccount__service	3,1	1,1	148
150	AuthorisedPersonBookPrice	bookpersoncreditcardaccount_price_service	3,1	1,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
151	AuthorisedPersonBookRecommendedPrice	bookpersoncreditcardaccount_recommendedprice_service	3,1	1,0	
152	AuthorisedPersonBookPrice	bookpersoncreditcardaccount_taxedfreeprice_service	3,1	1,0	
153	BShop book shopping	bookpersoncreditcardaccount__BShopservice	3,1	1,1	153
154	Book Shopping	bookpersoncreditcardaccount__service	3,1	1,1	153
155	BookPrice	bookpersonOptional_price_service	2,1	0,0	155
156	BookPrice	bookperson_price_service	2,1	1,0	155
157	Person Book	bookperson__service	2,1	1,1	
158	BookPrice	BookPrice	1,1	0,0	158
159	BookSearch	BookSearchService	1,1	0,0	143
160	Book Shopping	bookusercreditcardaccount__service	3,1	1,1	
161	BAT	book_authorbook-type_service	1,2	0,0	
162	BookAuthorPriceService	book_authorprice_Novelservice	1,2	0,0	162
163	BookAuthorPriceService	book_authorprice_service	1,2	1,0	162
164	BookAuthorService	book_authortext_service	1,2	0,0	
165	BookAuthorService	book_author_EncSSservice	1,1	0,0	165
166	BookAuthorService	book_author_service	1,1	0,0	165
167	Cheapest Book	book_Cheapestprice_service	1,1	0,0	158
168	BookPublisherService	book_person_Publisherservice	1,1	0,0	
169	BookPriceService	book_pricereviewbook_service	1,3	0,0	
170	BookPriceTypeSizeService	book_pricesizebook-type_service	1,3	0,0	
171	BookPriceService	book_price_service	1,1	0,0	158
172	BookPublisherService	book_publisher_service	1,1	0,0	
173	BookRRPService	book_readerreviewperson_service	1,3	0,0	
174	BookReaderReviewService	book_readerreview_service	1,2	0,0	
175	BookRecommendedPriceService	book_recommendedpriceindollar_service	1,1	0,0	175
176	BookRecommendedPriceService	book_recommendedprice_RegisteredUserservice	1,1	1,0	176
177	BookRecommendedPriceService	book_recommendedprice_service	1,1	0,0	176
178	BookPriceService	book_reviewprice_service	1,2	0,0	
179	BookPriceTaxedPriceService	book_taxedpriceprice_service	1,2	0,0	
180	BookTaxedPriceService	book_taxedprice_service	1,1	0,0	
181	BookCartService	book__ShoppingCartservice	1,1	1,1	
182	BreadBiscuitPriceService	breadorbiscuit_recPricetaxedpriceineuro_service	1,2	0,0	
183	Butter MaxPrice	butter_maxprice_service	1,1	0,0	
184	Butter RecommendedPrice	butter_recommendedprice_service	1,1	0,0	
185	Butter TaxedPrice	butter_taxedprice_service	1,1	0,0	
186	CAD	CAD_medical_service	1,2	0,0	
187	Distance calculator between two locations	calculateDistanceInMiles	4,1	1,0	187
188	Distance calculator between two locations	calculateDistanceUsingSphericalGeometry	4,1	1,0	187
189	Sunrise Time Calcutator Service.	calculateSunriseTime	4,1	1,0	
190	Distance calculator between two locations	calculatorDistanceSphericalLawOfCosines	4,1	1,0	187
191	Car2PersonBicyclePrice	car2personbicycle_price_service	2,1	0,0	
192	CarBicyclePrice	carbicycle_price_service	2,1	0,0	142
193	CarBicycle Recommended Price	carbicycle_recommendedprice_service	2,1	0,0	
194	Car Bicycle Taxed Price	carbicycle_taxedprice_service	2,1	0,0	
195	CarCyclePrice	carcycle_price_service	2,1	0,0	
196	CareOrganization Biopsy Availability	careorganization_biopsy_service	1,1	0,0	
197	CareOrganization DiagnosticProcess TimeDuration	careorganization_diagnosticprocesstimeduration_service	1,2	0,0	
198	CareOrganization DiagnosticProcess TimeInterval	careorganization_diagnosticprocesstimeinterval_service	1,2	0,0	
199	CareOrganization DiagnosticProcess TimeMeasure	careorganization_diagnosticprocesstimemeasure_service	1,2	0,0	
200	CareOrganization DiagnosticProcess	careorganization_diagnosticprocess_service	1,1	0,0	
201	CareCareOrganization Experiment	careorganization_experimenting_service	1,1	0,0	
202	CareOrganization Investigating	careorganization_investigating_service	1,1	0,0	
203	CareOrganization Predicting	careorganization_predicting_service	1,1	0,0	
204	Recommended price of car model	caryear_recommendedpriceineuro_service	2,1	0,0	
205	Car price	car_priceauto_service	1,2	0,0	
206	T-car price	car_pricecolor_service	1,2	0,0	
207	Car Price quality	car_pricequality_service	1,2	0,0	
208	car price report	car_pricereport_service	1,2	0,0	
209	car price	car_price_service	1,1	0,0	209
210	CarRecommendedPrice	car_recommendedpriceindollar_service	1,1	0,0	
211	car Recommended price	car_recommendedpriceineuro_service	1,1	0,0	
212	CarRecommendedPrice	car_recommendedprice_service	1,1	0,0	
213	car report	car_report_service	1,1	0,0	
214	car price	car_taxedpriceprice_service	1,2	0,0	
215	Car TaxedPrice Report	car_taxedpricereport_service	1,2	0,0	
216	CarTechnology	car_technology_service	1,1	0,0	
217	Car Year Price	car_yearprice_service	1,2	0,0	
218	CCP	CCP_service	1,2	0,0	
219	2for 1 Price	cdplayermp3player_price_service	2,1	0,0	
220	CDPlayer MaxPrice	cdplayer_maxprice_service	1,1	0,0	
221	CD Player recommended Price	cdplayer_recommendedprice_service	1,1	0,0	
222	CDPlayer TaxedPrice	cdplayer_taxedprice_service	1,1	0,0	
223	CheapCar 1PersonBicycle MaxPrice	cheapcar1personbicycle_maxprice_service	2,1	0,0	
224	CheapCar 1PersonBicycle Price	cheapcar1personbicycle_price_service	2,1	0,0	
225	CheapCar 2PersonBicycle MaxPrice	cheapcar2personbicycle_maxprice_service	2,1	0,0	
226	CheapCar 2PersonBicyclePrice	cheapcar2personbicycle_price_service	2,1	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
227	CheapCar Price Color	cheapcar_pricecolor_service	1,2	0,0	
228	CheapCar Price Report	cheapcar_pricereport_service	1,2	0,0	
229	Cheap car price	cheapcar_price_service	1,1	0,0	
230	CheapCar RecommendedPrice Color	cheapcar_recommendedpricecolor_service	1,2	0,0	
231	CheapCar TaxedPrice Report	cheapcar_taxedpricereport_service	1,2	0,0	
232	CheapCarTechnology	cheapcar_technology_service	1,1	0,0	
233	CheapCar Year Price	cheapcar_yearprice_service	1,2	0,0	
234	Check and Lookup address	checkAndLookupAddress	5,6	1,0	
235	CheckCostAndHealingPlan	CheckCostAndHealingPlan_service	2,1	0,0	
236	CheckEquipmentAvailability	CheckEquipmentAvailability_service	4,1	0,0	
237	CheckHospitalAvailability	CheckHospitalAvailability_service	3,1	0,0	237
238	CheckPersonnelAvailability	CheckPersonnelAvailability_service	3,1	0,0	237
239	CheckRoomAvailability	CheckRoomAvailability_service	3,1	0,0	237
240	citycity route finder	citycity_arrowfigure_service	2,1	1,0	
241	City2CityRouteFinderService	citycity_map_service	2,1	1,0	
242	HotelReserveService	citycountryduration__HotelReserveservice	3,1	1,1	
243	AccomodationInfoService	citycountry_accommodation_service	2,1	1,0	
244	CityCountaryInfoService	citycountry_destinationhotel_service	2,2	1,0	
245	HotelInfoService	citycountry_hotel_service	2,1	1,0	245
246	SkilledPositionsService	citycountry_skilledoccupation_service	2,1	1,0	
247	CityGovernement Lending	citygovernment_lending_service	1,1	0,0	
248	City Accommodation	city_accommodation_service	1,1	0,0	
249	City BedAndBreakfast	city_bedandbreakfast_service	1,1	0,0	
250	GermanCityHotelInfoService	city_hotel_Germanservice	1,1	1,0	250
251	CityHotelInfoService	city_hotel_Saarlandservice	1,1	1,0	
252	CityHotelInfoService	city_hotel_service	1,1	0,0	250
253	City LuxuryHotel	city_luxuryhotel_service	1,1	0,0	
254	SkilledPositionsService	city_skilledoccupation_service	1,1	0,0	
255	Zip codes finder service.	city_state_ZipCodes	3,1	1,0	255
256	CityWeatherfrontService	city_weatherfront_service	1,1	0,0	
257	City WeatherSeason	city_weatherseason_service	1,1	0,0	
258	City WeatherSystem	city_weathersystem_service	1,1	0,0	
259	Close door	close_door	1,1	0,0	259
260	Coconut Price	coconut_price_service	1,1	0,0	
261	BreakFastPriceService	coffeesandwich_price_service	2,1	0,0	
262	Coffee MaxPrice	coffee_maxprice_service	1,1	0,0	
263	Coffee RecommendedPrice	coffee_recommendedprice_service	1,1	0,0	
264	Coffee TaxedPrice	coffee_taxedprice_service	1,1	0,0	
265	CC	companycountry_skilledoccupation_service	2,1	0,0	
266	Company Profession	company_profession_service	1,1	0,0	
267	SkilledPositionsSearch	company_skilledoccupation_service	1,1	0,0	
268	ContactEMA	ContactEMA_services	2,1	0,0	
269	AppleProducerService	corporation_apple_service	1,1	0,0	
270	CapitalCityCountaryHotelInfoService	countrycapital-city_hotel_service	2,1	1,0	
271	HotelInfoService	countrycity_hotel_service	2,1	1,0	245
272	GelLuxuryHotelInfoService	countrycity_luxuryhotel_Gelservice	2,1	1,0	272
273	LuxuryHotelInfoService	countrycity_luxuryhotel_service	2,1	1,0	272
274	CityCountryInfoService	countrycity_sportshotel_service	2,2	1,0	
275	HotelInfoService	countryvillage_hotel_service	2,1	1,0	
276	Country Company OccupationalTrade	country_companyoccupationaltrade_service	1,2	0,0	
277	Country Company Profession	country_companyprofession_service	1,2	0,0	
278	OccupationFinder	country_companyskilledoccupation_service	1,2	0,0	278
279	Country Corporation OccupationalTrade	country_corporationoccupationaltrade_service	1,2	0,0	
280	Country Corporation Profession	country_corporationprofession_service	1,2	0,0	
281	OccupationFinder	country_corporationskilledoccupation_service	1,2	0,0	281
282	CDeaconService	country_deacon_service	1,1	0,0	
283	Country Drought	country_drought_service	1,1	0,0	
284	CountryHotelService	country_hotel_service	1,1	0,0	
285	Country Lightning	country_lightning_service	1,1	0,0	
286	CountaryMap	country_map_service	1,1	0,0	
287	Country OccupationalTrade FullTimePosition	country_occupationaltradefulltimeposition_service	1,2	0,0	
288	Country OccupationalTrade PartTimePosition	country_occupationaltradeparttimeposition_service	1,2	0,0	
289	Country OccupationalTrade TimeDuration	country_occupationaltradetimeduration_Service	1,2	0,0	
290	Country OccupationalTrade TimeMeasure	country_occupationaltradetimemeasure_Service	1,2	0,0	
291	Country OccupationalTrade	country_occupationaltrade_service	1,1	0,0	
292	Country Organization OccupationalTrade	country_organizationoccupationaltrade_service	1,2	0,0	
293	Country Organization Profession	country_organizationprofession_service	1,2	0,0	
294	OccupationFinder	country_organizationskilledoccupation_service	1,2	0,0	294
295	Country Profession FullTimePosition	country_professionfulltimeposition_service	1,2	0,0	
296	ProfessionIncomeTax	country_professionincometax_service	1,2	0,0	
297	Country Profession PartTimePosition	country_professionparttimeposition_service	1,2	0,0	
298	Country Profession TimeDuration	country_professiontimeduration_Service	1,2	0,0	
299	Country Profession TimeMeasure	country_professiontimemeasure_service	1,2	0,0	
300	ProfessionOffer	country_profession_service	1,1	0,0	
301	Country SkilledOccupation FullTimePosition	country_skilledoccupationfulltimeposition_service	1,2	0,0	
302	Country SkilledOccupation PartTimePosition	country_skilledoccupationparttimeposition_JobService	1,2	0,0	302

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
303	Country skilled and partTimePositionsService	country_skilledoccupationparttimeposition_service	1,2	0,0	302
304	countryJobs	country_skilledoccupationtimeduration_JobService	1,2	0,0	304
305	countryJobs	country_skilledoccupationtimeduration_service	1,2	0,0	304
306	Country SkilledOccupation TimeMeasure	country_skilledoccupationtimemeasure_service	1,2	0,0	
307	CountrySkilledPositionsService	country_skilledoccupation_jobsservice	1,1	0,0	307
308	CountrySOccupService	country_skilledoccupation_service	1,1	0,0	307
309	SportsPositionsService	country_sportsposition_service	1,1	0,0	
310	Country WarmFront	country_warmfront_service	1,1	0,0	
311	Country WeatherFront	country_weatherfront_service	1,1	0,0	
312	CountryWeatherProcessService	country_weatherprocess_service	1,1	0,0	
313	Country WeatherSeason	country_weatherseason_service	1,1	0,0	
314	Country WeatherSystem	country_weathersystem_service	1,1	0,0	
315	2in1cycleservice	cycle1personbicycle_price_service	2,1	0,0	
316	CarCyclePrice	cyclecar_pricetaxedprice_service	2,2	0,0	
317	CarCycle Recommended Price	cyclecar_recommendedpriceineuro_service	2,1	0,0	
318	Datefruit Price	datefruit_price_service	1,1	0,0	
319	GovernementOrganization ScholarshipService	degreegovernmentorganization_scholarship_service	2,1	0,0	
320	GovernmentDegreeFundingService	degreegovernment_funding_service	2,1	0,0	
321	GovernmengLendingForDegreeService	degreegovernment_lending_service	2,1	0,0	
322	Government Degree Scholarship	degreegovernment_scholarship_service	2,1	0,0	322
323	GovernmentDegreeSpecialOffersService	degreegovernment_unilateralgiving_service	2,1	0,0	
324	NationalGovernmentDegreeScholarshipService	degreenationalgovernment_scholarship_service	2,1	0,0	
325	GermanGovernment Degree Funding Duration	degree_fundingduration_GermanGovservice	1,2	0,0	
326	Degree Funding GermangovService	degree_funding_GermanGovservice	1,1	0,0	
327	GermanGovernment Degree Lending Duration	degree_lendingduration_GermanGovservice	1,2	0,0	
328	Degree Lending Germangov	degree_lending_GermanGovservice	1,1	0,0	
329	GermanGovernment Degree Scholarship Duration	degree_scholarshipduration_GermanGovservice	1,2	0,0	
330	Degree Scholarship GermangovService	degree_scholarship_GermanGovservice	1,1	0,0	
331	DeoSFNPrice	DeoSFN_price_service	2,1	0,0	331
332	DJob	DJob_service	1,2	1,0	304
333	DrugStoreTeaService	drugstore_tea_service	1,1	0,0	
334	HotelInfoService	durationcountrycity_hotel_service	3,1	1,0	
335	Duration Geopolitical-Entity City Accommodation InfoService	durationgeopolitical-entitycity_accommodation_service	3,1	1,0	
336	Duration Geopolitical-Entity City BedAndBreakfast InfoService	durationgeopolitical-entitycity_bedandbreakfast_service	3,1	1,0	
337	Duration Geopolitical-Entity City Hotel InfoService	durationgeopolitical-entitycity_hotel_service	3,1	1,0	
338	2for 1 Price	dvdplayermp3player_pricemessage_service	2,2	0,0	
339	MD 2For 1 Price	dvdplayermp3player_price_MDservice	2,1	0,0	339
340	2For 1 Price	dvdplayermp3player_price_Rservice	2,1	0,0	339
341	2For 1 Price	dvdplayermp3player_price_service	2,1	0,0	339
342	2For 1 Price	dvdplayermp3player_RecpriceEuro_service	2,1	0,0	
343	2for 1 Price	dvdplayermp3player_Recpricemessage_service	2,2	0,0	
344	2For 1 Price	dvdplayermp3player_Recprice_service	2,1	0,0	344
345	DVDPlayer MaxPrice	dvdplayer_maxprice_service	1,1	0,0	
346	DVDPlayer TaxedPrice	dvdplayer_taxedprice_service	1,1	0,0	
347	EBookOrder	EBookOrder1	2,1	1,1	347
348	EBookOrder2	EBookOrder2	2,1	1,1	347
349	EBookOrder3	EBookOrder3	2,1	1,1	347
350	Educational-Employee Address	educational-employee_address_service	1,1	0,0	
351	Educational-Employee Postal-Address	educational-employee_postal-address_service	1,1	0,0	
352	EducationalOrganizationLecturer	educational-organization_lecturer-in-academia_service	1,1	0,0	
353	ElectricDevice Price	electricdevice_price_service	1,1	0,0	
354	SelectOtherHospital	EmergencyPhysician_service	1,1	0,0	
355	Employee Address	employee_address_service	1,1	0,0	
356	Employee postal address	employee_postal-address_service	1,1	0,0	356
357	Employee postal address	employee_postal-address_XYZService	1,1	0,0	356
358	Encyclopedia Author Book-Type	encyclopedia_authorbook-type_service	1,2	0,0	
359	Encyclopedia Author	encyclopedia_author_service	1,1	0,0	
360	EncyclopediaPriceService	encyclopedia_price_service	1,1	0,0	
361	Encyclopedia Publisher	encyclopedia_publisher_service	1,1	0,0	
362	EntranceFee	EntranceFeeindollar_service	1,1	0,0	
363	EntranceFee	EntranceFee_service	1,1	0,0	
364	ExpensiveCar Price Color	expensivecar_pricecolor_service	1,2	0,0	
365	Expensive car price	expensivecar_price_service	1,1	0,0	
366	ExpensiveCar RecommendedPrice Color	expensivecar_recommendedpricecolor_service	1,2	0,0	
367	ExpensiveCarTechnology	expensivecar_technology_service	1,1	0,0	
368	ExpensiveCar Year Price	expensivecar_yearprice_service	1,2	0,0	
369	Fall Down Pill	fall_down_pill	1,6	0,0	
370	FantasyNovelprice	fantansynoveluser_price_service	2,1	1,0	
371	FastCar Price Color	fastcar_pricecolor_service	1,2	0,0	
372	FastCar Price Report	fastcar_pricereport_service	1,2	0,0	
373	FastCar RecommendedPrice Color	fastcar_recommendedpricecolor_service	1,2	0,0	
374	FastCar Recommended price	fastcar_recommendedprice_service	1,1	0,0	
375	FastCar TaxedPrice Report	fastcar_taxedpricereport_service	1,2	0,0	
376	FastCarTechnology	fastcar_technology_service	1,1	0,0	
377	FastCar Year Price	fastcar_yearprice_service	1,2	0,0	
378	Fill pills	fill_pills	1,1	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
379	Nearby Postal Codes Finder Service.	findNearbyPostalCodes	5,1	1,0	
380	Nearby Wikipedia Articles Finder Service.	findNearbyWikipediaArticles	6,4	1,0	
381	Postal Codes of Places Finder	findPlaceNamePostalCode	4,5	1,0	
382	Flip Down Slider	flip_down_slider	1,6	0,0	
383	Flip Slider	flip_slider	2,1	0,0	
384	Flip Up Slider	flip_up_slider	1,3	0,0	
385	Food MaxPrice Physical-Quantity	food_maxpricephysical-quantity_Aldiservice	1,2	0,0	
386	AldiFoodQuantityService	food_maxpricequantity_Aldiservice	1,2	0,0	386
387	AldiFoodQuantityService	food_maxpricequantity_service	1,2	0,0	386
388	Food Price Physical-Quantity	food_pricephysical-quantity_Aldiservice	1,2	0,0	
389	Food Price Quantity	food_pricequantity_Aldiservice	1,2	0,0	
390	B-co FoodPriceService	food_price_AnimalFoodservice	1,1	0,0	390
391	FoodPriceService	food_price_service	1,1	0,0	390
392	FoodRecommendPriceService	food_recommendedprice_service	1,1	0,0	
393	Food TaxedPrice Physical-Quantity	food_taxedpricephysical-quantity_Aldiservice	1,2	0,0	
394	Food TaxedPrice Quantity	food_taxedpricequantity_Aldiservice	1,2	0,0	
395	Food TaxFreePrice Physical-Quantity	food_taxfreepricephysical-quantity_Aldiservice	1,2	0,0	
396	Food TaxFreePrice Quantity	food_taxfreepricequantity_Aldiservice	1,2	0,0	
397	Gazetteer location finder service.	gazetteerLookupLocation	3,4	0,0	
398	SurfingGenericAgentDestinationService	generic-agentsports_destination_service	2,1	0,0	398
399	SurfingGenericAgentDestination SportsService	generic-agentsports_destination_SportsService	2,1	0,0	398
400	Postal Address geocoder service.	geocodeUSAddress	4,2	1,0	
401	roadway signs	geographical-regiongeographical-region_icon_service	2,1	1,0	
402	Gorge RouteFinder	geographical-regiongeographical-region_map_Gorgservice	2,1	0,0	402
403	RouteFinder	geographical-regiongeographical-region_map_service	2,1	1,0	402
404	Geographical-Region Accomodation	geographical-region_accomodation_service	1,1	0,0	
405	Geographical-Region BedAndBreakfast	geographical-region_bedandbreakfast_service	1,1	0,0	
406	Geographical-Region Company OccupationalTrade	geographical-region_companyoccupationaltrade_service	1,2	0,0	
407	Geographical-Region Company Profession	geographical-region_companyprofession_service	1,2	0,0	
408	OccupationFinder	geographical-region_companyskilledoccupation_service	1,2	0,0	278
409	Geographical-Region Corporation OccupationalTrade	geographical-region_corporationoccupationaltrade_service	1,2	0,0	
410	Geographical-Region Corporation Profession	geographical-region_corporationprofession_service	1,2	0,0	
411	OccupationFinder	geographical-region_corporationskilledoccupation_service	1,2	0,0	281
412	Geographical-Region Drought	geographical-region_drought_service	1,1	0,0	
413	GeographicalRegionHotelService	geographical-region_hotel_service	1,1	0,0	413
414	GeographicalRegionHotelService	geographical-region_hotel_XYZService	1,1	0,0	413
415	RegionLightiningConditionService	geographical-region_lightning_service	1,1	0,0	
416	Geographical-Region LuxuryHotel	geographical-region_luxuryhotel_service	1,1	0,0	
417	Route from Frankfurt	geographical-region_mapFromFrankfurt_service	1,1	0,0	417
418	Route to Berlin	geographical-region_mapToBerlin_service	1,1	0,0	417
419	Geographical-Region Organization OccupationalTrade	geographical-region_organizationoccupationaltrade_service	1,2	0,0	
420	Geographical-Region Organization Profession	geographical-region_organizationprofession_service	1,2	0,0	
421	OccupationFinder	geographical-region_organizationskilledoccupation_JobService	1,2	0,0	294
422	OccupationFinder	geographical-region_organizationskilledoccupation_service	1,2	0,0	294
423	Geographical-Region WarmFront	geographical-region_warmfront_service	1,1	0,0	
424	GRWF	geographical-region_weatherfront_service	1,1	0,0	424
425	GRWF	geographical-region_weatherfront_WService	1,1	0,0	424
426	GRW	geographical-region_weatherprocess_GRWservice	1,1	0,0	426
427	GRWeatherProcessService	geographical-region_weatherprocess_service	1,1	0,0	426
428	GEWS	geographical-region_weatherseason_service	1,1	0,0	
429	GRWF	geographical-region_weathersystem_service	1,1	0,0	
430	Inside GeopoliticalEntityInfoService	geopolitical-entityrecorded-video_activityhotel_service	2,2	0,0	
431	Geopolitical-Entity Accommodation	geopolitical-entity_accommodation_service	1,1	0,0	
432	Geopolitical-Entity BedAndBreakfast	geopolitical-entity_bedandbreakfast_service	1,1	0,0	
433	Geopolitical-Entity Company OccupationalTrade	geopolitical-entity_companyoccupationaltrade_service	1,2	0,0	
434	Geopolitical-Entity Company Profession	geopolitical-entity_companyprofession_service	1,2	0,0	
435	OccupationFinder	geopolitical-entity_companyskilledoccupation_service	1,2	0,0	278
436	Geopolitical-Entity Corporation OccupationalTrade	geopolitical-entity_corporationoccupationaltrade_service	1,2	0,0	
437	Geopolitical-Entity Corporation Profession	geopolitical-entity_corporationprofession_service	1,2	0,0	
438	OccupationFinder	geopolitical-entity_corporationskilledoccupation_service	1,2	0,0	281
439	Geopolitica-Entity Drought	geopolitical-entity_drought_service	1,1	0,0	
440	GeopoliticalHotelService	geopolitical-entity_hotel_service	1,1	0,0	
441	GIS	geopolitical-entity_internalchange_service	1,1	0,0	
442	Geopolitica-Entity Lightning	geopolitical-entity_lightning_service	1,1	0,0	
443	Geopolitical-Entity LuxuryHotel	geopolitical-entity_luxuryhotel_service	1,1	0,0	
444	Geopolitical-Entity OccupationalTrade FullTimePosition	geopolitical-entity_occupationaltradefulltimeposition_service	1,2	0,0	
445	Geopolitical-Entity OccupationalTrade PartTimePosition	geopolitical-entity_occupationaltradeparttimeposition_service	1,2	0,0	
446	Geopolitical-Entity OccupationalTrade TimeDuration	geopolitical-entity_occupationaltradetimeduration_service	1,2	0,0	
447	Geopolitical-Entity OccupationalTrade TimeMeasure	geopolitical-entity_occupationaltradetimemeasure_service	1,2	0,0	
448	Geopolitical-Entity OccupationalTrade	geopolitical-entity_occupationaltrade_service	1,1	0,0	
449	Geopolitical-Entity Organization OccupationalTrade	geopolitical-entity_organizationoccupationaltrade_service	1,2	0,0	
450	Geopolitical-Entity Organization Profession	geopolitical-entity_organizationprofession_service	1,2	0,0	
451	OccupationFinder	geopolitical-entity_organizationskilledoccupation_service	1,2	0,0	294
452	Geopolitical-Entity Profession FullTimePosition	geopolitical-entity_professionfulltimeposition_service	1,2	0,0	
453	Geopolitical-Entity Profession PartTimePosition	geopolitical-entity_professionparttimeposition_service	1,2	0,0	
454	Geopolitical-Entity Profession TimeDuration	geopolitical-entity_professiontimeduration_service	1,2	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
455	Geopolitical-Entity Profession TimeMeasure	geopolitical-entity_professiontimemeasure_service	1,2	0,0	
456	Geopolitical-Entity Profession	geopolitical-entity_profession_service	1,1	0,0	
457	OccupationFinder	geopolitical-entity_skilledoccupationcompany_service	1,2	0,0	
458	Geopolitical-Entity SkilledOccupation FullTimePosition	geopolitical-entity_skilledoccupationfulltimeposition_service	1,2	0,0	458
459	Geopolitical-Entity SkilledOccupation PartTimePosition	geopolitical-entity_skilledoccupationparttimeposition_service	1,2	0,0	459
460	Geopolitical-Entity SkilledOccupation TimeDuration	geopolitical-entity_skilledoccupationtimeduration_service	1,2	0,0	
461	Geopolitical-Entity SkilledOccupation TimeMeasure	geopolitical-entity_skilledoccupationtimemeasure_service	1,2	0,0	
462	GeoSkilledPositions	geopolitical-entity_skilledoccupation_service	1,1	0,0	
463	Geopolitica-Entity WarmFront	geopolitical-entity_warmfront_service	1,1	0,0	
464	Geopolitica-Entity WeatherFront	geopolitical-entity_weatherfront_service	1,1	0,0	
465	GeoEntityWeatherProcessService	geopolitical-entity_weatherprocess_service	1,1	0,0	
466	GEWS	geopolitical-entity_weatherseasonproposition_service	1,2	0,0	
467	GEIZ weather	geopolitical-entity_weatherseasontimeposition_service	1,2	0,0	
468	GEWS	geopolitical-entity_weatherseason_service	1,1	0,0	
469	GEWeatherSystemService	geopolitical-entity_weathersystem_service	1,1	0,0	
470	Addrees Finder Service.	getAddressOfLocation	3,3	1,0	
471	Altitude Above Sea Level Calculator Service.	getAltitudeAboveSeaLevelOfLocation	3,1	1,0	471
472	Altitude Calculator Service.	getAltitudeOfLocation	3,2	1,0	
473	ATM Location Finder Service.	getATMLocationsInCity	3,2	1,0	
474	Address coordinates finder service.	getCoordinatesOfAddress	5,3	1,0	
475	Worldwide Cities Distance Calculator	getDistanceBetweenCitiesWorldwide	6,1	1,0	
476	Distance calculator between two locations	getDistanceBetweenLocations	4,1	1,0	187
477	Distance finder between two US places	getDistanceBetweenPlaces	4,1	1,0	
478	Driving directions	getDrivingDirections	8,4	1,0	
479	Elevation Finder Service.	getElevationFromLocation	3,1	1,0	471
480	Geographic area of ZipCode finder Service.	getGeographicAreaOfZipCode	2,3	1,0	
481	City matcher service.	getListOfMatchingCities	2,5	1,0	
482	Address location finder service.	getLocationOfAddress	3,2	1,0	
483	Address location finder service.	getLocationOfAddressWorldwide	4,2	1,0	
484	Address location Yahoo Maps service.	getLocationOfAddressYahooMaps	1,4	0,0	
485	City State Location Geocoder service.	getLocationOfCityState	2,4	1,0	
486	GEO Location finder service.	getLocationOfCityWorldwide	3,2	1,0	
487	Zip code location finder service.	getLocationOfUSCity	2,2	1,0	
488	Zip code location finder service.	getLocationOfUSZipcode	1,2	0,0	
489	Location of ZipCode finder Service.	getLocationOfZipCodeWorldwide	2,4	1,0	
490	Check and Lookup address	getMapOfAddress	5,6	1,0	
491	Postal Address Map service.	getMapOfUSAddress	4,1	1,0	
492	GetMedicalFlightAccount	GetMedicalFlightAccount_service	6,1	0,0	
493	GetMedicalTransportAccount	GetMedicalTransportAccount_service	8,1	0,0	
494	GetNonMedicalFlightAccount	GetNonMedicalFlightAccount_service	8,1	0,0	
495	GetNonMedicalTransportAccount	GetNonMedicalTransportAccount_service	6,1	0,0	
496	Address Place finder service.	getPlaceOfAddress	7,3	1,0	
497	Population Density Calculator Service.	getPopulationDensityOfLocation	3,1	1,0	
498	Sunrise and Sunset Time Calcutator Service.	getSunsetAndSunriseTime	5,2	1,0	
499	Sunrise and Sunset Time Calcutator Service.	getSunsetSunriseTimeOfLocation	4,2	1,0	
500	Sunrise Sunset and Twilight Time Calcutator Service.	getSunsetSunriseTwilightTime	5,4	1,0	
501	Traffic information service.	getTrafficInformation	7,3	1,0	
502	US Zip code location finder service.	getUSZipCodeLocation	2,4	1,0	
503	Zip codes finder service.	getZipcodeForUSCity	3,1	1,0	255
504	Zip code info service.	getZipCodeInfo	2,4	1,0	
505	Zip Code info Service that works worldwide.	getZipCodeInfoWorldwide	2,5	1,0	
506	Zip code location finder service.	getZipCodeLocation	2,3	1,0	
507	Zip Codes finder within distance from a city.	getZipCodesWithinCityState	4,2	1,0	
508	GOOGLE GEOCODING API SERVICE.	googleGeocodingAPI	6,4	0,0	
509	GOOGLE STATIC MAPS API SERVICE.	googleStaticMapsAPI	7,1	0,0	
510	Government-Organization Profession	government-organization_profession_service	1,1	0,0	
511	Government-Organization SkilledOccupation	government-organization_skilledoccupation_service	1,1	0,0	
512	GovernmentAwardScholarshipService	governmentaward_scholarship_service	2,1	0,0	
513	GovernmentDegreeGivingBackService	governmentdegree_givingback_service	2,1	0,0	
514	GovernmentDegreeScholarshipQuantityService	governmentdegree_scholarshipquantity_service	2,2	0,0	
515	GovernmentDegreeScholarShipService	governmentdegree_scholarship_service	2,1	0,0	322
516	GovernmentScholarDegreeShipService	governmentdegree_scholarship_TheBestservice	2,1	0,0	322
517	GovernmentDegreeWelfareService	governmentdegree_welfare_service	2,1	0,0	
518	GovMissilesAndWeaponsFundingsService	governmentmissileweapon_fundingfunding_service	3,2	0,0	
519	GovMissilesAndWeaponsFundingService	governmentmissileweapon_funding_service	3,1	0,0	
520	GovMissilesFinancingService	governmentmissile_financing_service	2,1	0,0	
521	GovernmentMissileFundingService	governmentmissile_funding_Reliableservice	2,1	0,0	521
522	GovernmentMissileFundingService	governmentmissile_funding_service	2,1	0,0	521
523	GovOrgMissilesUniliteralGivingService	governmentorganizationmissile_unilateralgiving_service	2,1	0,0	
524	GovOrgMissileFundingService	governmentorganizationselfpowereddevice_funding_service	2,1	0,0	
525	GovernementOrganization ScholarshipService	governmentorganization_scholarship_service	1,1	0,0	
526	GovernmentWeaponFundingService	governmentweapon_funding_service	2,1	0,0	
527	GovernmentFundingService	government_funding_ABombservice	1,1	0,0	527
528	GovernmentFundingService	government_funding_BallMissileservice	1,1	0,0	527
529	GovernmentFundingService	government_funding_ForPhDservice	1,1	0,0	527
530	GovernmentFundingService	government_funding_Missileservice	1,1	0,0	527

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
531	Governement Lending	government_lending_service	1,1	0,0	
532	GovernmengOfferingScholarshipService	government_scholarshiporganization_service	1,2	0,0	
533	Governement Scholarships	government_scholarship_service	1,1	0,0	
534	GovernmengWelfaresService	government_welfare_service	1,1	0,0	
535	Green Light To Off	greenLight_to_off	1,2	0,0	535
536	Green Light To On	greenLight_to_on	1,2	0,0	535
537	GroceryStore BreadOrBiscuit Quantity	grocerystore_breadorbiscuitquantity_service	1,2	0,0	
538	GroceryStore Butter Quantity	grocerystore_butterquantity_service	1,2	0,0	
539	SoldButterService	grocerystore_butterselling_service	1,2	0,0	
540	CakeElementsService	grocerystore_flourdoughbutter_service	1,3	0,0	
541	FodderSellerService	grocerystore_fodder_AnimalFoodservice	1,1	0,0	
542	GroceryStore Food Quantity	grocerystore_foodquantity_service	1,2	0,0	
543	GroceryStoreFoodService	grocerystore_food_service	1,1	0,0	
544	GStorePFoodPriceService	grocerystore_preparedfoodprice_service	1,2	0,0	
545	GroceryStore PreparedFood Quantity	grocerystore_preparedfoodquantity_service	1,2	0,0	112
546	GroceryFoodService	grocerystore_preparedfood_service	1,1	0,0	
547	GroceryStore Sandwich Quantity	grocerystore_sandwichquantity_service	1,2	0,0	
548	AvailableTeaService	grocerystore_teaprice_service	1,2	0,0	
549	HDP2	HDP2_service	1,2	0,0	549
550	HDP	HDP_service	1,2	0,0	
551	SendEMAPhoneNumber	HealthInsurance_service	1,1	0,0	
552	HigherEducationalOrganizationLecturers	higher-educational-organization_lecturer-in-academia_MostUsedservice	1,1	0,0	552
553	HigherEducationalOrganizationLecturers	higher-educational-organization_lecturer-in-academia_service	1,1	0,0	552
554	Pioneer HigherEducationalOrganizationProfessors	higher-educational-organization_professor-in-academia_Pioneerservice	1,1	0,0	554
555	HigherEducationalOrganizationProfessors	higher-educational-organization_professor-in-academia_service	1,1	0,0	554
556	HigherEducationalOrganizationSeniorResearcherFellow	higher-educational-organization_senior-research-fellow-in-academia_Firstservice	1,1	0,0	556
557	HigherEducationalOrganizationSeniorResearcherFellow	higher-educational-organization_senior-research-fellow-in-academia_service	1,1	0,0	556
558	HikingSurfingCityService	hikingsurfing_city_service	2,1	0,0	558
559	Hiking BackpackersDestination	hiking_backpackersdestination_service	1,1	0,0	
560	HikingDestinationService	hiking_destination_service	1,1	0,0	
561	Hiking NationalPark	hiking_nationalpark_service	1,1	0,0	
562	Hiking RuralArea	hiking_ruralarea_service	1,1	0,0	
563	HikingTownService	hiking_town_service	1,1	0,0	
564	Hiking UrbanArea	hiking_urbanarea_service	1,1	0,0	
565	Honey Price	honey_price_service	1,1	0,0	
566	GetPatientMedicalRecords	HospitalPhysician_service	6,2	0,0	
567	Biopsy Availability	hospital_biopsy_service	1,1	0,0	
568	DPA	hospital_diagnosticprocesscost_service	1,2	0,0	
569	Hospital DiagnosticProcess TimeDuration	hospital_diagnosticprocesstimeduration_service	1,2	0,0	549
570	Hospital DiagnosticProcess TimeInterval	hospital_diagnosticprocesstimeinterval_service	1,2	0,0	
571	Hospital DiagnosticProcess TimeMeasure	hospital_diagnosticprocesstimemeasure_service	1,2	0,0	
572	MedDiag	hospital_diagnosticprocess_MedDiagservice	1,1	0,0	572
573	DiagnosticProcessAvailability	hospital_diagnosticprocess_service	1,1	0,0	572
574	Hospital Experiment	hospital_experimenting_service	1,1	0,0	
575	Historical IPP	hospital_IIPsummary_service	1,2	0,0	
576	HospitalInvestService	hospital_investigatingaddress_service	1,2	0,0	
577	InvestigatingFinding	hospital_investigating_service	1,1	0,0	
578	HospitalInvestService	hospital_postal-addressinvestigating_service	1,2	0,0	
579	HistoricalPredicting	hospital_predicting_service	1,1	0,0	
580	RecommendedPriceCoffeeWhiskey Hotel	hotelrecommendedprice_coffeewhiskey_service	2,2	0,0	
581	InformHospital	InformHospital_service	3,1	0,0	
582	BookProviderService	isbn_bookauthor_service	1,2	0,0	
583	BookProviderService	isbn_book_service	1,1	0,0	
584	BookProviderService	isbn_publicationauthor_service	1,2	0,0	
585	BookProviderService	isbn_publicationpublisher_service	1,2	0,0	
586	PublicationSearchingService	isbn_publication_service	1,1	0,0	
587	LetterFounderService	item-number_letter_service	1,1	0,0	
588	KLM LoginService	KLM-Login_service	2,1	0,0	
589	KodakPriceService	KodakDigCamera_price_service	1,1	0,0	589
590	SurfingLearningCentredOganizationDestinationService	learning-centred-organizationsurfing_destination_service	2,1	0,0	
591	LearnedCenteredOrganizationLecturer	learning-centred-organization_lecturer-in-academia_service	1,1	0,0	
592	SurfingLegalAgentDestinationService	legal-agentsurfing_destination_service	2,1	0,0	
593	Lemonfruit Price	lemonfruit_price_service	1,1	0,0	
594	leynthu rent a car	lenthu_rentcar_service	1,1	0,0	209
595	AvailableVideoService	linguisticexpression_videomedia_service	1,1	0,0	
596	RouteFinderService	locationlocation_arrowfigure_service	2,1	1,0	
597	LocationLocationIcon	locationlocation_icon_service	2,1	1,0	
598	RouteFinderService	locationlocation_map_service	2,1	1,0	598
599	SRI RouteFinderService	locationlocation_map_SRIservice	2,1	1,0	598
600	LocationTravelInfo	location_icon_service	1,1	0,0	
601	LocationPhotographs	location_photograph_service	1,1	0,0	
602	Lock door	lock_door	1,1	0,0	259
603	MachinePrice	machine_price_service	1,1	0,0	
604	MAK	MAK_service	1,1	0,0	604
605	MarkoPSservice	MarkoPS_service	1,3	0,0	
606	2in1ColaBeerService	maxprice_beercola_service	1,2	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
607	MaxPriceCoffeeWhiskey	maxprice_coffeewhiskey_service	1,2	0,0	
608	ColaBeerService	maxprice_colabeer_service	1,1	0,0	
609	BreadProviderService	maxprice_colabreadorbiscuit_Bothservice	1,2	0,0	609
610	BreadProviderService	maxprice_colabreadorbiscuit_service	1,2	0,0	609
611	BestColaService	maxprice_cola_Bestservice	1,1	0,0	611
612	AvailableColaService	maxprice_cola_service	1,1	0,0	611
613	DrinksService	maxprice_drinks_Hotservice	1,1	0,0	613
614	DrinksService	maxprice_drinks_service	1,1	0,0	613
615	LiquidService	maxprice_liquid_service	1,1	0,0	
616	WhiskeyService	maxprice_whiskeycolabeer_service	1,2	0,0	
617	Meat MaxPrice Physical-Quantity	meat_maxpricephysical-quantity_service	1,2	0,0	
618	Meat MaxPrice Quantity	meat_maxpricequantity_service	1,2	0,0	
619	Meat Price Physical-Quantity	meat_pricephysical-quantity_Aldiservice	1,2	0,0	
620	Meat Price Quantity	meat_pricequantity_Aldiservice	1,2	0,0	
621	Meat TaxedPrice Physical-Quantity	meat_taxedpricephysical-quantity_service	1,2	0,0	
622	Meat TaxedPrice Quantity	meat_taxedpricequantity_service	1,2	0,0	
623	Meat TaxFreePrice Physical-Quantity	meat_taxfreepricephysical-quantity_Aldiservice	1,2	0,0	
624	Meat TaxFreePrice Quantity	meat_taxfreepricequantity_service	1,2	0,0	
625	MediaPlayer MaxPrice	mediaplayer_maxprice_service	1,1	0,0	
626	MediaPlayer Price	mediaplayer_price_service	1,1	0,0	
627	German Media Player Price	mediaplayer_recommendedpriceineuro_service	1,1	0,0	
628	MediaPlayer TaxedPrice	mediaplayer_taxedprice_service	1,1	0,0	
629	MedicalClinic Biopsy Availability	medicalclinic_biopsy_service	1,1	0,0	
630	MedicalClinic DiagnosticProcess TimeDuration	medicalclinic_diagnosticprocesstimeduration_service	1,2	0,0	
631	MedicalClinic DiagnosticProcess TimeInterval	medicalclinic_diagnosticprocesstimeinterval_service	1,2	0,0	
632	MedicalClinic DiagnosticProcess TimeMeasure	medicalclinic_diagnosticprocesstimemeasure_service	1,2	0,0	
633	MedicalClinic DiagnosticProcess	medicalclinic_diagnosticprocess_service	1,1	0,0	
634	MedicalClinic Experiment	medicalclinic_experimenting_service	1,1	0,0	
635	MedService Investigating	medicalclinic_investigating_MedService	1,1	0,0	635
636	MedInvestigating	medicalclinic_investigating_service	1,1	0,0	635
637	MedicalClinic Predicting	medicalclinic_predicting_service	1,1	0,0	
638	MercantileOrganization Compact Price	mercantileorganization_compactprice_service	1,2	0,0	
639	MOFoodService	mercantileorganization_food_service	1,1	0,0	
640	MercantileOrganization SLR Price	mercantileorganization_slrprice_service	1,2	0,0	
641	MerkelD	MerkelD_service	1,2	0,0	
642	MicrowaveOven Price	microwaveoven_price_service	1,1	0,0	
643	Mile to Kilometer converter service.	mileToKilometerConverter	1,1	0,0	
644	GovernmentOrganization Missile Financing Range	missilegovernmentorganization_financingrange_service	2,2	0,0	
645	GovernmentOrganization Missile Funding Range	missilegovernmentorganization_fundingrange_service	2,2	0,0	
646	GovOrgMissileFundingService	missilegovernmentorganization_funding_service	2,1	0,0	
647	GovernmentOrganization Missile Giving Range	missilegovernmentorganization_givingrange_service	2,2	0,0	
648	GovernmentOrganization Missile Lending Range	missilegovernmentorganization_lendingrange_service	2,2	0,0	
649	Government Missile Financing Range	missilegovernment_financingrange_service	2,2	0,0	
650	Government Missile Funding Range	missilegovernment_fundingrange_service	2,2	0,0	
651	Government Missile Giving Range	missilegovernment_givingrange_service	2,2	0,0	
652	GovMissilesGivingService	missilegovernment_giving_Borrowservice	2,1	0,0	652
653	GovMissilesGivingService	missilegovernment_giving_service	2,1	0,0	652
654	Government Missile Lending Range	missilegovernment_lendingrange_service	2,2	0,0	
655	ChinaMissilesFinancingService	missile_financing_Chinaservice	1,1	0,0	655
656	RussianMissilesFinancingService	missile_financing_Russianservice	1,1	0,0	655
657	USMissilesFinancingService	missile_financing_USservice	1,1	0,0	655
658	AsianMissilesFundingService	missile_funding_Asianservice	1,1	0,0	658
659	IndiaMissilesFundingService	missile_funding_Indiaservice	1,1	0,0	658
660	NKoreaMissilesFundingService	missile_funding_NKoreaservice	1,1	0,0	658
661	PakistanMissilesFundingService	missile_funding_Pakservice	1,1	0,0	658
662	AuthorisedPersonMonographPriceService	monographpersoncreditcardaccount_recommendedprice_service	3,1	1,0	
663	MonographShopping	monographperson__service	2,1	1,1	
664	Monograph Author	monograph_author_service	1,1	0,0	
665	MonographPriceService	monograph_price_service	1,1	0,0	
666	Monograph Publisher	monograph_publisher_service	1,1	0,0	
667	RecommendedPriceService	monograph_recommendedpriceineuro_service	1,1	0,0	
668	2For 1 Price	mp3playercdplayermicrowaveoven_price_service	3,1	0,0	
669	2for 1 Price and Shipping	mp3playerdvdplayer_priceshipping_service	2,2	0,0	
670	2for 1 RecommendedPrice	mp3playerdvdplayer_recommendedprice_service	2,1	0,0	344
671	2for 1 Price and Shipping	mp3playerdvdplayer_Recpriceshipping_service	2,2	0,0	
672	US-MD	mp3playerdvdplayer_Recpriceshipping_USservice	2,2	0,0	
673	2For 1 Price	mp3playerportabledvdplayer_price_service	2,1	0,0	
674	2for 1 Price	mp3playerportabledvdplayer_recommendedpricequality_service	2,2	0,0	
675	MP3Player MaxPrice	mp3player_maxprice_service	1,1	0,0	
676	MP3Player TaxedPrice	mp3player_taxedprice_service	1,1	0,0	
677	Municipal-Unity Drought	municipal-unit_drought_service	1,1	0,0	
678	Municipal-Unity Lightning	municipal-unit_lightning_service	1,1	0,0	
679	Municipal-Unit OccupationalTrade FullTimePosition	municipal-unit_occupationaltradefulltimeposition_service	1,2	0,0	
680	Municipal-Unit OccupationalTrade PartTimePosition	municipal-unit_occupationaltradeparttimeposition_service	1,2	0,0	
681	Municipal-Unit OccupationalTrade TimeDuration	municipal-unit_occupationaltradetimeduration_service	1,2	0,0	
682	Municipal-Unit OccupationalTrade TimeMeasure	municipal-unit_occupationaltradetimemeasure_service	1,2	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
683	Municipal-Entity Profession FullTimePosition	municipal-unit_professionfulltimeposition_service	1,2	0,0	
684	Municipal-Entity Profession PartTimePosition	municipal-unit_professionparttimeposition_service	1,2	0,0	
685	Municipal-Unit Profession TimeDuration	municipal-unit_professiontimeduration_service	1,2	0,0	
686	Municipal-Unit Profession TimeMeasure	municipal-unit_professiontimemeasure_service	1,2	0,0	
687	Geopolitical-Entity SkilledOccupation FullTimePosition	municipal-unit_skilledoccupationfulltimeposition_service	1,2	0,0	458
688	Geopolitical-Entity SkilledOccupation PartTimePosition	municipal-unit_skilledoccupationparttimeposition_service	1,2	0,0	459
689	Municipal-Unit SkilledOcupation TimeDuration	municipal-unit_skilledoccupationtimeduration_service	1,2	0,0	
690	Municipal-Unit SkilledOcupation TimeMeasure	municipal-unit_skilledoccupationtimemeasure_service	1,2	0,0	
691	Municipal-Unit WarmFront	municipal-unit_warmfront_service	1,1	0,0	
692	Municipal-Unit WeatherFront	municipal-unit_weatherfront_service	1,1	0,0	
693	Municipal-Unit WeatherProcess	municipal-unit_weatherprocess_service	1,1	0,0	
694	Municipal-Unit WeatherSeason	municipal-unit_weatherseason_service	1,1	0,0	
695	Municipal-Unit WeatherSystem	municipal-unit_weathersystem_service	1,1	0,0	
696	NationalGovWeaponFundingService	nationalgovernmentweapon_funding_service	2,1	0,0	
697	NationalGovernement Lending	nationalgovernment_lending_service	1,1	0,0	
698	NationalGovernement offers	nationalgovernment_physical-quantityscholarshiplandarea_service	1,3	0,0	
699	NationalGovernmentScholarship	nationalgovernment_scholarshipquantityduration_service	1,3	0,0	
700	NationalGovernmentScholarship	nationalgovernment_scholarshipquantity_service	1,2	0,0	
701	NationalGovernmentScholarship	nationalgovernment_scholarship_service	1,1	0,0	
702	ProvideNonMedicalFlightInformation	NonMedicalFlightCompany_service	5,1	0,0	
703	NovelPrice	novelperson_price_service	2,1	0,0	
704	NovelPrice Reservation	novelperson__Reservationservice	2,1	1,1	
705	Novel Author Book-Type	novel_authorbook-type_service	1,2	0,0	
706	NovelAuthorCommittingService	novel_authorcommitting_service	1,2	0,0	
707	NovelAuthorGenreService	novel_authorgenre_service	1,2	0,0	
708	Novel Author MaxPrice	novel_authormaxprice_service	1,2	0,0	
709	Novel Author Price	novel_authorprice_service	1,2	0,0	
710	Novel Author RecommendedPrice	novel_authorrecommendedprice_service	1,2	0,0	
711	Novel Author TaxedPrice	novel_authortaxedprice_service	1,2	0,0	
712	NovelAuthorService	novel_authortime_service	1,2	0,0	
713	NovelAuthorService	novel_author_BookOntoservice	1,1	0,0	713
714	NovelAuthorService	novel_author_MyOntoservice	1,1	0,0	713
715	NovelAuthorService	novel_author_service	1,1	0,0	713
716	NovelPersonService	novel_person_Reserverservice	1,1	1,0	716
717	NovelPersonService	novel_person_Writerservice	1,1	0,0	716
718	NovelPrice	novel_price_service	1,1	0,0	
719	Novel Publisher	novel_publisher_service	1,1	0,0	
720	NovelAuthorURService	novel_userreviewauthor_service	1,2	0,0	
721	AuthorisedPersonObjectPriceService	objectpersoncreditaccount_price_service	3,1	1,0	
722	Shopping	objectperson__service	2,1	1,1	
723	Objects Mapping	objectsMappingService	8,6	0,0	
724	Open door	open_door	1,1	0,0	259
725	Orangefruit Price	orangefruit_price_service	1,1	0,0	
726	HikingOrganizationDestinationService	organizationhiking_destination_service	2,1	0,0	
727	ODGCService	organization_diagnosticprocesscost_service	1,2	0,0	
728	ODGService	organization_diagnosticprocess_service	1,1	0,0	
729	OrganizationExp	organization_experimentingtimeduration_service	1,2	0,0	
730	OrganizationExp	organization_experimenting_service	1,1	0,0	
731	OrganizationLecturer	organization_lecturer-in-academia_service	1,1	0,0	
732	PatientTransport	PatientTransport_service	4,2	0,0	
733	Pea Price	pea_price_service	1,1	0,0	
734	Book Shopping	personbookliabilityaccount__service	3,1	1,1	
735	PersonCityCountryInfoService	personcountrycity_sportshotel_service	3,2	1,0	
736	Monograph Shopping	personmonographcreditcardaccount__service	3,1	1,1	
737	Person address	person_address_service	1,1	0,0	
738	KodakPriceService	PhillipDigCamera_price_service	1,1	0,0	
739	2for 1 RecommendedPrice	portabledvdplayermp3player_recommendedpricetaxedprice_servi	2,2	0,0	739
740	2for 1 RecommendedPrice	portabledvdplayermp3player_recommendedpricetaxedprice_service	2,2	0,0	739
741	EuropeCityHotelInfoService	postal-addresscity_hotel_service	2,1	1,0	
742	GroceryStoreRecommendedPrice	preparedfood_GSprice_service	1,2	0,0	
743	PreparedFood MaxPrice	preparedfood_maxprice_service	1,1	0,0	
744	PFP	preparedfood_priceday_service	1,2	0,0	
745	PreparedFoodPriceService	preparedfood_price_service	1,1	0,0	
746	PreparedFood RecommendedPrice	preparedfood_recommendedprice_service	1,1	0,0	
747	SpainishTax on PrepardFood	preparedfood_SpainishTax_service	1,1	0,0	
748	PTP	preparedfood_taxedpriceindollarprice_service	1,2	0,0	
749	PreparedFood TaxedPrice	preparedfood_taxedprice_service	1,1	0,0	
750	PTS	preparedfood_taxfreeprice_service	1,1	0,0	
751	USTax on PrepardFood	preparedfood_USTax_service	1,1	0,0	
752	TCW	price_coffeewhiskeyqualitytimeposition_service	1,4	0,0	
753	POZ	price_coffeewhiskeyquality_service	1,3	0,0	
754	CWT	price_coffeewhiskeytimemeasure_service	1,3	0,0	
755	DFKI	price_coffeewhiskey_service	1,2	0,0	755
756	TheBest	price_coffeewhiskey_Thebestservice	1,2	0,0	755
757	CoffeewithWhiskeyPrice	price_coffeewithwhiskey_service	1,1	0,0	
758	Guddu ColaService	price_cola_Gudduservice	1,1	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
759	Hallo2 ColaService	price_cola_Hallo2service	1,1	0,0	759
760	Hallo ColaService	price_cola_Halloservice	1,1	0,0	759
761	DP	price_drinks_service	1,1	0,0	
762	MIR	price_irishcoffeemixerycola_service	1,2	0,0	
763	PriceCoffeeWhiskey	price_whiskeycoffee_service	1,2	0,0	755
764	PriceCoffeeWhiskey	price_whiskeycoffee_Zikoservice	1,2	0,0	755
765	AuthorisedPersonPrintedMaterialPriceService	printedmaterialpersoncreditcardaccount_price_service	3,1	1,0	
766	PrintedMaterialShopping	printedmaterialpersoncreditcardaccount__service	3,1	1,1	
767	PublicationShopping	printedmaterialperson__service	2,1	1,1	
768	PrintedMaterialPriceService	printedmaterial_price_service	1,1	0,0	
769	Professor address	professor-in-academia_address_service	1,1	0,0	
770	Profit-Organization Profession	profit-organization_profession_service	1,1	0,0	
771	Profit-Organization SkilledOccupation	profit-organization_skilledoccupation_service	1,1	0,0	
772	GovernmentProjectileWeaponFundingService	projectilegovernment_funding_service	2,1	0,0	
773	PrCJ	project_skilledoccupation_service	1,1	0,0	
774	ProvideMedicalFlightInformation	ProvideMedicalFlightInformation_services	8,1	0,0	
775	ProvideMedicalTransportInformation	ProvideMedicalTransportInformation_service	6,1	0,0	
776	ProvideNonMedicalTransportInformation	ProvideNonMedicalTransportInformation_service	4,1	0,0	
777	PCB	public-companycountry_skilledoccupation_service	2,1	0,0	
778	BookSearchingService	publication-number_bookauthorpublisher_service	1,3	0,0	
779	BookSearchingService	publication-number_bookauthor_service	1,2	0,0	
780	BookFinderService	publication-number_book_Portalservice	1,1	0,0	35
781	ABN/ISBN Booksearch	publication-number_book_service	1,1	0,0	35
782	ULiPublicationLocatorService	publication-number_currencypublication_service	1,2	0,0	
783	EditedBookFinderService	publication-number_edited-book_service	1,1	0,0	
784	PublicationFinderService	publication-number_publicationauthor_service	1,2	0,0	
785	PublicationSearchingService	publication-number_publication_service	1,1	0,0	785
786	Publication Shopping	publicationperson__service	2,1	1,1	
787	PublicationAuthorService	publication_author_service	1,1	0,0	
788	publication book	publication_book_service	1,1	0,0	
789	PublicationSearchingService	publication_number_publication_service	1,1	0,0	785
790	Publication Publisher	publication_publisher_service	1,1	0,0	
791	ColaProviderService	qualitymaxprice_cola_service	2,1	0,0	
792	Query Parser location finder service.	queryParserLocation	4,4	0,0	
793	DiagnosticProcessAvailability	questionhospital_diagnosticprocess_service	2,1	0,0	
794	RawFoodPrice	rawfood_price_service	1,1	0,0	
795	Reader-in-academia address	reader-in-academia_address_service	1,1	0,0	
796	Real-time geocoding service.	real-time_geocoding	4,1	0,0	
797	Real-time geocoding service.	real-time_geocodingStreetAddress	6,2	0,0	
798	USWhiskeyCoffee	recommendedpriceindollar_whiskeycoffee_service	1,2	0,0	
799	EuroPriceWhiskeyCoffee	recommendedpriceineuro_coffeewhiskey_service	1,2	0,0	
800	RPCW brand	recommendedprice_coffeewhiskeysymbolicstringquality_service	1,4	0,0	
801	2for1CW brand	recommendedprice_coffeewhiskeysymbolicstring_service	1,3	0,0	
802	RecommendedPriceCoffeeWhiskey	recommendedprice_coffeewhiskey_Bestservice	1,2	0,0	802
803	RecommendedPriceCoffeeWhiskey	recommendedprice_coffeewhiskey_service	1,2	0,0	802
804	CoffeewithWhiskeyPrice	recommendedprice_coffeewithwhiskeycontentbearingobject_service	1,2	0,0	
805	CoffeeWhiskeyPriceInfo	recommendedprice_contentbearingobjectwhiskeycoffee_service	1,3	0,0	
806	Ana IrishCoffeeforPrice	recommendedprice_irishcoffeetasting_service	1,2	0,0	
807	IrishCoffeeforPrice	recommendedprice_irishcoffee_service	1,1	0,0	
808	Red Light To Off	redLight_to_off	1,2	0,0	808
809	Red Light To On	redLight_to_on	1,2	0,0	808
810	Recommended price of Renault car	Renaultyear_recommendedpriceineuro_service	1,1	0,0	
811	Render Map Service.	renderMapService	5,1	0,0	
812	RequiredFoodService	Required_preparedfoodquantity_service	1,2	0,0	112
813	ResearchAssistant address	research-assistant-in-academia_address_service	1,1	0,0	
814	ResearchFellowPublicationReferences	research-fellow-in-academia_publication-reference_service	1,1	0,0	
815	Researcher abstract information	researcher-in-academia_abstract-information_service	1,1	0,0	
816	Researcher address	researcher-in-academia_address_service	1,1	0,0	816
817	TREE Researcher address	researcher-in-academia_address_TREEservice	1,1	0,0	816
818	ZOO Researcher address	researcher-in-academia_address_ZOOservice	1,1	0,0	816
819	Researcher postal address and publication references	researcher-in-academia_publication-referencepostal-address_	1,2	0,0	819
820	Researcher postal address and publication references	researcher-in-academia_publication-referencepostal-address_service	1,2	0,0	819
821	Researcher abstract information	researcher_abstract-information_service	1,1	0,0	
822	HOM2 Researcher address	researcher_address_HOM2service	1,1	0,0	822
823	Researcher address	researcher_address_service	1,1	0,0	822
824	Researcher postal address	researcher_postal-address_service	1,1	0,0	
825	AppleSellerService	retailstore_apple_service	1,1	0,0	
826	RetailStore BreadOrBiscuit Quantity	retailstore_breadorbiscuitquantity_service	1,2	0,0	
827	RetailStore Butter Quantity	retailstore_butterquantity_service	1,2	0,0	
828	RetailStore Compact Price	retailstore_compactprice_service	1,2	0,0	
829	RetailStoreFoodService	retailstore_foodquality_service	1,2	0,0	
830	RetailStore Food Quantity	retailstore_foodquantity_service	1,2	0,0	
831	RetailStore PreparedFood Quantity	retailstore_preparedfoodquantity_service	1,2	0,0	
832	RetailStoreFoodService	retailstore_preparedfood_service	1,1	0,0	
833	RetailStore Sandwich Quantity	retailstore_sandwichquantity_service	1,2	0,0	
834	RetailStore SLR Price	retailstore_slrprice_service	1,2	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
835	RetailStore SLR TaxedPrice	retailstore_slrtaxedprice_service	1,2	0,0	
836	RomanticNovel Author Book-Type	romanticnovel_authorbook-type_service	1,2	0,0	
837	RomanticNovel Author MaxPrice	romanticnovel_authormaxprice_service	1,2	0,0	
838	RomanticNovel Author Price	romanticnovel_authorprice_service	1,2	0,0	
839	RomanticNovel Author RecommendedPrice	romanticnovel_authorrecommendedprice_service	1,2	0,0	
840	RomanticNovel Author TaxedPrice	romanticnovel_authortaxedprice_service	1,2	0,0	
841	Romantic Publisher	romanticnovel_publisher_service	1,1	0,0	
842	Sandwich MaxPrice	sandwich_maxprice_service	1,1	0,0	
843	Sandwich RecommendedPrice	sandwich_recommendedprice_service	1,1	0,0	
844	Sandwich TaxedPrice	sandwich_taxedprice_service	1,1	0,0	
845	ScienceFictionNovelUserRecommendedPrice	science-fiction-noveluser_recommendedpriceindollar_service	2,1	1,0	
846	Science-Fiction-Novel Author Book-Type	science-fiction-novel_authorbook-type_service	1,2	0,0	
847	Science-Fiction-Novel Author MaxPrice	science-fiction-novel_authormaxprice_service	1,2	0,0	
848	Science-Fiction-Novel Author Price	science-fiction-novel_authorprice_service	1,2	0,0	848
849	Science-Fiction-Novel Author RecommendedPrice	science-fiction-novel_authorrecommendedprice_service	1,2	0,0	
850	Science-Fiction-Novel Author TaxedPrice	science-fiction-novel_authortaxedprice_service	1,2	0,0	
851	ScienceFictionNovel priceAuthor	science-fiction-novel_priceauthor_service	1,2	0,0	848
852	Science-Fiction-Novel Publisher	science-fiction-novel_publisher_service	1,1	0,0	
853	ScienceFictionBook Price	sciencefictionbookuser_price_service	2,1	1,0	
854	ScienceFictionBook Author Book-Type	sciencefictionbook_authorbook-type_service	1,2	0,0	
855	ScienceFictionBook Author MaxPrice	sciencefictionbook_authormaxprice_service	1,2	0,0	
856	ScienceFictionBook Author Price	sciencefictionbook_authorprice_service	1,2	0,0	
857	ScienceFictionBook Author RecommendedPrice	sciencefictionbook_authorrecommendedprice_service	1,2	0,0	
858	ScienceFictionBook Author TaxedPrice	sciencefictionbook_authortaxedprice_service	1,2	0,0	
859	SFBookAuthorService	sciencefictionbook_author_service	1,1	0,0	
860	ScienceFictionBook Publisher	sciencefictionbook_publisher_service	1,1	0,0	
861	SFNovelAuthorService	sciencefictionnovel_author_MyOntoservice	1,1	0,0	
862	SFNovelReview	ScienceFNovelReview_service	2,2	1,0	
863	Search Formatted Address	searchFormattedAddress	2,3	0,0	863
864	Search Raw address	searchRawAddress	2,3	0,0	863
865	SeePatientMedicalRecords	SeePatientMedicalRecords_service	4,1	0,0	
866	SelectFlight	SelectFlight_service	1,1	0,0	
867	SelectTransport	SelectTransport_service	1,1	0,0	
868	SetUpCostAndHealingPlan	SetUpCostAndHealingPlan_service	8,2	0,0	
869	ScienceFictionNovelReview	SFNovelReview_service	2,2	1,0	
870	Science-Fiction-Novel Author Author	sfnovel_authorauthor_BookOntoservice	1,2	0,0	
871	ScienceFictionNovelRecommendedPrice	SFNRecommendedPrice_service	2,1	1,0	
872	AnalogCameraService	shoppingmall_analogpricecalendar-date_service	1,3	0,0	
873	CameraPriceService	shoppingmall_calendar-datepricecamera_service	1,3	0,0	
874	ShoppingMallCameraPriceService	shoppingmall_cameraprice_service	1,2	0,0	
875	ShoppingMall Compact Price	shoppingmall_compactprice_service	1,2	0,0	
876	ShoppingMall Compact TaxedPrice	shoppingmall_compacttaxedprice_service	1,2	0,0	
877	DigitalSLRPriceService	shoppingmall_digital-slrpricecalendar-date_service	1,3	0,0	
878	DVCameraPriceService	shoppingmall_maxpricedigital-video_service	1,2	0,0	
879	HandyCameraPriceService	shoppingmall_pricecellphonewithcamera_service	1,2	0,0	
880	DigitalAnologCameraPrice	shoppingmall_pricedigitalanalog_service	1,3	0,0	
881	SPP	shoppingmall_pricepurchaseableitemrange_service	1,3	0,0	
882	SHOPPINGMALLPURCHASEABLEITEMPRICE	shoppingmall_purchaseableitemprice_service	1,2	0,0	
883	ShirtRecommendedPrice	shoppingmall_recommendedprice_Shirtservice	1,1	0,0	
884	ShoppingMall SLR Price	shoppingmall_slrprice_service	1,2	0,0	
885	Short-Story Author Book-Type	short-story_authorbook-type_service	1,2	0,0	
886	Short-Story Author MaxPrice	short-story_authormaxprice_service	1,2	0,0	
887	Short-Story Author Price	short-story_authorprice_service	1,2	0,0	
888	Short-Story Author RecommendedPrice	short-story_authorrecommendedprice_service	1,2	0,0	
889	Short-Story Author TaxedPrice	short-story_authortaxedprice_service	1,2	0,0	
890	Short-Story Author	short-story_author_service	1,1	0,0	
891	Short-Story Publisher	short-story_publisher_service	1,1	0,0	
892	Sightseeing City	sightseeing_city_service	1,1	0,0	
893	Sightseeing Nationalpark	sightseeing_nationalpark_service	1,1	0,0	
894	Sightseeing Town	sightseeing_town_service	1,1	0,0	
895	SPD-Grune	SPD-Grune_service	1,2	0,0	
896	SportsLegalAgentDestinationService	sportslegal-agent_destination_service	2,1	0,0	
897	SportsOrganizationDestination	sportsorganization_destination_service	2,1	0,0	
898	SportsBeachService	sports_beach_service	1,1	0,0	
899	SportsDestinationService	sports_destination_service	1,1	0,0	
900	Sports Farmland	sports_farmland_service	1,1	0,0	
901	SportsNationalParkService	sports_nationalpark_service	1,1	0,0	
902	SportsRuralAreaService	sports_ruralarea_service	1,1	0,0	
903	SportsTownService	sports_town_service	1,1	0,0	
904	SRCameraService	SRcamera_service	1,2	0,0	
905	STATIC MAPS DISPLAY SERVICE.	staticMapsDisplay	11,1	0,0	
906	MerchantService	store_preparedfood_Merchantservice	1,1	0,0	906
907	StorePFoodService	store_preparedfood_service	1,1	0,0	906
908	SurfingGenericAgentCityService	surfinggeneric-agent_city_service	2,1	0,0	
909	SurfingGenericAgentDestinationService	surfinggeneric-agent_destination_service	2,1	0,0	
910	SURFINGHIKINGCityService	surfinghiking_city_service	2,1	0,0	558

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
911	SFCityService	surfinghiking_city_SFservice	2,1	0,0	558
912	DFG HikingSurfingDestination	surfinghiking_destination_DFGservice	2,1	0,0	912
913	PDS HikingSurfingDestination	surfinghiking_destination_PDSservice	2,1	0,0	912
914	PF Destination	surfinghiking_destination_PFservice	2,1	0,0	912
915	HikingSurfingDestination	surfinghiking_destination_service	2,1	0,0	912
916	SurfingHikingNationalParkService	surfinghiking_nationalpark_service	2,1	0,0	
917	HikingSurfingRuralAreaService	surfinghiking_ruralarea_service	2,1	0,0	
918	SurfingHikingTownService	surfinghiking_town_service	2,1	0,0	
919	SurfingOrganizationDestinationService	surfingorganizationperson_destination_service	3,1	1,0	
920	SurfingOrganizationCityService	surfingorganization_city_service	2,1	0,0	
921	SurfingOrganizationDESTINATIONService	surfingorganization_destination_Bestservice	2,1	0,0	921
922	SAAR	surfingorganization_destination_SAARservice	2,1	0,0	921
923	SurfingOrganizationDESTINATIONService	surfingorganization_destination_service	2,1	0,0	921
924	SODService	surfingorganization_destination_SODservice	2,1	0,0	921
925	SurfingBeachService	surfing_beach_service	1,1	0,0	
926	Always SurfingDestinationService	surfing_destination_Alwaysservice	1,1	0,0	926
927	AUS SurfingDestinationService	surfing_destination_AUSservice	1,1	0,0	926
928	SurfingDestinationService	surfing_destination_service	1,1	0,0	926
929	SOH SurfingDestinationService	surfing_destination_SOHservice	1,1	0,0	926
930	SurfingFarmlandService	surfing_farmland_service	1,1	0,0	
931	SurfingNationalParkService	surfing_nationalpark_service	1,1	0,0	
932	SurfingRuralAreaService	surfing_ruralarea_service	1,1	0,0	
933	Switch Off MesseModul	switch_off_messemodul	1,1	0,0	933
934	Switch On MesseModul	switch_on_messemodul	1,1	0,0	933
935	TaxFreeColaService	taxfreeprice_cola_service	1,1	0,0	
936	TaxPCW	taxfreeprice_whiskeycoffee_service	1,2	0,0	
937	Tea MaxPrice	tea_maxprice_service	1,1	0,0	
938	Tea RecommendedPrice	tea_recommendedprice_service	1,1	0,0	
939	Tea TaxedPrice	tea_taxedprice_service	1,1	0,0	
940	PublicationFounderService	text_publication_service	1,1	0,0	
941	Time-Measure Country City Hotel InfoService	time-measurecountrycity_hotel_service	3,1	1,0	
942	Time-Measure Geopolitical-Entity City Accommodation InfoService	time-measuregeopolitical-entitycity_accommodation_service	3,1	1,0	
943	Time-Measure Geopolitical-Entity City BedAndBreakfast InfoService	time-measuregeopolitical-entitycity_bedandbreakfast_service	3,1	1,0	
944	Time-Measure Geopolitical-Entity City Hotel InfoService	time-measuregeopolitical-entitycity_hotel_service	3,1	1,0	
945	Title saving	TitleSaving_service	1,1	0,1	
946	Title ActionFilm MaxPrice Quality	title_actionfilmmaxpricequality_service	1,3	0,0	
947	Title ActionFilm Price Quality	title_actionfilmpricequality_service	1,3	0,0	
948	Title ActionFilm RecommendedPrice Quality	title_actionfilmrecommendedpricequality_service	1,3	0,0	
949	Title ActionFilm TaxedPrice Quality	title_actionfilmtaxedpricequality_service	1,3	0,0	
950	Title ActionFilm TaxFreePrice Quality	title_actionfilmtaxfreepricequality_service	1,3	0,0	
951	ABC Film finder	title_actionfilm_service	1,1	0,0	
952	CD Price Software Title	title_cdpricesoftwareStreaming_service	1,3	0,0	
953	Title Comedy MaxPrice Quality	title_comedyfilmmaxpricequality_service	1,3	0,0	
954	Title ComedyFilm Price Quality	title_comedyfilmpricequality_service	1,3	0,0	
955	Title ComedyFilm RecommendedPrice Quality	title_comedyfilmrecommendedpricequality_service	1,3	0,0	
956	Title ComedyFilm TaxedPrice Quality	title_comedyfilmtaxedpricequality_service	1,3	0,0	
957	Title ComedyFilm TaxFreePrice Quality	title_comedyfilmtaxfreepricequality_service	1,3	0,0	
958	BF Comedy Film Searcher	title_comedyfilm_BFservice	1,1	0,0	958
959	Mega Comedy Film finder	title_comedyfilm_Megaservice	1,1	0,0	958
960	Comedy Film finder	title_comedyfilm_service	1,1	0,0	958
961	HighComedy Action Film	title_filmActionComedy_service	1,1	0,0	961
962	Title Film MaxPrice Quality	title_filmmaxpricequality_service	1,3	0,0	
963	Film locator	title_filmP2P_service	1,1	0,0	961
964	ZAF Film finder	title_filmpricequality_service	1,3	0,0	
965	Title Film RecommendedPrice Quality	title_filmrecommendedpricequality_service	1,3	0,0	
966	Title Film TaxedPrice Quality	title_filmtaxedpricequality_service	1,3	0,0	
967	Title Film TaxFreePrice Quality	title_filmtaxfreepricequality_service	1,3	0,0	
968	Film finder	title_film_service	1,1	0,0	961
969	HighComedy Film	title_highcomedyfilmreport_service	1,2	0,0	
970	LowComedy Action Film	title_lowcomedyfilm_service	1,1	0,0	
971	Title Media MaxPrice Quality	title_mediamaxpricequality_service	1,3	0,0	
972	Title Media Price Quality	title_mediapricequality_service	1,3	0,0	
973	Title Media RecommendedPrice Quality	title_mediarecommendedpricequality_service	1,3	0,0	
974	Title Media TaxedPrice Quality	title_mediataxedpricequality_service	1,3	0,0	
975	Title Media TaxFreePrice Quality	title_mediataxfreepricequality_service	1,3	0,0	
976	Media finder	title_media_service	1,1	0,0	
977	Obtainalbe Video Media finder	title_obtainablevideomedia_service	1,1	0,0	
978	KAHN BookFinderPriceService	title_pricebook_service	1,2	0,0	
979	Title ScienceFictionFilm MaxPrice Quality	title_sciencefictionfilmmaxpricequality_service	1,3	0,0	
980	Title ScienceFictionFilm Price Quality	title_sciencefictionfilmpricequality_service	1,3	0,0	
981	Title ScienceFictionFilm RecommendedPrice Quality	title_sciencefictionfilmrecommendedpricequality_service	1,3	0,0	
982	Title ScienceFictionFilm TaxedPrice Quality	title_sciencefictionfilmtaxedpricequality_service	1,3	0,0	
983	Title ScienceFiction TaxFreePrice Quality	title_sciencefictionfilmtaxfreepricequality_service	1,3	0,0	
984	DVD and VHS	title_vhsdvd_service	1,2	0,0	
985	VHS finder	title_vhs_service	1,1	0,0	
986	Title VideoMedia MaxPrice Quality	title_videomediamaxpricequality_service	1,3	0,0	

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
987	MMVideoMediaService	title_videomediaMM_service	1,1	0,0	987
988	Title VideoMedia Price Quality	title_videomediapricequality_service	1,3	0,0	
989	Title VideoMedia RecommendedPrice Quality	title_videomediarecommendedpricequality_service	1,3	0,0	
990	videoMedia finder	title_videomediarecommendedprice_service	1,2	0,0	
991	Title VideoMedia TaxedPrice Quality	title_videomediataxedpricequality_service	1,3	0,0	
992	Title VideoMedia TaxFreePrice Quality	title_videomediataxfreepricequality_service	1,3	0,0	
993	AvailableVideoService	title_videomedia_service	1,1	0,0	987
994	Tizon RecommendedPriceService	Tizonbook_recommendedpriceindollar_service	1,1	0,0	175
995	HotelInfoService	towncountry_hotel_service	2,1	1,0	
996	UniversityAcademicSupportStaffService	university_academic-support-staff_service	1,1	0,0	
997	UniversityLecturerService	university_lecturer-in-academiaCurrentSemmester_service	1,1	0,0	997
998	UniversityLecturerService	university_lecturer-in-academia_Recommendservice	1,1	0,0	997
999	UniversityLecturerService	university_lecturer-in-academia_service	1,1	0,0	997
1000	UniversityProfessorService	university_professor-in-academia_service	1,1	0,0	
1001	UniversityResearchFellowService	university_research-fellow-in-academia_service	1,1	0,0	
1002	UniversityResearcherService	university_researcher_service	1,1	0,0	
1003	UniversityLecturerService	university_senior-lecturer-in-academia_service	1,1	0,0	
1004	Unlock door	unlock_door	1,1	0,0	259
1005	HistoricalDiagnositc	UnsuccessfulDiagnosis_service	1,2	1,0	
1006	UntagibleObjectsService	untangibleobjects_cola_service	1,1	0,0	
1007	UpdatePatientMedicalRecords	UpdatePatientMedicalRecords_service	9,1	0,0	
1008	UserBook Price	userbook_price_service	2,1	1,0	
1009	RomanticNovelPrice	userRomanticnovel_price_service	2,1	1,0	
1010	SFNovelPrice	userscience-fiction-novel_price_Bestservice	2,1	1,0	331
1011	SFNovelPrice	userscience-fiction-novel_price_service	2,1	1,0	331
1012	SFNovelPrice	userscience-fiction-novel_Relprice_service	2,1	1,0	331
1013	Shoppingstatus	user_price_ShoppingStatusservice	1,1	1,1	
1014	Distance finder between two postal codes	usPostalCode_distance	2,1	0,0	
1015	Distance finder between two zipcodes	uszipcode_distance	2,1	0,0	
1016	Vehicle price	vehicle_price_service	1,1	0,0	
1017	VillageHotelService	village_hotel_service	1,1	0,0	
1018	Visiting Researcher address	visiting-researcher_address_service	1,1	0,0	
1019	IraqMissilesAndMassDestructionWeaponsFundingService	weaponmissile_funding_Iraqservice	2,1	0,0	
1020	WStoreFoodService	wholesalestore_preparedfood_service	1,1	0,0	
1021	2 for 1 Price	wirelessplayerwmaplayer_price_service	2,1	0,0	
1022	ZAD	ZAD_service	1,1	0,0	604
1023	Three wheeled Car price	_3WheeledAudiCarprice_service	1,1	0,0	589
1024	Another car price	_3WheeledOpelCarPrice_service	1,1	0,0	589
1025	MuseumCamera	_aps-slrpricereport_Musuemservice	1,3	0,0	
1026	CompJ AuthorService	_author_CompJservice	1,1	0,0	1026
1027	DataMiningAuthorService	_author_DMservice	1,1	0,0	1026
1028	CityBankBankers	_bankeraddress_CityBankservice	1,2	0,0	
1029	Computer Networking Training	_book_Oracleservice	1,1	0,0	
1030	Camera Price	_cameraprice_MyShopservice	1,2	0,0	1030
1031	CameraTDPriceService	_camerataxedpricedutytax_service	1,3	0,0	
1032	CameraTaxPriceService	_camerataxedprice_service	1,2	0,0	
1033	TheFinestSeedProviderService	_cerealgrain_PlantSeedservice	1,1	0,0	
1034	TeaCoffeeDifferencesService	_coffeeteareport_service	1,3	0,0	
1035	Comedy Film locator	_comedyfilmactionfilm_service	1,2	0,0	
1036	CF Films finder	_comedyfilmfantacyfilm_service	1,2	0,0	
1037	DiagP	_diagnosticprocessorganization_service	1,2	0,0	
1038	MM-DigitalAnologCameraPrice	_digitalstandardpriceprice_MediaMarktservice	1,4	0,0	
1039	DroughtReport	_droughtreport_service	1,2	0,0	
1040	Discovery Films	_filmDiscovery_service	1,1	0,0	1040
1041	Recommended Films	_filmHighlyRated_service	1,1	0,0	1040
1042	Discovery Channel	_filmvideomediaDiscoveryChannel_service	1,2	0,0	
1043	ImportFodderService	_fodder_USimportservice	1,1	0,0	
1044	ExportFoodService	_food_Exportservice	1,1	0,0	1044
1045	HEBFoodService	_food_HEBgroceryCompservice	1,1	0,0	1044
1046	FranceMapService	_Francemap_service	1,1	0,0	1046
1047	TheBestHoneyProvider	_honey_Providerservice	1,1	0,0	
1048	WorldwideHotelInfoService	_hotel_Worldwideservice	1,1	0,0	
1049	German Icing condition	_icing_Germanservice	1,1	0,0	
1050	InformationBearingProviderService	_information-bearing-object_Messageservice	1,1	0,0	
1051	EarthSystem	_internalchange_EarthSystemservice	1,1	0,0	
1052	SaarlandHospitalInvestService	_investigating_Saarservice	1,1	0,0	
1053	Computer Based Training (CBT)	_journal_Tutorialservice	1,1	0,0	
1054	MunchenUniversityLecturers	_lecturer-in-academiaMunchenUniversity_service	1,1	0,0	1054
1055	SaarlandUniversityLecturers	_lecturer-in-academiaSaarlandUniversity_service	1,1	0,0	1054
1056	ZambiaUniversityLecturers	_lecturer-in-academiaZambiaUniversity_service	1,1	0,0	1054
1057	FrankfurtBerlinMap	_mapFrankfurtBerlin_service	1,1	0,0	1046
1058	GermanMap	_mapGerman_service	1,1	0,0	1046
1059	UNOMedical Jobs	_medicaldoctor_UNOservice	1,1	0,0	
1060	myShop Person	_person_MyShopservice	1,1	0,0	
1061	GermanPollingService	_polling_Germanservice	1,1	0,0	
1062	WallmartCPriceService	_pricecamera_Wallmartservice	1,2	0,0	1030

Continued on next page

Table A.3 – continued from previous page

ID	Name	Filename (.owls)	IO	PR	Class
1063	CannonCameraPriceService	_price_CannonCameraservice	1,1	0,0	589
1064	CannonCameraPriceService	_price_Fishservice	1,1	0,0	589
1065	USprofessional career	_professiongeographical-region_USservice	1,2	0,0	
1066	Motion	_propositionmotion_service	1,2	0,0	
1067	PPService	_publication_PPservice	1,1	0,0	
1068	Red Ferrari price	_RedFerrariprice_service	1,1	0,0	589
1069	BMWSkilledPositions	_skilledoccupation_BMWservice	1,1	0,0	
1070	ToyotaCar Price	_Toyotaprice_service	1,1	0,0	589
1071	TranslocationService	_translocation_Chemicalservice	1,1	0,0	
1072	USJudge	_USjudge_service	1,1	0,0	
1073	BBC News Video Media	_videomediaBBC_service	1,1	0,0	1073
1074	SaturnVideoMediaService	_videomediaSaturn_service	1,1	0,0	1073
1075	Smith Lee productions	_videomediaSmithLee_service	1,1	0,0	1073
1076	Italy WarmFront	_warmfront_Italyservice	1,1	0,0	
1077	Bombay WeatherFront	_weatherfront_BombayIndiaservice	1,1	0,0	1077
1078	BritishWeatherfrontService	_weatherfront_Britishservice	1,1	0,0	1077
1079	German WeatherFront	_weatherfront_Germanyservice	1,1	0,0	1077
1080	GermanWeatherProcessService	_weatherprocess_Germanservice	1,1	0,0	
1081	MyDESTINATIONService	__destination_MyOfficeservice	2,1	0,0	921
1082	HeidelburgLuxuryHotelInfoService	__luxuryhotel_Heidelburgservice	1,1	0,0	

Table A.4.: IOR and IOPR matches in OWLS-TC

Row	ID 1	Service Name 1	ID 2	Service Name 2	Class ID	Match
1	0	4WheeledCar1PersonBicyclePrice	9	4WheeledCar 1PersonBicycle Price	0	IOPR
2	1	Kohl Car1PersonBicyclePrice	2	Car1PersonBicyclePrice	1	IOPR
3	1	Kohl Car1PersonBicyclePrice	3	Car1PersonBicyclePrice	1	IOPR
4	2	Car1PersonBicyclePrice	3	Car1PersonBicyclePrice	1	IOPR
5	4	Bicycle4Wheeledcar_Price_service	11	4WheeledCar 2PersonBicyclePrice	4	IOPR
6	12	4WheeledCarBicyclePrice	139	Bicycle4Wheeledcar_Price_service	12	IOPR
7	32	Academic Book Number booksearch	36	AcademicBookNumberSearch	32	IOPR
8	35	AcademicBookNumberOrISBNSearch	780	BookFinderService	35	IOPR
9	35	AcademicBookNumberOrISBNSearch	781	ABN/ISBN Booksearch	35	IOPR
10	100	Auto Bicycle Price	140	BicycleAuto_Price_service	100	IOPR
11	112	AvailablePreparedFoodService	545	GroceryStore PreparedFood Quantity	112	IOPR
12	112	AvailablePreparedFoodService	812	RequiredFoodService	112	IOPR
13	119	GermanGovernmentAwardScholarshipService	120	GermanGovernmentAwardScholarshipService	119	IOPR
14	142	BicycleCar_Price_service	192	CarBicyclePrice	142	IOPR
15	143	BookFinder	159	BookSearch	143	IOPR
16	148	Bea Book Shopping	149	Book Shopping	148	IOPR
17	153	BShop book shopping	154	Book Shopping	153	IOPR
18	155	BookPrice	156	BookPrice	155	IOR
19	158	BookPrice	167	Cheapest Book	158	IOPR
20	158	BookPrice	171	BookPriceService	158	IOPR
21	162	BookAuthorPriceService	163	BookAuthorPriceService	162	IOR
22	165	BookAuthorService	166	BookAuthorService	165	IOPR
23	167	Cheapest Book	171	BookPriceService	158	IOPR
24	175	BookRecommendedPriceService	994	Tizon RecommendedPriceService	175	IOPR
25	176	BookRecommendedPriceService	177	BookRecommendedPriceService	176	IOR
26	187	Distance calculator between two locations	188	Distance calculator between two locations	187	IOPR
27	187	Distance calculator between two locations	190	Distance calculator between two locations	187	IOPR
28	187	Distance calculator between two locations	476	Distance calculator between two locations	187	IOPR
29	188	Distance calculator between two locations	190	Distance calculator between two locations	187	IOPR
30	188	Distance calculator between two locations	476	Distance calculator between two locations	187	IOPR
31	190	Distance calculator between two locations	476	Distance calculator between two locations	187	IOPR
32	209	car price	594	leynthu rent a car	209	IOPR
33	237	CheckHospitalAvailability	238	CheckPersonnelAvailability	237	IOPR
34	237	CheckHospitalAvailability	239	CheckRoomAvailability	237	IOPR
35	238	CheckPersonnelAvailability	239	CheckRoomAvailability	237	IOPR
36	245	HotelInfoService	271	HotelInfoService	245	IOPR
37	250	GermanCityHotelInfoService	252	CityHotelInfoService	250	IOR
38	255	Zip codes finder service.	503	Zip codes finder service.	255	IOPR
39	259	Close door	602	Lock door	259	IOPR
40	259	Close door	724	Open door	259	IOPR
41	259	Close door	1004	Unlock door	259	IOPR
42	272	GelLuxuryHotelInfoService	273	LuxuryHotelInfoService	272	IOPR
43	278	OccupationFinder	408	OccupationFinder	278	IOPR
44	278	OccupationFinder	435	OccupationFinder	278	IOPR
45	281	OccupationFinder	411	OccupationFinder	281	IOPR
46	281	OccupationFinder	438	OccupationFinder	281	IOPR
47	294	OccupationFinder	421	OccupationFinder	294	IOPR
48	294	OccupationFinder	422	OccupationFinder	294	IOPR
49	294	OccupationFinder	451	OccupationFinder	294	IOPR

Continued on next page

Table A.4 – continued from previous page

Row	ID 1	Service Name 1	ID 2	Service Name 2	Class ID	Match
50	302	Country SkilledOccupation PartTimePosition	303	Country skilled and partTimePositionsService	302	IOPR
51	304	countryJobs	305	countryJobs	304	IOPR
52	304	countryJobs	332	DJob	304	IOR
53	305	countryJobs	332	DJob	304	IOR
54	307	CountrySkilledPositionsService	308	CountrySOccupService	307	IOPR
55	322	Government Degree Scholarship	515	GovernmentDegreeScholarShipService	322	IOPR
56	322	Government Degree Scholarship	516	GovernmentScholarDegreeShipService	322	IOPR
57	331	DeoSFNPrice	1010	SFNovelPrice	331	IOR
58	331	DeoSFNPrice	1011	SFNovelPrice	331	IOR
59	331	DeoSFNPrice	1012	SFNovelPrice	331	IOR
60	339	MD 2For 1 Price	340	2For 1 Price	339	IOPR
61	339	MD 2For 1 Price	341	2For 1 Price	339	IOPR
62	340	2For 1 Price	341	2For 1 Price	339	IOPR
63	344	2For 1 Price	670	2for 1 RecommendedPrice	344	IOPR
64	347	EBookOrder	348	EBookOrder2	347	IOPR
65	347	EBookOrder	349	EBookOrder3	347	IOPR
66	348	EBookOrder2	349	EBookOrder3	347	IOPR
67	356	Employee postal address	357	Employee postal address	356	IOPR
68	386	AldiFoodQuantityService	387	AldiFoodQuantityService	386	IOPR
69	390	B-co FoodPriceService	391	FoodPriceService	390	IOPR
70	398	SurfingGenericAgentDestinationService	399	SurfingGenericAgentDestination SportsService	398	IOPR
71	402	Gorge RouteFinder	403	RouteFinder	402	IOR
72	408	OccupationFinder	435	OccupationFinder	278	IOPR
73	411	OccupationFinder	438	OccupationFinder	281	IOPR
74	413	GeographicalRegionHotelService	414	GeographicalRegionHotelService	413	IOPR
75	417	Route from Frankfurt	418	Route to Berlin	417	IOPR
76	421	OccupationFinder	422	OccupationFinder	294	IOPR
77	421	OccupationFinder	451	OccupationFinder	294	IOPR
78	422	OccupationFinder	451	OccupationFinder	294	IOPR
79	424	GRWF	425	GRWF	424	IOPR
80	426	GRW	427	GRWeatherProcessService	426	IOPR
81	458	Geopolitical-Entity SkilledOccupation FullTimePosition	687	Geopolitical-Entity SkilledOccupation FullTimePosition	458	IOPR
82	459	Geopolitical-Entity SkilledOccupation PartTimePosition	688	Geopolitical-Entity SkilledOccupation PartTimePosition	459	IOPR
83	471	Altitude Above Sea Level Calculator Service.	479	Elevation Finder Service.	471	IOPR
84	515	GovernmentDegreeScholarShipService	516	GovernmentScholarDegreeShipService	322	IOPR
85	521	GovernmentMissileFundingService	522	GovernmentMissileFundingService	521	IOPR
86	527	GovernmentFundingService	528	GovernmentFundingService	527	IOPR
87	527	GovernmentFundingService	529	GovernmentFundingService	527	IOPR
88	527	GovernmentFundingService	530	GovernmentFundingService	527	IOPR
89	528	GovernmentFundingService	529	GovernmentFundingService	527	IOPR
90	528	GovernmentFundingService	530	GovernmentFundingService	527	IOPR
91	529	GovernmentFundingService	530	GovernmentFundingService	527	IOPR
92	535	Green Light To Off	536	Green Light To On	535	IOPR
93	545	GroceryStore PreparedFood Quantity	812	RequiredFoodService	112	IOPR
94	549	HDP2	569	Hospital DiagnosticProcess TimeDuration	549	IOPR
95	552	HigherEducationalOrganizationLecturers	553	HigherEducationalOrganizationLecturers	552	IOPR
96	554	Pioneer HigherEducationalOrganizationProfessors	555	HigherEducationalOrganizationProfessors	554	IOPR
97	556	HigherEducationalOrganizationSeniorResearcherFellow	557	HigherEducationalOrganizationSeniorResearcherFellow	556	IOPR
98	558	HikingSurfingCityService	910	SURFINGHIKINGCityService	558	IOPR
99	558	HikingSurfingCityService	911	SFCityService	558	IOPR
100	572	MedDiag	573	DiagnosticProcessAvailability	572	IOPR
101	589	KodakPriceService	1023	Three wheeled Car price	589	IOPR
102	589	KodakPriceService	1024	Another car price	589	IOPR
103	589	KodakPriceService	1063	CannonCameraPriceService	589	IOPR
104	589	KodakPriceService	1064	CannonCameraPriceService	589	IOPR
105	589	KodakPriceService	1068	Red Ferrari price	589	IOPR
106	589	KodakPriceService	1070	ToyotaCar Price	589	IOPR
107	598	RouteFinderService	599	SRI RouteFinderService	598	IOPR
108	602	Lock door	724	Open door	259	IOPR
109	602	Lock door	1004	Unlock door	259	IOPR
110	604	MAK	1022	ZAD	604	IOPR
111	609	BreadProviderService	610	BreadProviderService	609	IOPR
112	611	BestColaService	612	AvailableColaService	611	IOPR
113	613	DrinksService	614	DrinksService	613	IOPR
114	635	MedService Investigating	636	MedInvestigating	635	IOPR
115	652	GovMissilesGivingService	653	GovMissilesGivingService	652	IOPR
116	655	ChinaMissilesFinancingService	656	RussianMissilesFinancingService	655	IOPR
117	655	ChinaMissilesFinancingService	657	USMissilesFinancingService	655	IOPR
118	656	RussianMissilesFinancingService	657	USMissilesFinancingService	655	IOPR
119	658	AsianMissilesFundingService	659	IndiaMissilesFundingService	658	IOPR
120	658	AsianMissilesFundingService	660	NKoreaMissilesFundingService	658	IOPR
121	658	AsianMissilesFundingService	661	PakistanMissilesFundingService	658	IOPR
122	659	IndiaMissilesFundingService	660	NKoreaMissilesFundingService	658	IOPR
123	659	IndiaMissilesFundingService	661	PakistanMissilesFundingService	658	IOPR
124	660	NKoreaMissilesFundingService	661	PakistanMissilesFundingService	658	IOPR
125	713	NovelAuthorService	714	NovelAuthorService	713	IOPR

Continued on next page

Table A.4 – continued from previous page

Row	ID 1	Service Name 1	ID 2	Service Name 2	Class ID	Match
126	713	NovelAuthorService	715	NovelAuthorService	713	IOPR
127	714	NovelAuthorService	715	NovelAuthorService	713	IOPR
128	716	NovelPersonService	717	NovelPersonService	716	IOR
129	724	Open door	1004	Unlock door	259	IOPR
130	739	2for 1 RecommendedPrice	740	2for 1 RecommendedPrice	739	IOPR
131	755	DFKI	756	TheBest	755	IOPR
132	755	DFKI	763	PriceCoffeeWhiskey	755	IOPR
133	755	DFKI	764	PriceCoffeeWhiskey	755	IOPR
134	756	TheBest	763	PriceCoffeeWhiskey	755	IOPR
135	756	TheBest	764	PriceCoffeeWhiskey	755	IOPR
136	759	Hallo2 ColaService	760	Hallo ColaService	759	IOPR
137	763	PriceCoffeeWhiskey	764	PriceCoffeeWhiskey	755	IOPR
138	780	BookFinderService	781	ABN/ISBN Booksearch	35	IOPR
139	785	PublicationSearchingService	789	PublicationSearchingService	785	IOPR
140	802	RecommendedPriceCoffeeWhiskey	803	RecommendedPriceCoffeeWhiskey	802	IOPR
141	808	Red Light To Off	809	Red Light To On	808	IOPR
142	816	Researcher address	817	TREE Researcher address	816	IOPR
143	816	Researcher address	818	ZOO Researcher address	816	IOPR
144	817	TREE Researcher address	818	ZOO Researcher address	816	IOPR
145	819	Researcher postal address and publication references	820	Researcher postal address and publication references	819	IOPR
146	822	HOM2 Researcher address	823	Researcher address	822	IOPR
147	848	Science-Fiction-Novel Author Price	851	ScienceFictionNovel priceAuthor	848	IOPR
148	863	Search Formatted Address	864	Search Raw address	863	IOPR
149	906	MerchantService	907	StorePFoodService	906	IOPR
150	910	SURFINGHIKINGCityService	911	SFCityService	558	IOPR
151	912	DFG HikingSurfingDestination	913	PDS HikingSurfingDestination	912	IOPR
152	912	DFG HikingSurfingDestination	914	PF Destination	912	IOPR
153	912	DFG HikingSurfingDestination	915	HikingSurfingDestination	912	IOPR
154	913	PDS HikingSurfingDestination	914	PF Destination	912	IOPR
155	913	PDS HikingSurfingDestination	915	HikingSurfingDestination	912	IOPR
156	914	PF Destination	915	HikingSurfingDestination	912	IOPR
157	921	SurfingOrganizationDESTINATIONService	922	SAAR	921	IOPR
158	921	SurfingOrganizationDESTINATIONService	923	SurfingOrganizationDESTINATIONService	921	IOPR
159	921	SurfingOrganizationDESTINATIONService	924	SODService	921	IOPR
160	921	SurfingOrganizationDESTINATIONService	1081	MyDESTINATIONService	921	IOPR
161	922	SAAR	923	SurfingOrganizationDESTINATIONService	921	IOPR
162	922	SAAR	924	SODService	921	IOPR
163	922	SAAR	1081	MyDESTINATIONService	921	IOPR
164	923	SurfingOrganizationDESTINATIONService	924	SODService	921	IOPR
165	923	SurfingOrganizationDESTINATIONService	1081	MyDESTINATIONService	921	IOPR
166	924	SODService	1081	MyDESTINATIONService	921	IOPR
167	926	Always SurfingDestinationService	927	AUS SurfingDestinationService	926	IOPR
168	926	Always SurfingDestinationService	928	SurfingDestinationService	926	IOPR
169	926	Always SurfingDestinationService	929	SOH SurfingDestinationService	926	IOPR
170	927	AUS SurfingDestinationService	928	SurfingDestinationService	926	IOPR
171	927	AUS SurfingDestinationService	929	SOH SurfingDestinationService	926	IOPR
172	928	SurfingDestinationService	929	SOH SurfingDestinationService	926	IOPR
173	933	Switch Off MesseModul	934	Switch On MesseModul	933	IOPR
174	958	BF Comedy Film Searcher	959	Mega Comedy Film finder	958	IOPR
175	958	BF Comedy Film Searcher	960	Comedy Film finder	958	IOPR
176	959	Mega Comedy Film finder	960	Comedy Film finder	958	IOPR
177	961	HighComedy Action Film	963	Film locator	961	IOPR
178	961	HighComedy Action Film	968	Film finder	961	IOPR
179	963	Film locator	968	Film finder	961	IOPR
180	987	MMVideoMediaService	993	AvailableVideoService	987	IOPR
181	997	UniversityLecturerService	998	UniversityLecturerService	997	IOPR
182	997	UniversityLecturerService	999	UniversityLecturerService	997	IOPR
183	998	UniversityLecturerService	999	UniversityLecturerService	997	IOPR
184	1010	SFNovelPrice	1011	SFNovelPrice	331	IOPR
185	1010	SFNovelPrice	1012	SFNovelPrice	331	IOPR
186	1011	SFNovelPrice	1012	SFNovelPrice	331	IOPR
187	1023	Three wheeled Car price	1024	Another car price	589	IOPR
188	1023	Three wheeled Car price	1063	CannonCameraPriceService	589	IOPR
189	1023	Three wheeled Car price	1064	CannonCameraPriceService	589	IOPR
190	1023	Three wheeled Car price	1068	Red Ferrari price	589	IOPR
191	1023	Three wheeled Car price	1070	ToyotaCar Price	589	IOPR
192	1024	Another car price	1063	CannonCameraPriceService	589	IOPR
193	1024	Another car price	1064	CannonCameraPriceService	589	IOPR
194	1024	Another car price	1068	Red Ferrari price	589	IOPR
195	1024	Another car price	1070	ToyotaCar Price	589	IOPR
196	1026	CompJ AuthorService	1027	DataMiningAuthorService	1026	IOPR
197	1030	Camera Price	1062	WallmartCPriceService	1030	IOPR
198	1040	Discovery Films	1041	Recommended Films	1040	IOPR
199	1044	ExportFoodService	1045	HEBFoodService	1044	IOPR
200	1046	FranceMapService	1057	FrankfurtBerlinMap	1046	IOPR
201	1046	FranceMapService	1058	GermanMap	1046	IOPR

Continued on next page

Table A.4 – continued from previous page

Row	ID 1	Service Name 1	ID 2	Service Name 2	Class ID	Match
202	1054	MunchenUniversityLecturers	1055	SaarlandUniversityLecturers	1054	IOPR
203	1054	MunchenUniversityLecturers	1056	ZambiaUniversityLecturers	1054	IOPR
204	1055	SaarlandUniversityLecturers	1056	ZambiaUniversityLecturers	1054	IOPR
205	1057	FrankfurtBerlinMap	1058	GermanMap	1046	IOPR
206	1063	CannonCameraPriceService	1064	CannonCameraPriceService	589	IOPR
207	1063	CannonCameraPriceService	1068	Red Ferrari price	589	IOPR
208	1063	CannonCameraPriceService	1070	ToyotaCar Price	589	IOPR
209	1064	CannonCameraPriceService	1068	Red Ferrari price	589	IOPR
210	1064	CannonCameraPriceService	1070	ToyotaCar Price	589	IOPR
211	1068	Red Ferrari price	1070	ToyotaCar Price	589	IOPR
212	1073	BBC News Video Media	1074	SaturnVideoMediaService	1073	IOPR
213	1073	BBC News Video Media	1075	Smith Lee productions	1073	IOPR
214	1074	SaturnVideoMediaService	1075	Smith Lee productions	1073	IOPR
215	1077	Bombay WeatherFront	1078	BritishWeatherfrontService	1077	IOPR
216	1077	Bombay WeatherFront	1079	German WeatherFront	1077	IOPR
217	1078	BritishWeatherfrontService	1079	German WeatherFront	1077	IOPR

Table A.5.: IOR matches in OWLS-TC which their precondition do not match

Row	Row in Table A.4	ID 1	Service Name 1	ID 2	Service Name 2	Class ID
1	18	155	BookPrice	156	BookPrice	155
2	21	162	BookAuthorPriceService	163	BookAuthorPriceService	162
3	25	176	BookRecommendedPriceService	177	BookRecommendedPriceService	176
4	37	250	GermanCityHotelInfoService	252	CityHotelInfoService	250
5	52	304	countryJobs	332	DJob	304
6	53	305	countryJobs	332	DJob	304
7	57	331	DeoSFNPrice	1010	SFNovelPrice	331
8	58	331	DeoSFNPrice	1011	SFNovelPrice	331
9	59	331	DeoSFNPrice	1012	SFNovelPrice	331
10	71	402	Gorge RouteFinder	403	RouteFinder	402
11	128	716	NovelPersonService	717	NovelPersonService	716

Table A.6.: Service classes in OWLS-TC

Service Class ID	Service ID	Service Name
0	0	4WheeledCar1PersonBicyclePrice
	9	4WheeledCar 1PersonBicycle Price
1	1	Kohl Car1PersonBicyclePrice
	2	Car1PersonBicyclePrice
	3	Car1PersonBicyclePrice
4	4	Bicycle4Wheeledcar_Price_service
	11	4WheeledCar 2PersonBicyclePrice
12	12	4WheeledCarBicyclePrice
	139	Bicycle4Wheeledcar_Price_service
32	32	Academic Book Number booksearch
	36	AcademicBookNumberSearch
35	35	AcademicBookNumberOrISBNSearch
	780	BookFinderService
	781	ABN/ISBN Booksearch
100	100	Auto Bicycle Price
	140	BicycleAuto_Price_service
112	112	AvailablePreparedFoodService
	545	GroceryStore PreparedFood Quantity
	812	RequiredFoodService
119	119	GermanGovernmentAwardScholarshipService
	120	GermanGovernmentAwardScholarshipService
142	142	BicycleCar_Price_service
	192	CarBicyclePrice
143	143	BookFinder
	159	BookSearch
148	148	Bea Book Shopping
	149	Book Shopping
153	153	BShop book shopping
	154	Book Shopping
155	155	BookPrice
	156	BookPrice
158	158	BookPrice
	167	Cheapest Book
	171	BookPriceService
162	162	BookAuthorPriceService
	163	BookAuthorPriceService

Continued on next page

Table A.6 – continued from previous page

Service Class ID	Service ID	Service Name
165	165	BookAuthorService
	166	BookAuthorService
175	175	BookRecommendedPriceService
	994	Tizon RecommendedPriceService
176	176	BookRecommendedPriceService
	177	BookRecommendedPriceService
187	187	Distance calculator between two locations
	188	Distance calculator between two locations
	190	Distance calculator between two locations
	476	Distance calculator between two locations
209	209	car price
	594	leynthu rent a car
237	237	CheckHospitalAvailability
	238	CheckPersonnelAvailability
	239	CheckRoomAvailability
245	245	HotelInfoService
	271	HotelInfoService
250	250	GermanCityHotelInfoService
	252	CityHotelInfoService
255	255	Zip codes finder service.
	503	Zip codes finder service.
259	259	Close door
	602	Lock door
	724	Open door
	1004	Unlock door
272	272	GelLuxuryHotelInfoService
	273	LuxuryHotelInfoService
278	278	OccupationFinder
	408	OccupationFinder
	435	OccupationFinder
281	281	OccupationFinder
	411	OccupationFinder
	438	OccupationFinder
294	294	OccupationFinder
	421	OccupationFinder
	422	OccupationFinder
	451	OccupationFinder
302	302	Country SkilledOccupation PartTimePosition
	303	Country skilled and partTimePositionsService
304	304	countryJobs
	305	countryJobs
	332	DJob
307	307	CountrySkilledPositionsService
	308	CountrySOccupService
322	322	Government Degree Scholarship
	515	GovernmentDegreeScholarShipService
	516	GovernmentScholarDegreeShipService
331	331	DeoSFNPrice
	1010	SFNovelPrice
	1011	SFNovelPrice
	1012	SFNovelPrice
339	339	MD 2For 1 Price
	340	2For 1 Price
	341	2For 1 Price
344	344	2For 1 Price
	670	2for 1 RecommendedPrice
347	347	EBookOrder
	348	EBookOrder2
	349	EBookOrder3
356	356	Employee postal address
	357	Employee postal address
386	386	AldiFoodQuantityService
	387	AldiFoodQuantityService
390	390	B-co FoodPriceService
	391	FoodPriceService
398	398	SurfingGenericAgentDestinationService
	399	SurfingGenericAgentDestination SportsService
402	402	Gorge RouteFinder
	403	RouteFinder
413	413	GeographicalRegionHotelService
	414	GeographicalRegionHotelService
417	417	Route from Frankfurt
	418	Route to Berlin
424	424	GRWF
	425	GRWF
426	426	GRW
	427	GRWeatherProcessService
458	458	Geopolitical-Entity SkilledOccupation FullTimePosition

Continued on next page

Table A.6 – continued from previous page

Service Class ID	Service ID	Service Name
	687	Geopolitical-Entity SkilledOccupation FullTimePosition
459	459	Geopolitical-Entity SkilledOccupation PartTimePosition
	688	Geopolitical-Entity SkilledOccupation PartTimePosition
471	471	Altitude Above Sea Level Calculator Service.
	479	Elevation Finder Service.
521	521	GovernmentMissileFundingService
	522	GovernmentMissileFundingService
527	527	GovernmentFundingService
	528	GovernmentFundingService
	529	GovernmentFundingService
	530	GovernmentFundingService
535	535	Green Light To Off
	536	Green Light To On
549	549	HDP2
	569	Hospital DiagnosticProcess TimeDuration
552	552	HigherEducationalOrganizationLecturers
	553	HigherEducationalOrganizationLecturers
554	554	Pioneer HigherEducationalOrganizationProfessors
	555	HigherEducationalOrganizationProfessors
556	556	HigherEducationalOrganizationSeniorResearcherFellow
	557	HigherEducationalOrganizationSeniorResearcherFellow
558	558	HikingSurfingCityService
	910	SURFINGHIKINGCityService
	911	SFCityService
572	572	MedDiag
	573	DiagnosticProcessAvailability
589	589	KodakPriceService
	1023	Three wheeled Car price
	1024	Another car price
	1063	CannonCameraPriceService
	1064	CannonCameraPriceService
	1068	Red Ferrari price
	1070	ToyotaCar Price
598	598	RouteFinderService
	599	SRI RouteFinderService
604	604	MAK
	1022	ZAD
609	609	BreadProviderService
	610	BreadProviderService
611	611	BestColaService
	612	AvailableColaService
613	613	DrinksService
	614	DrinksService
635	635	MedService Investigating
	636	MedInvestigating
652	652	GovMissilesGivingService
	653	GovMissilesGivingService
655	655	ChinaMissilesFinancingService
	656	RussianMissilesFinancingService
	657	USMissilesFinancingService
658	658	AsianMissilesFundingService
	659	IndiaMissilesFundingService
	660	NKoreaMissilesFundingService
	661	PakistanMissilesFundingService
713	713	NovelAuthorService
	714	NovelAuthorService
	715	NovelAuthorService
716	716	NovelPersonService
	717	NovelPersonService
739	739	2for 1 RecommendedPrice
	740	2for 1 RecommendedPrice
755	755	DFKI
	756	TheBest
	763	PriceCoffeeWhiskey
	764	PriceCoffeeWhiskey
759	759	Hallo2 ColaService
	760	Hallo ColaService
785	785	PublicationSearchingService
	789	PublicationSearchingService
802	802	RecommendedPriceCoffeeWhiskey
	803	RecommendedPriceCoffeeWhiskey
808	808	Red Light To Off
	809	Red Light To On
816	816	Researcher address
	817	TREE Researcher address
	818	ZOO Researcher address
819	819	Researcher postal address and publication references
	820	Researcher postal address and publication references
822	822	HOM2 Researcher address

Continued on next page

Table A.6 – continued from previous page

Service Class ID	Service ID	Service Name
	823	Researcher address
848	848	Science-Fiction-Novel Author Price
	851	ScienceFictionNovel priceAuthor
863	863	Search Formatted Address
	864	Search Raw address
906	906	MerchantService
	907	StorePFoodService
912	912	DFG HikingSurfingDestination
	913	PDS HikingSurfingDestination
	914	PF Destination
	915	HikingSurfingDestination
921	921	SurfingOrganizationDESTINATIONService
	922	SAAR
	923	SurfingOrganizationDESTINATIONService
	924	SODService
	1081	MyDESTINATIONService
926	926	Always SurfingDestinationService
	927	AUS SurfingDestinationService
	928	SurfingDestinationService
	929	SOH SurfingDestinationService
933	933	Switch Off MesseModul
	934	Switch On MesseModul
958	958	BF Comedy Film Searcher
	959	Mega Comedy Film finder
	960	Comedy Film finder
961	961	HighComedy Action Film
	963	Film locator
	968	Film finder
987	987	MMVideoMediaService
	993	AvailableVideoService
997	997	UniversityLecturerService
	998	UniversityLecturerService
	999	UniversityLecturerService
1026	1026	CompJ AuthorService
	1027	DataMiningAuthorService
1030	1030	Camera Price
	1062	WallmartCPriceService
1040	1040	Discovery Films
	1041	Recommended Films
1044	1044	ExportFoodService
	1045	HEBFoodService
1046	1046	FranceMapService
	1057	FrankfurtBerlinMap
	1058	GermanMap
1054	1054	MunchenUniversityLecturers
	1055	SaarlandUniversityLecturers
	1056	ZambiaUniversityLecturers
1073	1073	BBC News Video Media
	1074	SaturnVideoMediaService
	1075	Smith Lee productions
1077	1077	Bombay WeatherFront
	1078	BritishWeatherfrontService
	1079	German WeatherFront

Appendix B: Publications

Journal Papers:

- **Saboohi, Hadi** & Abdul Kareem, Sameem (2013). An automatic subdigraph renovation plan for failure recovery of composite semantic Web services. Frontiers of Computer Science. Springer. Accepted.
- **Saboohi, Hadi** & Abdul Kareem, Sameem (2012). Requirements of a recovery solution for failure of composite web services. International Journal of Web & Semantic Technology (IJWeST), 3(4), 15-21.

Conference Papers:

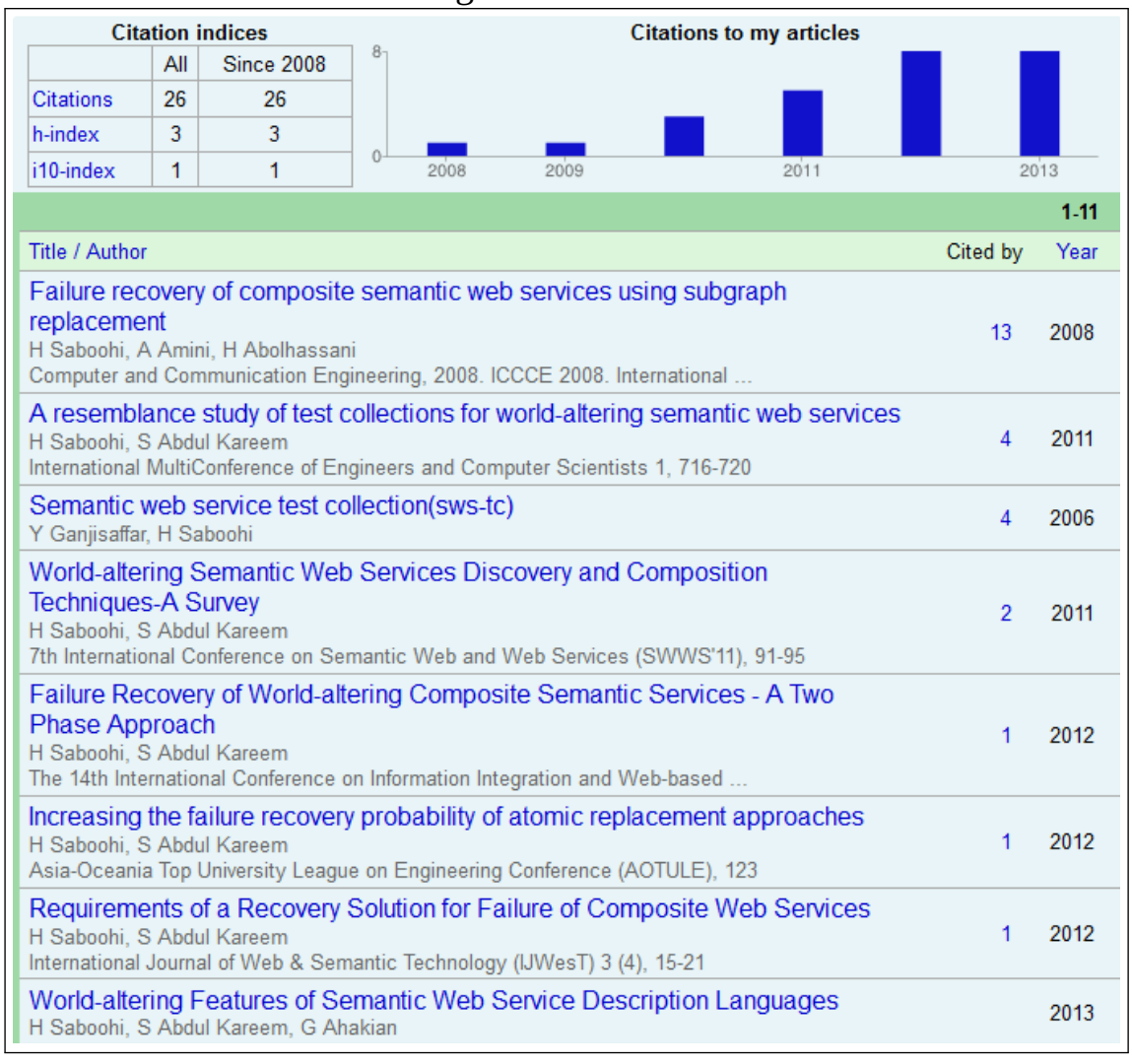
- **Saboohi, Hadi**, Abdul Kareem, Sameem, & Ahakian, Gholamreza (2013). World-altering features of semantic web service description languages. In The Second International Conference on e-Technologies and Networks for Development (ICeND) (pp. 132-136). Kuala Lumpur, Malaysia.
- **Saboohi, Hadi** & Abdul Kareem, Sameem (2012b). Failure recovery of world-altering composite semantic services - a two phase approach. In 14th International Conference on Information Integration and Web-based Applications & Services (iiWAS) (pp. 299-302). Bali, Indonesia: ACM.
- **Saboohi, Hadi** & Abdul Kareem, Sameem (2012a). Increasing the failure recovery probability of atomic replacement approaches. In Asia-Oceania Top University League on Engineering Student Conference (AOTULE) (pp. 123). Kuala Lumpur, Malaysia.
- **Saboohi, Hadi** & Abdul Kareem, Sameem (2011b). World-altering semantic web services discovery and composition techniques - a survey. In The 7th International Conference on Semantic Web and Web Services (SWWS) (pp. 91-95). Las Vegas, USA.
- **Saboohi, Hadi** & Abdul Kareem, Sameem (2011a). A resemblance study of test collections for world-altering semantic web services. In International Conference on Internet Computing

and Web Services (ICICWS) in The International MultiConference of Engineers and Computer Scientists (IMECS), (pp. 716-720). Hong Kong.

- **Saboohi, Hadi**, Amini, Amineh, & Abolhassani, Hassan (2008). Failure recovery of composite semantic web services using subgraph replacement. In International Conference on Computer and Communication Engineering (ICCCE) (pp. 489-493). Kuala Lumpur, Malaysia.
- Ganjisaffar, Yasser & **Saboohi, Hadi** (2006). Semantic web services' test collection SWS-TC. Available online at:

  http://www.semwebcentral.org/projects/sws-tc/

Google Scholar Profile



Citation indices

	All	Since 2008
Citations	26	26
h-index	3	3
i10-index	1	1

Citations to my articles

1-11

Title / Author	Cited by	Year
Failure recovery of composite semantic web services using subgraph replacement H Saboohi, A Amini, H Abolhassani Computer and Communication Engineering, 2008. ICCCE 2008. International ...	13	2008
A resemblance study of test collections for world-altering semantic web services H Saboohi, S Abdul Kareem International MultiConference of Engineers and Computer Scientists 1, 716-720	4	2011
Semantic web service test collection(sws-tc) Y Ganjisaffar, H Saboohi	4	2006
World-altering Semantic Web Services Discovery and Composition Techniques-A Survey H Saboohi, S Abdul Kareem 7th International Conference on Semantic Web and Web Services (SWWS'11), 91-95	2	2011
Failure Recovery of World-altering Composite Semantic Services - A Two Phase Approach H Saboohi, S Abdul Kareem The 14th International Conference on Information Integration and Web-based ...	1	2012
Increasing the failure recovery probability of atomic replacement approaches H Saboohi, S Abdul Kareem Asia-Oceania Top University League on Engineering Conference (AOTULE), 123	1	2012
Requirements of a Recovery Solution for Failure of Composite Web Services H Saboohi, S Abdul Kareem International Journal of Web & Semantic Technology (IJWesT) 3 (4), 15-21	1	2012
World-altering Features of Semantic Web Service Description Languages H Saboohi, S Abdul Kareem, G Ahakian		2013

http://www.hadisaboohi.com/