

# **METRICS FOR EVOLVING SERVICES IN SERVICE ORIENTED ARCHITECTURE**

*Thesis submitted to Jawaharlal Nehru University  
in partial fulfillment of the requirement  
for the award of the degree of*

**DOCTOR OF PHILOSOPHY**

**in**

**COMPUTER SCIENCE**

**by**

**RACHNA KOHAR**

Under the Supervision of

**Prof. Parimala N.**



**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI- 110067,  
INDIA**

**July 2017**



SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
जवाहरलाल नेहरू विश्वविद्यालय  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI- 110067

---

### DECLARATION

This is to certify that the thesis entitled “**Metrics for Evolving Services in Service Oriented Architecture**” which is being submitted to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for the award of the degree of **Doctor of Philosophy** in Computer Science, is a bonafide research work carried out by me under the supervision of **Prof. Parimala N.**

This research work is original and has not been submitted, in part or full, to any other university or institution for the award of any other degree.

*Rachna*

**Rachna Kohar**

Enrollment No.: 12/10/PC/002

School of Computer and Systems Sciences,

Jawaharlal Nehru University,

New Delhi - 110067



SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
जवाहरलाल नेहरू विश्वविद्यालय  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI- 110067

**CERTIFICATE**

This is to certify that the thesis entitled “**Metrics for Evolving Services in Service Oriented Architecture**” submitted by **Rachna Kohar** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for the award of the degree of **Doctor of Philosophy** in Computer Science, is a bonafide research work carried out under the supervision of **Prof. Parimala N.**

This research work is original and has not been submitted, in part or full, to any other university or institution for the award of any other degree.

*D.K. Lobiyal*  
24/7/17  
**Prof. D.K. Lobiyal**  
(Dean)

School of Computer & Systems Sciences  
Jawaharlal Nehru University  
New Delhi - 110067

Dean  
School of Computer & Systems Sciences  
Jawaharlal Nehru University  
New Delhi-110067

*Parimala N.*  
24/7/2017  
**Prof. Parimala N.**  
(Supervisor)

School of Computer & Systems Sciences  
Jawaharlal Nehru University  
New Delhi - 110067  
Professor

School of Computer & Systems Science  
Jawaharlal Nehru University  
New Delhi - 110 067 (India)

## ACKNOWLEDGEMENT

First, I would like to express my profound gratitude to the Almighty whose grace provided me the strength to complete my Ph.D. thesis.

I want to express my sincere and heartiest thanks to my supervisor and mentor Prof. Parimala N. for her excellent guidance and encouragement throughout my Ph.D. research work. Her keen insight, enlightening discussions, expertise and moral support enabled me to accomplish my research work.

I would like to express my sincere gratitude and thanks to Dean, Prof. D. K. Lobiyal and Ex-Dean, Prof. R.K. Agrawal for providing me all the necessary facilities and an ideal environment for doing my research. I would also like to thank all the staff members of the school for their prompt assistance throughout my research.

I would like to give special thanks to my family for their continuous support. Words cannot express how grateful I am to all my family members for all the sacrifices they have made on my behalf. My beloved family has always been my support in the moments when there was no one to answer my queries. Their prayers have sustained me so far.

I would also like to thank all of my friends for their support.



**Rachna Kohar**

Enrollment No.: 12/10/PC/002

School of Computer and Systems Sciences,  
Jawaharlal Nehru University,

New Delhi - 110067

## **ABSTRACT**

Service Oriented Architecture (SOA) is an architecture which uses a service as the fundamental element for developing applications. A service has a very high reuse capability to be reused in other applications. In SOA, there is a service provider who describes the service and publishes it in a central repository from where service consumer can invoke the service. The main aim of a service in SOA is to share application logic across different systems having different operating systems and development environments.

Services are aggregated to form a composite service in two different ways. The first is orchestration in which there is a central coordinator to manage the flow among services. Orchestration is achieved via Web Service Business Process Execution Language (WS-BPEL). Composite service developed using WS-BPEL is also known as WS-BPEL process. The second is choreography in which there is no central coordinator over the services in service composition. Instead, there is a global coordination between the services. Choreography is achieved via Web service choreography description language (WS-CDL). Composite service developed using WS-CDL is also known as WS-CDL process.

Services whether single or composite evolve over time. The evolution occurs due to the changing demands of the market and due to the enhancements which the service provider wants to incorporate to improve its performance. This evolution may affect the service provider as well as the service consumer. Evolution comprises of changes such as additions, deletions, modifications etc. Different changes may affect the provider and the consumer in different ways. A provider may want to track different phases of service evolution to identify the nature of changes and the amount by which changes have taken place. A consumer may be interested in knowing whether the usefulness of a service has changed for her/ him during service evolution. Therefore, in this thesis, we propose metrics to measure evolution in services and the manner in which this evolution affects the provider and the consumer. Metrics are proposed for both a single service as well as a composite service.

First, consider the case of a single service. A service provider may want to know how much a service has evolved across versions. This quantitative measure is provided by Service Evolution Metric (SEM). A service is invoked through service client code. When a service evolves, the client code may have to adapt to these changes. Some may be mandatory and some may be optional to incorporate. This impact of evolution in client code is measured by Service Client-code Evolution Metrics (SCEM<sub>M</sub>, SCEM<sub>O</sub> and SCEM<sub>T</sub>). For the service consumer, the usefulness of the service may increase or decrease as the service evolves. The impact on the usefulness during evolution is measured by Service Usefulness Evolution Metric (SUEM).

Now, consider the composite service. As discussed above, there are two ways in which services are composed. Metrics are proposed for each of these.

Consider, first, composition through orchestration. A WS-BPEL process is an executable process which is consumed by its consumer. Thus, metrics are proposed for both the provider as well as the consumer of the process. A process interacts with many partner (external) services in order to achieve the desired business functionalities for its consumer. Also, it has some of its internal logic to coordinate between these services. So, when it evolves, there may be external and/or internal evolution. The nature and quantum of evolution is computed by the proposed metrics. Two BPEL Evolution Metrics are proposed. One is for the external evolution (BEM<sub>E</sub>) and the other is for the internal evolution (BEM<sub>I</sub>). For a service consumer, BPEL Process Usefulness Metric under Evolution in a positive sense (BUME<sub>P</sub>) and BPEL Process Usefulness Metric under Evolution in a negative sense (BUME<sub>N</sub>) measures the impact of process evolution on the usefulness for the consumer.

The second method of composing services is through choreography. A WS-CDL process is not an executable process. It is used to specify interactions between the involved participants in the choreography. Therefore, the perspective of the provider is considered while proposing metrics. During evolution, new participants may be added with new interactions and new roles or old participants may get deleted along with their interactions with the existing ones. In this way, there are some changes

which may account to increase the number of entities in the choreography i.e. they are additive in nature and some changes may decrease the number of entities in the existing choreography i.e. they are subtractive in nature. Therefore, the provider may want to know which kinds of changes are made in the evolving choreography. The proposed metrics to measure these changes are Additive Evolution Metric (AEM<sup>+</sup>) and Subtractive Evolution Metric (SEM). Moreover, the provider may also want to know the overall amount of process evolution. This is measured using Evolution Metric (EM).

All the proposed metrics in this thesis have been theoretically validated using Zuse framework. All metrics are found to be above the ordinal scale. Empirical validation of all the metrics is done using real time data wherever the data is available and with simulated data wherever the data is not available.

To sum up, in this thesis we have proposed metrics for measuring evolution of services. For a single service, metrics are proposed for measuring the impact on the service client code and usefulness during evolution. Metrics are also proposed to measure evolution and its impact on the usefulness of a WS-BPEL process across its different versions. The proposed metrics for the WS-CDL process measure its evolution in terms of the nature of changes.

## LIST OF PUBLICATIONS

1. Rachna Kohar and Parimala N., “A Metrics Framework for Measuring Quality of a Web Service as it Evolves”, International Journal of System Assurance Engineering and Management, Springer, DOI: 10.1007/s13198-017-0591-y011, pp. 1-15, Feb. 2017, ISSN No: 0976-4348 (SCOPUS, Emerging Sources Citation Index)
2. Parimala N. and Rachna Kohar, "A Quality Metric for BPEL Process under Evolution", Eleventh International Conference on Digital Information Management (ICDIM-2016), pp.197-202, 19-21 Sept. 2016. ISBN No: 978-1-5090-2641-8 (SCOPUS)
3. Parimala N. and Rachna Kohar, “Evolution Metrics for a BPEL Process”, Second International Conference on Intelligent Computing & Communication, ICICC-2017, (Accepted), To appear in Advances in Intelligent and Soft Computing (AISC) Series, Springer.
4. Rachna Kohar and Parimala N., “A Metrics framework for a WS-CDL Process under Evolution” (Communicated, 2017).



## LIST OF FIGURES

Figure 1.1 Basic entities of service oriented architecture.....	3
Figure 1.2 Single service evolution.....	6
Figure 1.3 Composite service evolution - Orchestration.....	7
Figure 1.4 Composite service evolution - Choreography.....	9
Figure 3.1 WSDL 2.0 Infoset.....	25
Figure 3.2 Graph showing correlation of SEM & SCEM <sub>M</sub> and SEM & SUEM.....	39
Figure 3.3 Graph showing correlation of SEM & SCEM <sub>O</sub> and SEM & SUEM.....	39
Figure 3.4 Graph showing correlation of SEM & SCEM <sub>T</sub> and SEM & SUEM.....	39
Figure 5.1 WS-CDL Process Version 1.....	79
Figure 5.2 WS-CDL Process Version 2.....	80
Figure 5.3 WS-CDL Process Version 3.....	81
Figure 5.4 WS-CDL Process Version 4.....	82
Figure 5.5 WS-CDL Process Version 5.....	84
Figure 6.1 MCS Architecture.....	92
Figure 6.2 User Interface to initiate interaction with MCS.....	94
Figure 6.3 User Interface of MCS for a single service.....	95
Figure 6.4 User Interface of MCS for a single service - real world.....	96
Figure 6.5 Computed metrics for a single service - real world.....	97
Figure 6.6 User Interface of MCS for a single service - simulated.....	98
Figure 6.7 Computed metrics for a single service - simulated.....	99
Figure 6.8 User Interface of MCS for a composite service.....	100

Figure 6.9 User Interface of MCS for a composite service - Orchestration.....	101
Figure 6.10 Computed metrics for a composite service - Orchestration.....	102
Figure 6.11 User Interface of MCS for a composite service - Choreography.....	103
Figure 6.12 Computed metrics for a composite service - Choreography.....	104

## LIST OF TABLES

Table 3.1 WSDL 2.0 description.....	25
Table 3.2 Categories for changes in Service Client Code.....	29
Table 3.3 Categories for changes for Usefulness of service.....	32
Table 3.4 Metrics for the real-world service.....	35
Table 3.5 Metrics for the simulated service.....	36
Table 3.6 Correlation among metrics.....	38
Table 3.7 Table showing evolution data for operation element.....	41
Table 3.8 Summarized Zuse Framework.....	42
Table 3.9 Summary of formal validation of metrics of a single service.....	46
Table 4.1 WS-BPEL process activities.....	49
Table 4.2 Category of changes for a WS-BPEL process.....	50
Table 4.3 Category of changes for usefulness of a WS-BPEL process.....	53
Table 4.4 Basic Activities for Favorable Changes.....	54
Table 4.5 Basic Activities for Unfavorable Changes.....	57
Table 4.6 Description of the changes in the service and process.....	61
Table 4.7 Metrics for the Airline service and the Travel booking process.....	61
Table 4.8 Metric values for a WS-BPEL process.....	62
Table 4.9 Table showing changes for invoke element.....	64
Table 4.10 Summary of formal validation of metrics for WS-BPEL process.....	68
Table 5.1 WS-CDL process entities.....	70

Table 5.2 Evolution description of version 1 to version 2 of the process.....	80
Table 5.3 Evolution description of version 2 to version 3 of the process.....	82
Table 5.4 Evolution description of version 3 to version 4 of the process.....	83
Table 5.5: Evolution description of version 4 to version 5 of the process.....	85
Table 5.6 Metric values of the WS-CDL process.....	85
Table 5.7 Summary of formal validation of metrics of a WS-CDL process.....	90

## **LIST OF ABBREVIATIONS**

SOA - Service Oriented Architecture

WSDL - Web Service Description Language

SOAP - Simple Object Access Protocol

WS-BPEL - Web Service - Business Process Execution Language

WS-CDL - Web Service - Choreography Description Language

SEM - Service Evolution Metric

SCEM - Service Client-code Evolution Metric

SUEM - Service Usefulness Evolution Metric

BEM - BPEL Evolution Metric

BUME - BPEL Process Usefulness Metric under Evolution

AEM<sup>+</sup> - Additive Evolution Metric

SEM<sup>-</sup> - Subtractive Evolution Metric

EM - Evolution Metric

ED - Evolution Data

CS - Compute and Store

MC - Metrics Computation

## TABLE OF CONTENTS

DECLARATION.....	ii
CERTIFICATE.....	iii
ACKNOWLEDGEMENT.....	iv
ABSTRACT.....	v
LIST OF PUBLICATIONS.....	viii
LIST OF FIGURES.....	ix
LIST OF TABLES.....	xi
LIST OF ABBREVIATIONS.....	xiii
Chapter 1 Introduction.....	1
1.1 Service Oriented Architecture (SOA).....	2
1.2 Evolution in SOA.....	4
1.2.1 Single Service.....	4
1.2.2 Composite service - Orchestration.....	5
1.2.3 Composite Service - Choreography.....	5
1.3 Problem Statement and Motivation.....	5
1.3.1 Single Service.....	5
1.3.2 Composite Service.....	7
1.3.2.1 Composite Service - Orchestration.....	7
1.3.2.2 Composite Service - Choreography.....	8
1.4 Thesis Contribution.....	10
1.4.1 Single Service.....	10
1.4.2 Composite Service - Orchestration.....	10
1.4.3 Composite Service- Choreography.....	11
Chapter 2 State of the Art.....	13
2.1 Introduction.....	13
2.2 Metrics.....	14
2.3 Metrics for a single service.....	15
2.4 Metrics for a composite service.....	19
2.5 Summary.....	22
Chapter 3 Metrics for an Evolving Single Service.....	23
3.1 Introduction.....	23
3.2 Service Interface.....	24
3.3 Proposed Metrics for a Single Service.....	26
3.3.1 Service Evolution Metric (SEM).....	26
3.3.2 Service Client- code Evolution Metric (SCEM).....	28
3.3.3 Service Usefulness Evolution Metric (SUEM).....	31
3.4 Experiments and Analysis.....	34
3.5 Time Complexity.....	40
3.6 Metrics Formal Validation.....	41
3.7 Summary.....	46

Chapter 4 Metrics for an Evolving Composite Service – Orchestration .....	47
4.1 Introduction.....	47
4.2 WS-BPEL Process .....	49
4.3 Proposed Metrics for a Composite Service – Orchestration.....	50
4.3.1 BPEL Process Evolution Metrics ( $BEM_I$ and $BEM_E$ ).....	50
4.3.2 BPEL Process Usefulness Metric under Evolution ( $BUMEP$ & $BUMEN$ )..	54
4.4 Experiments and Analysis.....	60
4.5 Time Complexity .....	63
4.6 Metrics Formal Validation.....	64
4.7 Summary .....	68
Chapter 5 Metrics for an Evolving Composite Service -Choreography .....	69
5.1 Introduction.....	69
5.2 WS-CDL Process .....	70
5.3 Proposed Metrics for a Composite Service - Choreography .....	71
5.3.1 Metrics for interaction entity .....	73
5.3.2 Metrics for role entity .....	75
5.3.3 Metrics for participant entity .....	76
5.3.4 Additive/Subtractive Evolution Metric ( $AEM^+$ / $SEM^-$ ) .....	78
5.3.5 Evolution Metric (EM) .....	78
5.4 Experiments and Analysis.....	79
5.5 Time Complexity .....	86
5.6 Metrics Formal Validation.....	86
5.7 Summary .....	90
Chapter 6 Implementation.....	91
6.1 Architecture of MCS.....	91
6.2 MCS User Interface .....	94
6.2.1 Using MCS for a Single Service.....	95
6.2.1.1 Real world data .....	96
6.2.1.2 Simulated data.....	98
6.2.2 Using MCS for a Composite Service.....	100
6.2.2.1 Orchestration.....	101
6.2.2.2 Choreography.....	103
6.3 Implementation of MCS .....	105
6.3.1 Single Service - Real World Data.....	105
6.3.2 Single Service – Simulated Data.....	106
6.3.3 Composite Service - Orchestration .....	107
6.3.4 Composite Service - Choreography .....	108
6.4 Summary .....	109
Chapter 7 Conclusion.....	110
7.1 Future Work.....	111
References.....	112

# Chapter 1 Introduction

---

Service Oriented Architecture (SOA) is an architecture which guides the creation and usage of services [1]. One of the important aspects of SOA is that the implementation of a service is independent of its interface [2]–[4]. The service interface exposes functionalities provided by the service provider. It is expressed using Web Service Description Language (WSDL) which is based on Extensible Markup Language (XML) [5].

Service composition is defined as the process of assembling the existing services to make a composite service [1], [6]. A composite service is also known as a business process because it involves coordination/collaboration among services to achieve a specific business goal [7]. It is usually meant for complex or large applications. It can be achieved in two ways. The first is orchestration in which there is a central process which controls and coordinates the services. Web Service Business Process Execution Language (WS-BPEL) is a de facto language to represent web based business processes [8], [9]. A composite service realized using WS-BPEL is also known as a WS-BPEL process. Another way of service composition is through choreography in which there is no single process to control the flow of messages between web services. It describes the collaboration of services to achieve a common business goal. Web service Choreography Description Language (WS-CDL) is a standard language used for choreography specification [10]. Composite service realized by WS-CDL is also known as a WS-CDL process [1], [11], [12].

The layout of this chapter is as follows.

Section 1.1 describes the basics of SOA. Evolution in a single service and a composite service is discussed in Section 1.2. Section 1.3 defines motivation and objective of the thesis. The contribution of the thesis is defined in Section 1.4.



## 1.1 Service Oriented Architecture (SOA)

SOA provides business functionalities as a service which is reusable and platform independent [13], [14]. According to [15], a service is central to represent the logic of the business functionality to be provided. It enhances the efficiency and productivity of an organization. The design principles of SOA are discussed below [15], [16].

Loose coupling - ensure that the service is not tightly coupled to the underlying service logic and implementation.

Service contract - comprises of one or more documents that express its technical interface (to specify the offered functionalities), service level agreement (to describe the quality of service features) etc.

Autonomy - ensures that services have control over their underlying logic and execution environment.

Abstraction - means that the essential information is described in the service contract and its logic remains hidden.

Reusability - intends to reuse the service in other functional contexts.

Composability - different services are assembled together to form composite service, generally, for large complex applications.

Statelessness - minimizes resource consumption by not maintaining the service state.

Discoverability - Services are designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanisms.

Figure 1.1 shows the basic entities of SOA which are service consumer, service provider and a service repository. A service provider publishes a service in the central repository which is consumed by the service consumer [17].

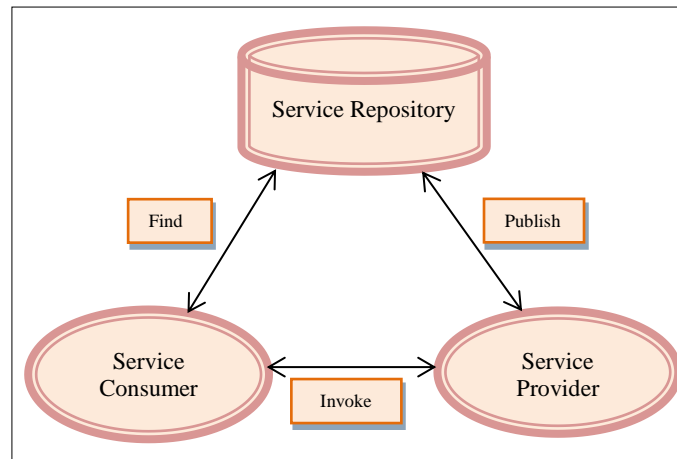


Figure 1.1: Basic entities of service oriented architecture

Now, the basic standards of SOA are discussed.

**1) Web Service:** A web service is defined by World Wide Web Consortium (W3C) which is an international standards organization to develop open standards for the World Wide Web.

“A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.”[18]<sup>1</sup>

Services are the basic building blocks of SOA. They are self-contained units to perform specific tasks. They have an interface to describe service functionalities.

---

<sup>1</sup> <https://www.w3.org/TR/ws-arch/>  
 "Copyright © [2004] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang).  
<http://www.w3.org/Consortium/Legal/2015/doc-license>  
 This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at <http://www.w3.org/TR/>. This is a public Working Group Note produced by the W3C Web Services Architecture Working Group, which is part of the W3C Web Services Activity. This publication as a Working Group Note coincides with the end of the Working Group's charter period, and represents the culmination of the group's work. Discussion of this document is invited on the public mailing list [www-ws-arch@w3.org](mailto:www-ws-arch@w3.org) (public archives). A list of remaining open issues is included in 4 Conclusions. Patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page. Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress. Other documents may supersede this document.

**2) Web Service Description Language (WSDL):** Interface of a web service is described using WSDL. WSDL is an XML based language which describes a service as a set of endpoints, operations and messages to be exchanged between service provider and consumer [19].

**3) Simple Object Access Protocol (SOAP):** It is a XML based communication protocol which is used to exchange messages among services [20]. A SOAP message is exchanged in different ways: one-way, two-way (request-response), two-way with fault message etc [21].

**4) Web Service Business Process Execution Language (WS-BPEL):** It is a standard language to represent web based business processes. It specifies the business process behavior to perform tasks to achieve a business goal. It is used to accomplish the service composition through orchestration [9], [22].

**5) Web Service Choreography Description Language (WS-CDL):** It is a standard language which is used for choreography specification. It is used to describe multi-party interactions based on web services from a global point-of-view [11], [23].

## **1.2 Evolution in SOA**

In today's fast growing environment, services evolve over time. There are many factors which drive these changes. One is the changing demands of the market and another is the desire of business entrepreneurs to enhance the productivity and to increase the value of their services [24], [25].

### **1.2.1 Single Service**

A service is described by its interface. A service interface contains various elements such as operations which could be invoked by the consumer, messages which are exchanged between consumer and provider, service address etc. When a service evolves, its interface may also evolve. The changes that may occur in its elements are addition, deletion, modification, split or merge [26]–[28].

### **1.2.2 Composite service - Orchestration**

A composite service realized through WS-BPEL consists of two types of activities: basic and structured. Basic activities are used to perform basic steps such as to invoke a service, to receive input from a service etc. Examples of basic activities are invoke, receive, reply activities. Whereas structured activities describe the order of execution of activities. If, while, etc. are some of the structured activities. Changes that can occur in a process could be addition, deletion, modification, split and merge in these different activities [29]–[33].

### **1.2.3 Composite Service - Choreography**

A composite service realized using WS-CDL aims to describe interactions among different participants having different roles. It comprises of different entities such as interaction, role, participant etc. WS-CDL process may evolve over time. During this evolution changes such as addition of new participants in the choreography or some interactions being deleted from the existing choreography etc. may occur [34]–[37].

## **1.3 Problem Statement and Motivation**

In SOA, evolution occurs both at a single service level and at a composite service level. At both levels, there is a service provider and a service consumer. Whenever there is a change in the service, it may impact both service consumer as well as service provider. We now discuss this impact at each level in detail.

### **1.3.1 Single Service**

When a service evolves, the service provider is concerned with the impact on its interface in which offered functionalities are described. She/he is interested in analyzing the evolution phases to improve the service interface for the service consumer. On the other hand, a service consumer is concerned with the impact of service evolution on its client code which is used to invoke the service. Also, she/he may also want to know the impact of this evolution on the usefulness of the service.

Figure 1.2 depicts these different aspects of service evolution for the consumer and provider.

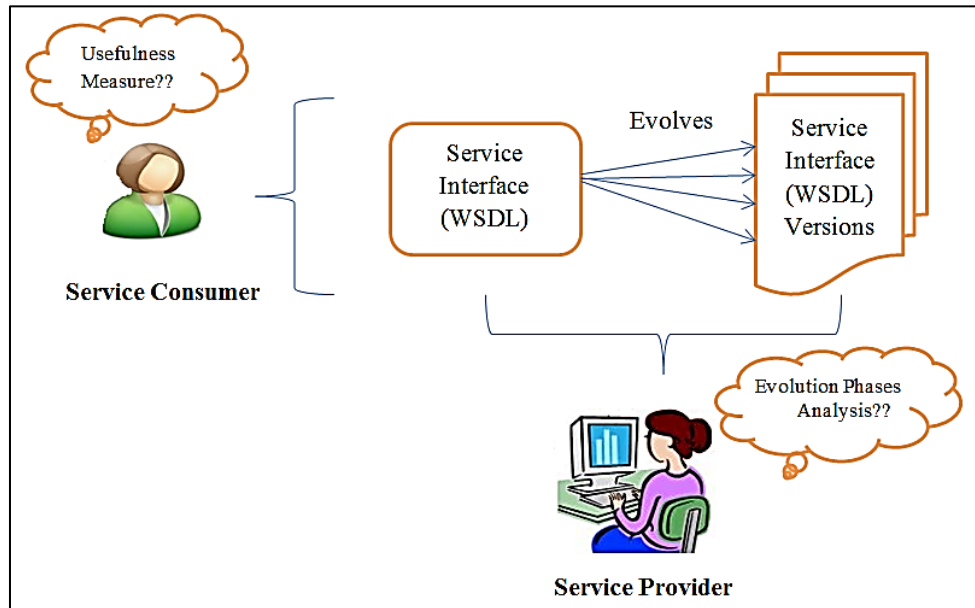


Figure 1.2: Single service evolution

Consider an order placing service,  $S_{\text{orderv1}}$ , which provides ‘place order offline’, and ‘cancel order’ functionalities. Suppose the service changes and its new version is created,  $S_{\text{orderv2}}$ . Assume that there are three changes in the new version which are addition of ‘place order online’ (to support green energy) and ‘change order’ functionalities and removal of ‘cancel order’ functionality. In this example, clearly, the service interface would change. The functionalities of a service are realized by operations in the service interface. When the functionalities change, service interface operations and other corresponding interface elements also change. The service provider is interested in knowing the impact of this evolution on the service interface. Now, consider from the perspective of service consumer. She/he may add the invocation statements for the newly added functionalities and/or may remove the invocation statements for the deleted functionality from the client code. These changes have also changed the usefulness of the service because new functionalities are now offered to the consumer and a previously offered functionality has been removed. Therefore, the service consumer wants to know the impact of this evolution on the service client code and on the service usefulness.

### 1.3.2 Composite Service

Service composition achieved through orchestration or through choreography may evolve over a period of time. First, we discuss evolution and its impact in a composite service achieved through orchestration.

#### 1.3.2.1 Composite Service - Orchestration

Here, we discuss evolution from the perspectives of both the provider as well as the consumer. First, consider the provider. A WS-BPEL process could evolve due to the change in its internal logic such as addition of wait activity or deletion of If activity. Another kind of evolution may involve changes in the interactions with the external service partners such as addition of invoke or receive activity. The provider may want to analyze the kind and quantum of evolution that has taken place. Now, consider the process consumer perspective. A WS-BPEL process is an executable process which has a consumer who uses its functionalities. Evolution in the process may have an impact on its usefulness for the consumer. Therefore, the consumer may want to make a decision about its further consumption based on the quantum of impact on its usefulness. Figure 1.3 shows these aspects of process evolution.

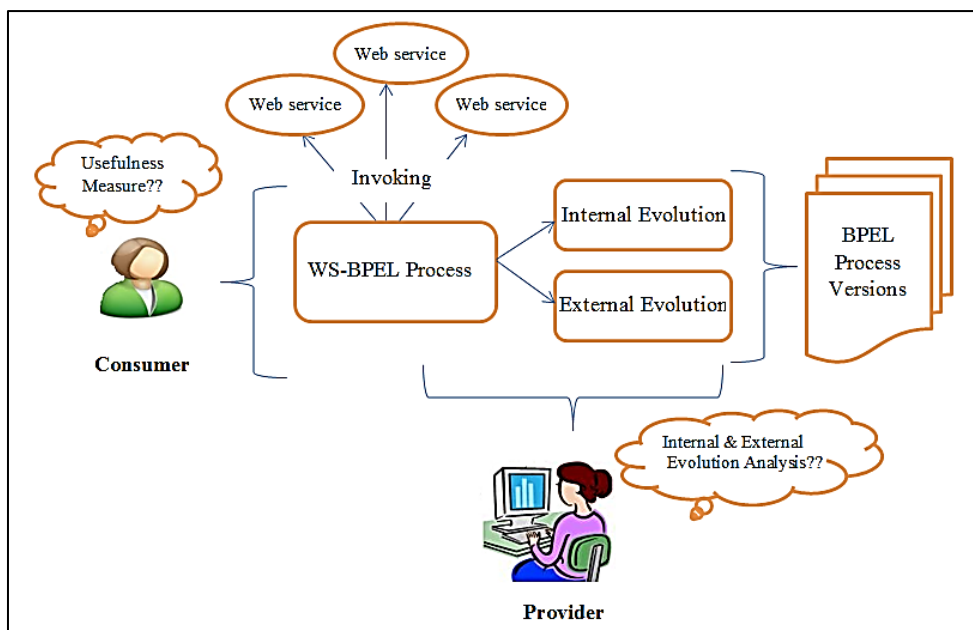


Figure 1.3: Composite service evolution - Orchestration

As an example, consider a Purchase Order (PO) WS-BPEL process which coordinates with the consumer, process order and payment services. Let the activities present in PO version 1 (POv1) be 'receive PO', 'invoke process order service', 'invoke payment service', 'receive change order request', 'invoke process order service' (to change order), 'reply PO Confirmation'. Assume that the process changes and its new version is created, PO version 2 (POv2). The changes done in this version are deletion of 'receive change order request' activity, addition of 'receive order relay coupon request' activity and 'wait' activity for this newly added activity. Clearly, this evolution from POv1 to POv2 has an internal change i.e. addition of 'wait' activity and rest of the changes are external changes. The process usefulness has also changed because a new functionality is now offered to the consumer in POv2 and one offered functionality has been removed from POv1.

#### **1.3.2.2 Composite Service - Choreography**

A WS-CDL process is not an executable process i.e. the aim of the process is to describe peer-to-peer collaborations (interactions) among the participants and to serve an abstract purpose of defining abstract interactions among participants (services) [38]. The process evolves over time due to different types of changes in its entities i.e. participants/ roles/ interactions. Here, some changes are additive in nature such as addition of new participants and some are subtractive in nature such as merging of various interactions in the existing choreography. Therefore, the provider may want to know the nature of evolution and the quantum of change. Figure 1.4 depicts the evolution in composite service using choreography.

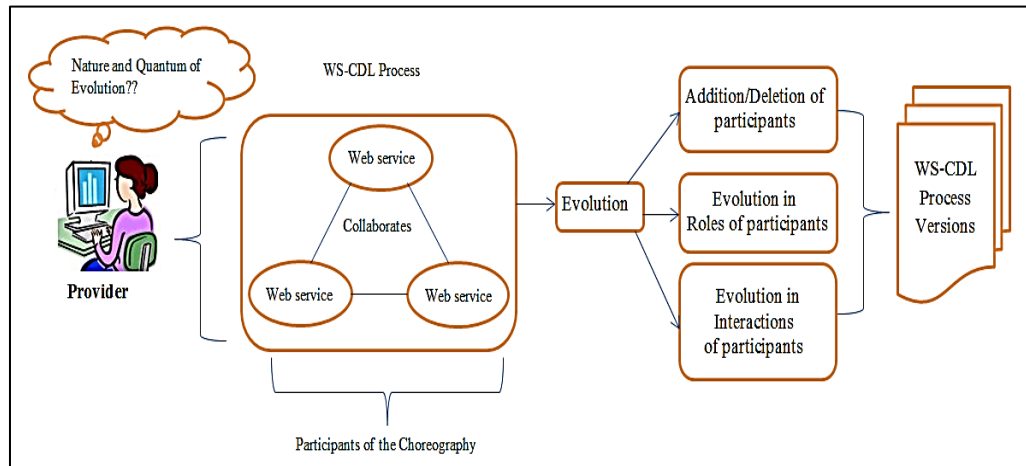


Figure 1.4: Composite service evolution - Choreography

Summarizing, evolution occurs both at a single service level and at a composite service level. At both levels, there is a service provider and a service consumer. The service provider may want to know the nature and quantum of evolution that has taken place in the service. The service consumer may be interested in knowing how much impact the evolution has on service usefulness. Therefore, there is a need to have a measure for evolution in services.

Metrics are a standard of measurement which provides a measure of progress or quality of a process or a product [39]–[43]. Authors have proposed metrics to analyze the evolution in a software and to consider the impact of evolution from an internal (structural) perspective i.e. methods and classes [44]–[49]. We also consider the structure of services to measure the service evolution and its impact on the provider and the consumer. To do so, we propose metrics.

In SOA, metrics have been proposed for a single service [50]–[59] as well as for a composite service (orchestration) [60]–[66]. However, the existing metrics are proposed either:

- a) for the quality attributes such as availability, reliability, performance, security, etc.
- b) or for the structure of a service.



In other words, for a single service and a composite service (orchestration), metrics for evolution or its impact on the service provider and the service consumer have not been proposed. Moreover, we are not aware of any metrics proposed for a composite service (choreography).

In order to address the aforementioned research gap, we propose metrics for evolving services in SOA.

## **1.4 Thesis Contribution**

In this thesis, we propose metrics framework for a single service and a composite service to provide a measure of the evolution and its impact. The proposed metrics framework aims to measure the totality of the changes both for the service provider and consumer.

### **1.4.1 Single Service**

We propose a suite of metrics for a single service using its interface i.e. WSDL document. The suite consists of Service Evolution Metric (SEM), Service Client-Code Evolution Metrics ( $SCEM_M$ ,  $SCEM_O$  and  $SCEM_T$ ) and Service Usefulness Evolution Metric (SUEM) which measures service evolution for the service provider, impact of service evolution on the client code and on the usefulness for the service consumer respectively. The study of correlation between these metrics is conducted which indicates to the service provider whether the changes made in the service have tangible benefits for the consumer. The proposed metrics are empirically validated using real world services of Amazon and also using simulated data. Theoretical validation of the metrics is done using Zuse framework [67].

### **1.4.2 Composite Service - Orchestration**

We propose a metrics framework for a composite service when composition of different services occurs through orchestration. WS-BPEL process document is used to compute metrics. Metrics are proposed both, for the provider as well as the consumer. For the provider, WS-BPEL Evolution Metrics are proposed to provide a

measure of evolution. The metric  $BEM_E$  provides a measure of the quantum of evolution due to the changes in the interactions with the external partner services. On the other hand,  $BEM_I$  metric measure evolution due to change in the internal logic in the process. For the consumer of the process, BPEL Process Usefulness Metric under Evolution in a positive sense ( $BUME_P$ ) and BPEL Process Usefulness Metric under Evolution in a negative sense ( $BUME_N$ ) are proposed to measure the impact on the usefulness for the consumer when it evolves. A WS-BPEL process is a consumer of the partner services. Therefore, when a service evolves, the process may also evolve. Thus, we show that the proposed metrics for the process truly reflect the cohesiveness of changes in a process vis-a-vis changes in services. Proposed metrics are empirically validated. They are also theoretically validated using Zuse framework [67].

### **1.4.3 Composite Service- Choreography**

The entities of the WS-CDL process document are used while defining metrics for a composite service via choreography. Additive Evolution Metric ( $AEM^+$ ) and Subtractive Evolution Metric ( $SEM^-$ ) are proposed to provide the measure for the changes in WS-CDL entities which are additive in nature such as addition of participants and for those changes which are subtractive in nature such as deletion of participants. Evolution Metric (EM) is proposed to measure the overall evolution for the provider. All the proposed metrics are empirically validated using a case study. Theoretical validation using Zuse framework [67] shows that the metrics are above the ordinal scale level.

To summarize, the contribution of the thesis is as follows:

a) Metrics (Service Evolution Metric (SEM), Service Client-Code Evolution Metrics ( $SCEM_M$ ,  $SCEM_O$  and  $SCEM_T$ ) and Service Usefulness Evolution Metric (SUEM)) are proposed for a single service. SEM provides a measure of service evolution for the provider, SCEM and SUEM measures the impact on the client code and usefulness of the service for the consumer.

b) Metrics ( $BEM_I$ ,  $BEM_E$ ,  $BUME_P$  and  $BUME_N$ ) are proposed for a composite service, which is achieved through orchestration, to measure the evolution for the provider and the impact on its usefulness for the consumer.

c) Metrics ( $AEM^+$ ,  $SEM^-$  and  $EM$ ) are proposed for a composite service, which is achieved through choreography, to measure the evolution in a composite service (choreography) as it evolves for the provider.

d) All metrics are theoretically validated using Zuse framework and are found to be above ordinal scale.

e) All metrics are empirically validated using real time data wherever it is available and using simulated data wherever it is not.

The layout of the thesis is as follows:

Chapter 2 provides an overview of the literature on metrics for services. The metrics proposed both for a single service and composite service are discussed.

Chapter 3 discusses the proposed metrics framework for an evolving single service. Metrics are proposed for both the service provider and the consumer.

Chapter 4 discusses the metrics proposed for an evolving composite service achieved through orchestration. Metrics are proposed for both the provider and consumer.

Chapter 5 discusses the proposed metrics framework for an evolving composite service achieved through choreography. A case study is given to analyze the metrics.

Chapter 6 provides conclusion of the thesis, with possible further studies.

# Chapter 2      State of the Art

---

## 2.1 Introduction

As brought out in chapter 1, in SOA, evolution occurs both in a single service as well as in a composite service. In this thesis, the evolution in services is studied via metrics. In this chapter, we review the work which is done on metrics in SOA.

Metrics, in general, are vastly used to provide a quantitative measure of the evolution in software. Metrics have been proposed by authors considering the structural perspective of software. In these metrics, the structure of a software is considered in terms of the classes, methods etc. The successive versions of software are used while proposing metrics. Similarly, in this thesis, we consider the structure of services across its different versions while proposing metrics.

In SOA, a number of efforts have been made to propose metrics for a single service and for a composite service achieved via orchestration. For a composite service using choreography, we are not aware of any work that proposes metrics.

For a single service, metrics are proposed to measure different service attributes. Availability, reliability, performance, security, reputation, resource quality, granularity, complexity, coupling and cohesion etc. are such attributes which are considered by different authors for proposing metrics [50]–[59].

For a composite service via orchestration, authors have used various attributes such as performance, reusability, reliability, complexity, availability etc. [60]–[66]

The layout of this chapter is as follows. Metrics, in general, are discussed in section 2.2. In this section, the importance of metrics, in general, as well as in the context of SOA is mentioned in this section. In section 2.3, metrics proposed for a single service are discussed in detail. Subsequently, review of state of the art of the work done in

proposing metrics for a composite service via orchestration is discussed in section 2.4. Finally, section 2.5 is the concluding section.

## **2.2 Metrics**

Formally, a measure is a number which is assigned to an entity to characterize its attributes [68][69][70]. In general, a measure is used because without a measure one has views or assumptions. A measure may be used to evaluate, control or improve.

Software measurement is a key aspect of good software engineering practice [71][72][73]. It is used to numerically define the attributes, characteristics or properties of a software. It provides a scientific base in the field of software engineering to analyze the software development and maintenance process. Analysis of the measures is done to understand the collected data. Measurement analysis is performed to make or to revisit the decisions that are taken during the software development. Therefore, measurement and its analysis help to understand the software processes. It also provides a way to evaluate the process so as to make decisions about improvement in the process [74].

Metrics are used for software measurement and analysis. A metric represents standard quantitative measurement for the assessment of quality, progress or performance of a software product or a process. They help to determine whether or not we are progressing towards the goal.

In the context of evolution in software, metrics can be used to understand and analyze the evolution in a software system. The internal (structural) perspective of a software across its successive releases [44]–[49] is used while proposing metrics. Software quality from an internal (structural) perspective i.e. methods and classes of a software is considered in [44]. Metrics help to identify the high-risk and low-risk classes of software. The author has performed an empirical analysis of the proposed metrics using successive released versions of software systems. The progress of software development is viewed as a sequence of changes in [45]. Metrics are proposed in order to analyze the development patterns when a software evolves. In [46], the

author has discussed various approaches which are used to understand and improve a software evolution process via metrics. The work done in [47] uses metrics to analyze the evolution in software by detecting relationship between the quantum of change, its type and the time of occurrence of these changes. Evolution in a software system is analyzed using metrics in [48]. Changes in classes e.g., additions or deletions are considered. Metrics are used to compare different versions of a software to analyze whether the decisions taken during software design erodes during evolution or not [49].

In the context of SOA, several metrics have been proposed for different attributes of a service. Metrics may help to know the value of the services of an organization. They can be a powerful tool for informing and guiding decision making at all levels of an organization. Metrics may be useful for both the service provider as well as to the service consumer. A service provider wants to know about the progress of its services and on the other hand service consumer wish to know about the status of the offered functionalities.

In the next section, we discuss different metrics which are proposed for a single service.

### **2.3 Metrics for a single service**

Metrics have been proposed for a single service to measure its attributes such as availability, reliability, performance, security, reputation, resource quality, granularity, and cohesion [50]–[59].

Different requirements of service consumer and service provider have been taken care of while developing a quality metrics model in [50]. Metrics are presented for the service provider for availability and throughput. For the service consumer, the quality attributes for which metrics are proposed are response time and reliability. Metric proposed for a consumer to measure the response time is given below.

$$\text{Response Time} = \text{Completion time of receiving responses from the service} - \text{Completion time of sending the request to the service} \quad (2.1)$$

The quality attribute for the provider, Availability, is defined as

$$\text{Availability} = \text{Uptime} / (\text{Uptime} - \text{Downtime}) \quad (2.2)$$

where Uptime is the total time in which service is available during the time of measurement and the Downtime is the total time that the service is down during measurement time.

In [51], a set of metrics is proposed for the single service taking into account quality features such as availability in terms of time, reliability in terms of process requests, and performance to measure throughput, discoverability etc. The authors have proposed metrics keeping in mind that services in good quality should be published for the service consumer. For example, proposed metric for measuring reliability is

$$\text{RRR} = \text{NumberofReliableResponses} / \text{TotalNumberofRequests} \quad (2.3)$$

Availability of Web Service (AWS) is defined as

$$\text{AWS} = \text{WSOT} / (\text{WSOT} + \text{WSRT}) \quad (2.4)$$

where WSOT represents web service operating time and WSRT represents web service repairing time.

Authors define metrics in [52] to measure the quality of service resources. Here, throughput and utilization for a service provider are measured in terms of how many versions of a service have been made and what is the total lifespan of the service. For measuring the utilization of a service in SOA, metrics proposed by the authors is

$$\text{AUO}[m] - \text{average utilization of Operation } m \quad (2.5)$$

$$\text{AIMSO}[m] - \text{Average Input Message Size for Operation } m \quad (2.6)$$

$$\text{AOMSO}[m] - \text{Average Output Message Size for Operation } m \quad (2.7)$$

The metrics defined in [53] measures the quality of a web service in SOA. The first quality factor is business value in which service price, reputation, recognition etc. are used to define the metrics. Business value of web services represents the financial growth achieved by offering web services in a particular business. The other quality factor for which authors have proposed metric is for the service consumer. This metric is termed as Service measurement. Response time, throughput and accessibility etc. are the sub-quality factors which are defined under this factor. One such metric proposed by the authors is  $\text{Maximum}_{\text{Throughput}}$  which is defined as

$$\text{max}(\text{NumberofRequestsProcessedbyServiceProviderInMeasuredTime}) / \text{MeasuredTime} \quad (2.8)$$

Service cohesion metrics are presented in [54]. The authors have extended the notion of the cohesion of object oriented design while proposing metrics. Some of the metrics are given below:

Service Interface Usage Cohesion (SIUC) represents the cohesion level of a service in terms of its behavioural communication with its consumers.

$$\text{SIUC: } \text{INV}(\text{clients}, \text{SO}(\text{si}_s) / (\text{num\_clients} * |\text{SO}(\text{si}_s)|) \quad (2.9)$$

where  $\text{SO}(\text{si}_s)$  represents the set of all the operations present in the service  $s$ ;  $\text{INV}$  is a function to count the number of operations which are invoked by a consumer and  $\text{num\_clients}$  is the count of clients of  $s$ .

Service Interface Data Cohesion (SIDC) represents the level of cohesiveness of the service operations with other services in terms of how many parameters they share with each other.

$$\text{SIDC}(s) = |\text{Common}(\text{Param}(\text{so} \in \text{SO}(\text{si}_s)) / \text{totalParamTypes} \quad (2.10)$$



Cohesion metrics, defined in [55], are a measure of how much a service provides agnostic or non-agnostic functionalities. Author has defined agnostic functionality as a generalized functionality i.e. it can be used in other contexts. One of the proposed metrics is DANF (Division of agnostic and non-agnostic functionality). This metric is a measure of how much a service provides agnostic or non-agnostic functionalities. The desired value of the metric is either 0 or 1.

$$\text{DANF} = \frac{|\text{AF}(\text{O}(\text{RI}(\text{SI}(s))))|}{|\text{O}(\text{RI}(\text{SI}(s)))|} \quad (2.11)$$

where  $s$  is service;  $\text{SI}$  is service interface;  $\text{RI}$  is realized interface;  $\text{AF}$  is Agnostic functionality;  $\text{O}$  is total number of operations.

Lack of cohesion metrics, proposed by authors in [56], provides a measure which is the complement of the average sequential and communicational similarity between the pairs of operations in service. A service interface is said to be sequentially cohesive when it has pairs of operations which comprise of common elements in their input and output message. The proposed metric for the sequential cohesion is Lack of sequential cohesion ( $\text{LoC}_S(s_i)$ ) metric. The metric is the complement of the average sequential similarity between the pairs of operations that belong to  $\text{CS}(s_i)$ .

$$\text{LoC}_S(s_i) = 1 - \left[ \frac{\sum_{(op_i, op_j) \in \text{CS}_{s_i}} \text{OpS}_{\text{seq}}(op_i, op_j)}{(|s_i.O| * (|s_i.O| - 1)) / 2} \right] \quad (2.12)$$

where  $s_i.O$  is the service operation,  $op_i$  and  $op_j$  are the operations,  $\text{OpS}_{\text{seq}}(op_i, op_j)$  are the operations that are operations that are sequentially related to each other i.e. input message of a service is the output message of other service.

Authors in [57] have presented service coupling metrics. The notion of the coupling of object oriented design is used in the work to propose coupling metrics in service-oriented design. Metrics such as Weighted Intra-Service Coupling between Elements (WISCE), Weighted Extra-Service Incoming Coupling of an Element (WESICE), Weighted Extra-Service Outgoing Coupling of an Element (WESOCE) etc. are proposed. Authors have used implementation elements of a service interface i.e. OO class, interface, package and business scripts.

In [58], metrics are proposed for service reputation in terms of how trustworthy a service provider has been in complying with the agreed SLA levels. The metrics are defined for service verity, service compliance and service reputation. Service verity is the amount of variance among all the compliance levels of services provided by the service provider. It is defined by the authors as

$$SPL_{\text{verity}} = \sum (WSL_{\text{compl}}^i - \mu)^2 / n \quad (2.13)$$

where  $WSL_{\text{compl}}^i$  is the local compliance of an  $i^{\text{th}}$  service and  $n$  is the count of services. Compliance of a service provider is the average of the SLA compliance values of the offered web services. Reputation is used as a parameter for user ranking.

Reusability of services is evaluated using a quality model in [59]. The model is developed for the service reusability features: business commonality, standard conformance, discoverability etc. One of the proposed metrics for a service measures functional commonality ( $FC_{\text{Opi}}$ ) of an operation of a service. It is defined as below

$$FC_{\text{Opi}} = \text{Num}_{\text{ConsumerRequiringFRofOpi}} / \text{Num}_{\text{TotalConsumers}} \quad (2.14)$$

where  $\text{Num}_{\text{ConsumerRequiringFRofOpi}}$  is the count of consumers who want to invoke the functionality offered via operation  $i$  of the service and  $\text{Num}_{\text{TotalConsumers}}$  is the count of total number of service consumers.

From the foregoing, we conclude that metrics have not been proposed to provide a measure of the service evolution and its impact across its different versions. Moreover, there is a lack of metrics which covers both perspectives of service consumer as well as of the provider. In the next section, we discuss metrics for a composite service.

## 2.4 Metrics for a composite service

In SOA, metrics have been proposed for a composite service via orchestration [50], [60]–[66]. Authors have used WS-BPEL process to measure its performance, granularity, coupling etc.

In [50], metrics are defined for measuring the quality of the composite service quality. The quality attributes which are considered while proposing metrics are availability, composability, performance etc. Availability of Business Process (ABP) is a metric proposed for availability is given below.

$$ABP = BPOT / BPOT + BPRT \quad (2.15)$$

Where BPOT represents operating time of the composite service; BPRT means repairing time for the composite service after a failure has occurred. Values of these metrics are collected by using log files obtained from the BPEL engine.

Metrics proposed in [60] considers service level agreement between consumer and provider of the composite service to measure how much performance, reliability and availability are actually met. A metric latest start time (LST) is defined to measure the time required to start processing the requests. It is given below.

$$LST_i = R_{SLA} - (R_{SLA} / C_{max}) \quad (2.16)$$

where  $R_{SLA}$  is the maximum response time and  $C_{max}$  is the estimated completion time.

A metric is defined in [61] to quantitatively measure the granularity appropriateness of a composite service. In order to determine this, attributes of granularity such as business value, reusability, context independency etc. are considered. The proposed metric Weighted Granularity Level Appropriateness (WGLA) is defined as below.

$$WGLA = ((w_1 \times S_{BV}) \times (w_2 \times S_R) \times (w_3 \times S_{CI})) / (w_4 \times S_{Co}) \quad (2.17)$$

where  $S_{BV}$  is the business value of a service,  $S_R$  is service reusability,  $S_{CI}$  is service the value for the attribute context-independency and  $S_{Co}$  is the service complexity value.

A WS-BPEL process performance monitoring model is developed in [62]. In this model, performance metrics are computed at the run time of the process. These

metrics are Instance and Aggregate metrics. Their computation is based on duration, state, time etc parameters of a BPEL process instances during the run-time.

The work in [63] provides metrics for measuring decoupling of a process taking into account the factors of how many operations of web services are invoked and how many services are present in a process. Their computation is based on duration, state, time etc parameters of a process instances during the run-time. One of the coupling metrics is Average Required Services Dependency Metric (ARSD) which is defined as follows.

$$ARSD = \sum R_i / n \quad (2.18)$$

where  $R_i$  is the count of services which a given service needs to complete its functionalities and  $n$  is the total number of services available in a business domain.

Authors have proposed coupling metrics to measure the number of relationships between services in [64] in a service composition achieved through orchestration. Degree of coupling within a given set of services metric (DCSS) is defined by the authors in terms of a graph drawn for the service connectivity in the system and is calculated by the below formula.

$$DCSS = [Max - \sum \sum d(u,v) ] / Max - Min \quad (2.19)$$

where  $d(u,v)$  is number of calls from node  $u$  to  $v$  for all the services in system.

Authors in [65] have defined metrics for measuring the quality of a composite service in terms of coupling and granularity (i.e. based on principles of service design). The proposed metrics are based on information-theory. They have considered two types of elements in a composition. One is atomic and the other is complex. An atomic element is a service operation and the complex element is a composite service. The information entropy  $H(S_i)$  metric defined for a complex element  $S_i$  is as follows.

$$H(S_i) = p \sum (-\log(P_L(j))) = \sum j (-\log(P_L(j))) / n \quad (2.20)$$

where  $P_L(j)$  represents the probability of invocation of atomic  $j^{\text{th}}$  element (service) by the complex element (composite service). If a composite service invokes a service, then the coupling index is high for the composite service. As per the service design principles, coupling should be as low as possible. The metric reflects that how much is the coupling index.

The work in [66] has used metrics for the service provider to monitor the SLA violations in a service composition. One of such metric is Mean Prediction Error ( $e^-$ ) which is the average of the differences between predicted ( $p_i$ ) and monitored ( $m_i$ ) values for a given number of instances ( $n$ ).

$$e^- = (\sum |m_i - p_i|) / n \quad (2.21)$$

From the aforementioned, it could be seen that the existing metrics do not measure evolution across different versions of a composite service in SOA. Also, there is a lack of metrics proposed for both consumer as well as of the provider of the composite service.

## 2.5 Summary

In this chapter, we have described a general definition of metrics. We discussed the existing proposed metrics for a single service and a composite service in SOA.

We found that the existing metrics do not cover the aspect of evolution in single as well as composite service.

As mentioned in chapter 1, we now propose the metrics framework for both the composite and non-composite service in the coming chapters of the thesis. In the next chapter, we discuss the metrics framework for a single evolving service in detail.

# Chapter 3

## Metrics for an Evolving Single Service

---

### 3.1 Introduction

In SOA, a service is provided by a service provider and consumed by a service consumer. In today's fast growing environment, service evolves over time [24], [25]. These changes in a service are studied via metrics in this chapter. Further, the perspectives of both the service provider and service consumer have been taken care of while defining metrics.

Consider a service which is expressed via an interface. The service interface contains operations, messages, service address etc. Changes that can occur in the service interface are addition, deletion, modification, split or merge [26]–[28].

Changes in a service need to be carefully analyzed by the service provider because the changes which are accumulated through its successive versions may affect the service consumer as well as the provider. As an example, consider an order placing service as mentioned in the section 1.3.1. In this example, the questions that need to be answered are what and how much is the

1. impact on the service interface?
2. impact on the service client code?
3. impact on the usefulness of a service?

In this chapter, metrics to answer the above three questions are proposed. The first metric, Service Evolution Metric (SEM), is an answer to the first question. It is a quantitative measure to represent the amount of evolution in different versions of a service interface. The second metric, Service Client-code Evolution Metric (SCEM), is a measure for the impact on client code when a service undergoes changes. The last

metric, Service Usefulness Evolution Metric (SUEM) is a measure of usefulness of the service for the consumer when a service changes.

The layout of this chapter is as follows. Section 3.2 defines the service interface (WSDL document). The service evolution metric (SEM), service client code evolution metrics (SCEM) and service usefulness evolution metric (SUEM) are presented in section 3.3. Time complexity of metrics is given in section 3.4. In section 3.5, experiments and analysis are shown. The formal validation of metrics using Zuse framework is presented in section 3.6. Finally, the chapter is concluded in section 3.7.

### 3.2 Service Interface

The interface of a service is described as a WSDL document. A WSDL document is based on XML which contains a set of operations with inputs/outputs which are exchanged when the operations are invoked [19]<sup>2</sup>. The WSDL 2.0 is defined in terms of XML Infoset in Figure 3.1. This WSDL 2.0 Infoset is described in Table 3.1. It is used to compute the metrics.

---

<sup>2</sup> <https://www.w3.org/TR/wsdl20-primer/>  
"Copyright © [26 June 2007] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang).  
<http://www.w3.org/Consortium/Legal/2015/doc-license>"

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at <http://www.w3.org/TR/>. This is the W3C Recommendation of Web Services Description Language (WSDL) Version 2.0 Part 0: Primer for review by W3C Members and other interested parties. It has been produced by the Web Services Description Working Group, which is part of the W3C Web Services Activity. Please send comments about this document to the public [public-ws-desc-comments@w3.org](mailto:public-ws-desc-comments@w3.org) mailing list (public archive). The Working Group released a test suite along with an implementation report. A diff-marked version against the previous version of this document is available. This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web. This document is governed by the 24 January 2002 CPP as amended by the W3C Patent Policy Transition Procedure. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

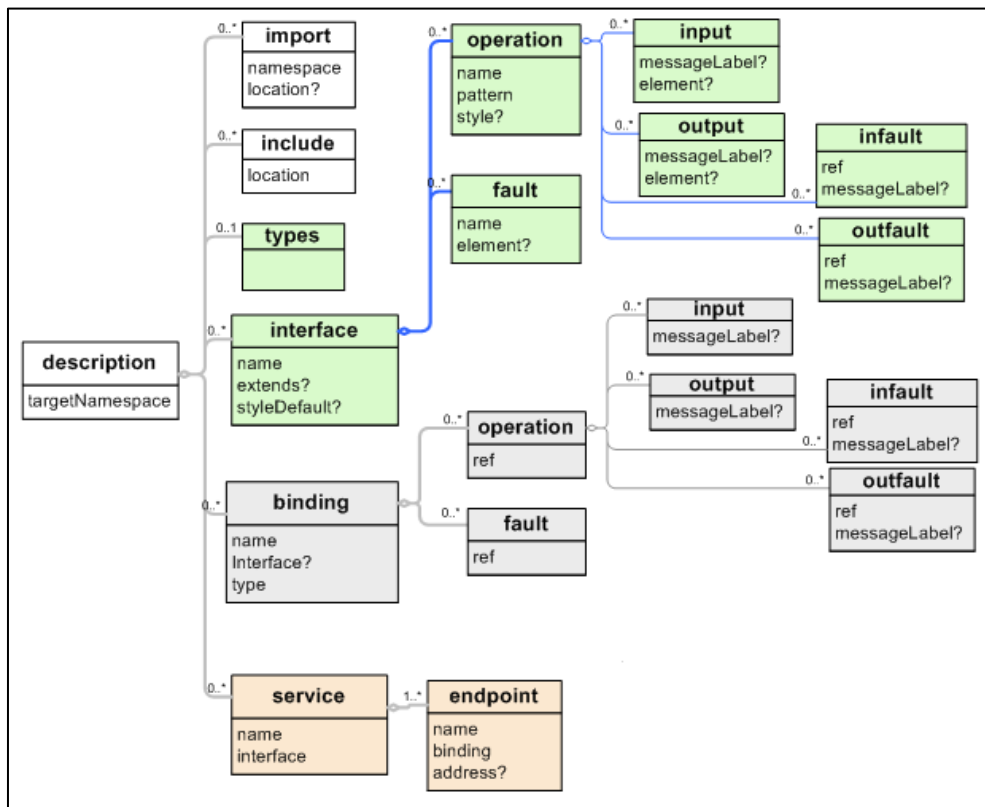


Figure 3.1: WSDL 2.0 Infoset

Table 3.1: WSDL 2.0 description

S.No	Elements with their attributes	Description
1	Description	Root element and all WSDL elements are nested inside this element.
2	Import	Import XML Schemas or other WSDL documents.
3	Include	Assemble contents of a given WSDL 2.0 namespace from several WSDL 2.0 documents that define elements for that namespace.
4	Types	A specification of the data types exchanged between client and service
5	Interface Operation Input Output InFault OutFault Fault	Describes what operations a service has, and what messages are exchanged for each operation (input/output) and also describes possible fault messages.
6	Binding Operation Fault	Describes how a service is accessed over the network. A binding operation describes a concrete binding of an interface operation to a concrete message format. A binding fault associates a concrete message format with an abstract fault of an interface.
7	Service Endpoint	Describes where web service can be accessed on the network via a URL (Endpoint).
8	Documentation	Is optional and contain a humanly readable description of the service.



In the next section, the metrics are presented.

### **3.3 Proposed Metrics for a Single Service**

As stated in section 3.2, the aim is to propose metrics for a service when changes occur. In this section, we propose three metrics for an evolving service which are: Service Evolution Metric (SEM), Service Client Code Evolution Metric (SCEM) and Service Usefulness Evolution Metric (SUEM). These are explained below.

1. *Service Evolution Metric (SEM)*: SEM gives a measure of how much a service interface has changed. A service contains elements and sub-elements as depicted in Figure 3.1. The metric measures the changes in the service vis-à-vis changes in these elements and sub-elements.
2. *Service Client Code Evolution Metric (SCEM)*: SCEM provides a measure of the impact of changes in service on the service client code. The three types of changes are categorized according to their impact on the client code: mandatory, optional and trivial. The computation of the metric is based on this categorization so that the consumer is provided with the measure of how much change he has to accommodate in his code to access the service.
3. *Service Usefulness Evolution Metric (SUEM)*: SUEM is meant to provide a measure of the impact of changes in service on the usefulness for the consumer. The changes are classified into three categories namely Favorable, Unfavorable and Uncertain as per their impact on service usefulness and then define metrics for each of them. The Usefulness Evolution metric is then computed by combining the individual measures for the three categories using weights.

Next, the metrics are discussed in detail.

#### **3.3.1 Service Evolution Metric (SEM)**

In this section, metrics are defined to quantify the amount of changes in service interface that have occurred. Changes in all the elements and their sub-elements listed in Table 3.1 contribute to the computation of the metric.

Let,  $x$  and  $x+1$  be two versions of a service and let,  $T_{x,x+1}$  denote the table containing the changes between them. Let  $i$  be an element /sub-element and  $C_i (T_{x,x+1})$  be the number of changes for that element in the table  $T_{x,x+1}$ . For example, the element operation can undergo a number of changes such as addition, deletion etc. Suppose 5 operation elements are added in the table  $T_{x,x+1}$ , then the value for  $C_i$  is 5. When the context of the changes is unambiguous,  $T_{x,x+1}$  is not defined while defining the metrics.

In general, when there is a change in an element, it may also bring about changes in its sub-elements. For example, if an operation is added then the input and output and even infault and outfault could be added.

In the WSDL 2.0 infoset, each element and its sub-element is at a certain depth. This depth is used to assign a weight to the elements and sub-elements. Let  $D_i$  be the depth of an element. Its weight is  $\prod_{j=1}^i \frac{1}{D_j}$ . For example, the element operation is at depth 2, its sub-elements input and output are at depth 3. The weight for the operation element is  $1/(1*2)$  and weight for its sub-elements input and output is  $1/(2*3)$ .

The Service Evolution Metric is computed as the summation of all the changes in elements and sub-elements together with their respective weights.

$$SEM = \sum_{i=1}^n (\prod_{j=1}^i \frac{1}{D_j}) * C_i \quad (3.1)$$

where  $n$  is the total number of changed elements and sub-elements.

For example, let  $n$  be 4 i.e. there is a total of four changed elements. Let these be interface, operation, input and output. Let the number of changes for element interface be 2, 5 for its sub-element operation, 8 for operations' sub-element input and 9 for output. Then, Service Evolution Metric is

$$\begin{aligned} SEM &= (1/1)*2 + (1/(1*2)) *5 + (1/(1*2*3))*8+(1/(1*2*3))*9 = .5+.4+1.33+1.5 \\ &= 3.73 \end{aligned}$$

In the next section, we define metrics from the perspective of the service consumer.

### **3.3.2 Service Client- code Evolution Metric (SCEM)**

A web service client code is required to invoke the service. This client code can be developed in any language such as Java, .NET, C# etc. In Eclipse, web service client code to invoke the service is developed using IDE which has Web Tools Platform (WTP). The client is generated based on the WSDL document of the service.

The elements of the WSDL document which are required and used to develop the client code are interface, port, operation and types [75]–[78]. Figure 3.2 shows that changes in the WSDL document may introduce changes in the generated web service client code. The changes are classified according to their effect on the client code. These categories are discussed below.

1. **Mandatory changes:** These are the changes which a client has to include in the code. An example of mandatory change is deletion of operation which necessitates the client to remove the corresponding invocation in the code.
2. **Optional changes:** These are the changes which are not compulsory for a client i.e. client may opt to include them in the code. Addition of operation in the service is an example of optional change because it may make the client to use the provided functionality and include the invocation in the code.
3. **Trivial changes:** These are the changes which are immaterial to include in the code. Addition of import or include is an example of trivial change as it does not need any modification in the client code because they are not used while writing the client code; modification in the operation will not affect its invocation in the code.

This classification of changes is presented in Table 3.2 in detail. Now, metrics are defined.

Table 3.2: Categories of changes for Service Client Code

S.No	Category	Description	Changes
1	Mandatory	Compulsory to accommodate in the client code	<p><i>Delete:</i> Types, Interface, Operation, Fault, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault Service, Endpoint</p> <p><i>Split:</i> Types, Interface, Operation, Fault, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault, Service,Endpoint</p> <p><i>Merge:</i> Types, Interface, Operation,Fault,Input,Output,InFault,OutFault, Binding,BindingOperation,BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault, Service,Endpoint</p>
2	Optional	Not necessarily to accommodate in the client code	<p><i>Add:</i> Types, Interface, Operation,Fault,Input,Output,InFault,OutFault, Binding,BindingOperation,BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault, Service, Endpoint</p>
3	Trivial	Insignificant to include in client code	<p><i>Add:</i> Import, Include, Documentation</p> <p><i>Delete:</i> Import Include, Documentation</p> <p><i>Modify:</i> Import, Include, Types, Interface,Operation,Fault,Input,Output,InFault,OutFault, Binding,BindingOperation,BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault, Service, Endpoint</p> <p>Import, Include, Documentation, Description, Namespace</p> <p><i>Split:</i> Import, Include, Documentation</p> <p><i>Merge:</i> Import, Include, Documentation</p>

Let  $C_M$  be the total number of changes under the first category,  $C_O$  be the total number of changes under the second category and  $C_T$  be the total number of changes under the third category. The individual metrics for each category are computed. The weight is computed as a proportion of the number of changes for each type of change within the category to the total number of changes in the category.

The Service Client-Code Evolution Metric for the first category is computed as

$$SCEM_M = \sum_{i=1}^n W_{M_i} * C_{M_i} \quad (3.2)$$

where n is the count of different types of changes in this category,  $C_{M_i}$  is the total number of changes for  $i^{th}$  type of change within this category and  $W_{M_i}$  is the weight for the  $i^{th}$  type of change. For example, if types and operations are deleted then there are two types of changes i.e.  $n=2$ . If 5 types and 3 operations are deleted then  $C_{M1}$  is 5 &  $W_{M1}$  is 5/8 and  $C_{M2}$  is 3 &  $W_{M2}$  is 3/8. So,  $SCEM_M = (5/8)*5 + (3/8)*3 = 4.25$

The Service Client-Code Evolution Metric for the second category is computed as

$$SCEM_O = \sum_{j=1}^m W_{O_j} * C_{O_j} \quad (3.3)$$

where m is the total number of different types of changes in this category (listed in Column 4 of Table 3.2) and  $C_{O_j}$  is the total number of changes for  $j^{th}$  type of change under this category and  $W_{O_j}$  is the weight for the  $j^{th}$  type of change. For example, if types and operations are added then there are two types of changes i.e.  $m=2$ . If 4 types and 3 operations are added then  $C_{O1}$  is 4 &  $W_{O1}$  is 4/7 and  $C_{O2}$  is 3 &  $W_{O2}$  is 3/7. So,  $SCEM_O = (4/7)*4 + (3/7)*3 = 3.57$

The Service Client-Code Evolution Metric for the third category is computed as

$$SCEM_T = \sum_{k=1}^l W_{T_k} * C_{T_k} \quad (3.4)$$

where l is the total count of different types of changes in this category,  $C_{T_k}$  is the total number of changes for  $k^{th}$  type of change  $C_T$  and  $W_{T_k}$  is the weight for the  $k^{th}$  type of change. For example, if import, include and documentation are added then there are three types of changes i.e.  $l=3$ . If 1 import, 2 include and 5 documentation are added then  $C_{T1}$  is 1 &  $W_{T1}$  is 1/8,  $C_{T2}$  is 2 &  $W_{T2}$  is 2/8 and  $C_{T3}$  is 5 &  $W_{T3}$  is 5/8. So,  $SCEM_T = (1/8)*1 + (2/8)*2 + (5/8)*5 = 3.75$

### **3.3.3 Service Usefulness Evolution Metric (SUEM)**

Different changes have different impact on the usefulness for the consumer. Some changes make the service favorable to the consumer; some do not and some are neutral. Therefore, we first, categorize the changes as per their impact on usefulness for the consumer.

The proposed three categories are discussed below.

1. Favorable changes: These changes make the service favorable to the consumer. Addition of operation is a favorable change as a new functionality is added in a service for the consumer, import or include are Favorable changes because they add more elements in the service which make the service more advantageous for the consumer; addition of documentation makes the service more understandable and easy to use to the consumer.

2. Unfavorable changes: These changes make the service less favorable to the consumer. For example, deletion of an operation deprives the consumer of the functionalities that he may be using.

3. Indifferent changes: These changes have insignificant impact on the consumer i.e. neither they make the service favorable nor they make the service unfavorable to the consumer. For example, modification in the service address or merging of two imports will not affect the consumer in the sense that the functionalities will still be accessible to the consumer.

The classification of changes with respect to the impact on the usefulness of a service in Table 3.3 is presented in detail.

Table 3.3: Categories of changes for Usefulness of service

S.No	Category	Description	Changes
1	Favorable	Makes the service more favorable to the consumer	<i>Add:</i> Import, Include, Types, Interface, Operation, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault, Service, Endpoint, Documentation
2	Unfavorable	Makes the service less favorable to the consumer	<i>Delete:</i> Import, Include, Types, Interface, Operation, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault Service, Endpoint, Documentation
3	Indifferent	Have insignificant impact on the consumer	<p><i>Modify:</i> Documentation, Description, Namespace, Service, Endpoint, Import, Include, Types, Interface, Operation, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault</p> <p><i>Merge:</i> Import, Include, Types, Interface, Operation, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault Service, Endpoint, Documentation</p> <p><i>Split:</i> Import, Include, Types, Interface, Operation, Input, Output, InFault, OutFault, Binding, BindingOperation, BindingFault, BindingInput, BindingOutput, BindingInFault, BindingOutFault Service, Endpoint, Documentation</p>

Consider, now, the metrics. Let  $C_F$  be the total number of changes under the first category,  $C_{UF}$  be the total number of changes under the second category and  $C_U$  be the total number of changes under the third category.

The Service Usefulness Evolution Metric for the first category is computed as

$$\text{SUEM}_F = \sum_{i=1}^n W_{F_i} * C_{F_i} \quad (3.5)$$

where n is the total number of different types of changes in this category (listed in Column 4 of Table 3.3) ,  $C_{F_i}$  is the total number of changes for  $i^{\text{th}}$  type of change within this category and  $W_{F_i}$  is the weight for the  $i^{\text{th}}$  type of change. For example, if types and operations are added then there are two types of changes i.e.  $n=2$ . If 2 inputs and 3 outputs are added then  $C_{F_1}$  is 2 &  $W_{F_1}$  is  $2/5$  and  $C_{F_2}$  is 3 &  $W_{F_2}$  is  $3/5$ . So,  $\text{SUEM}_F = (2/5)*2 + (3/5)*3 = 2.6$ .

The Service Usefulness Evolution Metric for the second category is computed as

$$\text{SUEM}_{UF} = \sum_{j=1}^m W_{UF_j} * C_{UF_j} \quad (3.6)$$

where m is the count of different types of changes in this category and  $C_{UF_j}$  is the total number of changes for  $j^{\text{th}}$  type of change under this category and  $W_{UF_j}$  is the weight for the  $j^{\text{th}}$  type of change. For example, if types and operations are deleted then there are two types of changes i.e.  $m=2$ . If 1 fault and 2 operations are deleted then  $C_{UF_1}$  is 1 &  $W_{UF_1}$  is  $1/3$  and  $C_{UF_2}$  is 2 &  $W_{UF_2}$  is  $2/3$ . So,  $\text{SUEM}_{UF} = (1/3)*1 + (2/3)*2 = 1.67$

The Service Usefulness Evolution Metric for the third category is computed as

$$\text{SUEM}_I = \sum_{k=1}^l W_{I_k} * C_{I_k} \quad (3.7)$$

where l is the count of different types of changes in this category,  $C_{I_k}$  is the total number of changes for  $k^{\text{th}}$  type of change within this category and  $W_{I_k}$  is the weight for the  $k^{\text{th}}$  type of change. For example, if operations are merged and documentation is changed, then, there are two types of changes i.e.  $l=2$ . If 2 operations are merged and 5 documentation are changed then  $C_{I_1}$  is 2 &  $W_{I_1}$  is  $2/7$ ,  $C_{I_2}$  is 5 &  $W_{I_2}$  is  $5/7$ . So,  $\text{SUEM}_I = (2/7)*2 + (5/7)*5 = 4.14$ .



Now, we compute the Service Usefulness Evolution Metric of a service by combining all the above defined metrics. We combine these metrics because a consumer is always interested in knowing the impact on the overall usefulness when a service evolves. Weights are assigned to each category bearing in mind that each category has different significance in terms of the impact of evolution on the usefulness of service for the consumer.

$w_{F^*}$ ,  $w_{UF^*}$ ,  $w_{I^*}$  denotes the weights of first, second and third category respectively. The weights are:  $w_{F^*} = .6$ ,  $w_{UF^*} = .3$ ,  $w_{I^*} = .1$  so that  $w_{F^*} + w_{UF^*} + w_{I^*} = 1$ . The Service Usefulness Evolution Metric is defined as

$$SUEM = w_{F^*} * SUEM_F + w_{UF^*} * SUEM_{UF} + w_{I^*} * SUEM_I \quad (3.8)$$

In the example taken above,  $SUEM_F$  is 2.6,  $SUEM_{UF}$  is 1.67 and  $SUEM_I$  is 4.14 so,

$$SUEM = (.6 * 2.6) + (.3 * 1.67) + (.1 * 4.14) = 1.56 + 0.501 + .414 = 2.475$$

### 3.4 Experiments and Analysis

The metrics are evaluated using the real world service- “Amazon Elastic load balancing”. The changed versions of the WSDL documents of the service are available at the link:

<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/DocumentHistory.html>. The versions are identified by the “release date”, mentioned in Table 3.4. Version 1 of the service is the one with the oldest “release date” and the subsequent versions are assigned in increasing numbers thereafter. The subsequent versions are compared and the changes are identified manually. The metrics are computed and results are shown in Table 5. The Pearson correlation coefficient between SEM and  $SCEM_M$  is 0; SEM and  $SCEM_O$  is 1; SEM and  $SCEM_T$  is .74 and SEM and SUEM is 1. It can be seen from the correlation values between the metrics that even though the client code need not necessarily change as the service evolves, the usefulness of the service for the consumer increases.

Table 3.4: Metrics for the real-world service

Service	Versions	Number of changes	SEM	SCEM			SUEM
				SCEM <sub>M</sub>	SCEM <sub>O</sub>	SCEM <sub>T</sub>	
Amazon Elastic Load Balancing	1 & 2	None	0.00	0.00	0.00	0.00	0.00
	2 & 3	Add Types-20, Modify Namespace-1, Add Operation-4, Add Input-4, Add Output-4	24.32	0.00	14.00	1.00	8.50
	3&4	None	0.00	0.00	0.00	0.00	0.00
	4&5	Add Types-15, Modify Namespace-1, Add Operation-3, Add Input-3, Add Output-3	18.49	0.00	10.50	1.00	6.40
	5&6	None	0.00	0.00	0.00	0.00	0.00
	6&7	None	0.00	0.00	0.00	0.00	0.00
	7&8	Add Documentation-15	15.00	0.00	0.00	15.00	9.00
	8&9	Add Types-20, Modify Namespace-1, Add Operation-4, Add Input-4, Add Output-4	24.32	0.00	14.00	1.00	8.50
	9&10	None	0.00	0.00	0.00	0.00	0.00
	10&11	Add Types-15, Modify Namespace-1, Add Operation-3, Add Input-3, Add Output-3 Add Documentation-8	26.65	0.00	10.50	7.22	6.03
	11&12	None	0.00	0.00	0.00	0.00	0.00
	12&13	Modify Namespace-1, Add Documentation-1	1.50	0.00	0.00	1.50	1.50
	13&14	None	0.00	0.00	0.00	0.00	0.00
	14&15	None	0.00	0.00	0.00	0.00	0.00
	15&16	None	0.00	0.00	0.00	0.00	0.00
	16&17	None	0.00	0.00	0.00	0.00	0.00

Some limitations can be observed in the above real world data in Table 5 which makes the metrics hard to analyze. For example, sparsity in the table for the changes hinders the extensive analysis of the SCEM metric vis-à-vis changes. To analyze

SCEM<sub>M</sub>, there are no mandatory changes in the available versions of the service. Therefore, SCEM<sub>M</sub> cannot be analyzed. Similarly, to analyze SUEM, there are no unfavorable changes in the available versions of the service. Moreover, the table has mostly the same number of changes. This means that using Table 3.5, the effect of varying changes cannot be studied and therefore relation among the metrics cannot be found. Therefore, to overcome the above limitations, the simulated data is used and then the metrics are analyzed. Subsequently, the correlations among the metrics are analyzed.

All the changes (additions, deletions, change, split and merge) are simulated for all elements and sub-elements of a service and its 100 versions are generated. The changes are generated randomly in 10 sets each containing 10 versions of the service. The sets are denoted as SET 1, SET 2, ..., SET 10. The information of all the changes in these sets is stored in their respective tables. The metrics are computed and are shown in Table 3.5.

Table 3.5: Metrics for the simulated service

SET 1										
Metric	V1&2	V2&3	V3&4	V4&5	V5&6	V6&7	V7&8	V8&9	V9&10	V10&11
SEM	276.43	239.97	272.74	291.70	256.04	286.36	277.43	279.05	273.78	285.62
SCEM <sub>m</sub>	7.04	5.99	7.22	7.11	7.51	6.89	6.75	7.55	7.49	6.87
SCEM <sub>o</sub>	5.20	5.32	5.02	6.88	5.80	8.14	5.34	6.30	6.65	5.98
SCEM <sub>t</sub>	0.91	1.00	1.03	1.06	0.99	0.97	0.94	0.93	0.97	0.95
SUEM	6.81	6.15	6.93	6.93	6.34	7.90	6.41	7.08	7.28	6.94
SET 2										
Metric	V11&12	V12&13	V13&14	V14&15	V15&16	V16&17	V17&18	V18&19	V19&20	V20&21
SEM	253.92	232.25	243.58	293.09	265.71	261.75	289.92	270.23	265.43	288.72
SCEM <sub>m</sub>	7.42	7.08	6.96	6.93	7.21	6.88	7.02	7.52	7.33	7.15
SCEM <sub>o</sub>	6.37	6.00	5.19	7.86	5.40	6.59	6.59	6.09	5.60	5.05
SCEM <sub>t</sub>	1.03	0.91	0.97	0.98	0.93	1.01	0.94	0.93	1.05	0.97
SUEM	7.37	6.40	6.82	7.78	6.60	7.39	6.91	7.51	6.84	6.68
SET 3										
Metric	V21&22	V22&23	V23&24	V24&25	V25&26	V26&27	V27&28	V28&29	V29&30	V30&31
SEM	278.92	284.27	249.14	248.73	303.09	269.22	242.33	264.60	248.02	261.23
SCEM <sub>m</sub>	6.75	7.24	6.56	6.06	7.60	7.64	6.53	6.80	6.83	7.18
SCEM <sub>o</sub>	6.09	6.11	7.32	6.37	5.74	6.70	6.06	6.91	6.28	6.37
SCEM <sub>t</sub>	1.03	0.96	0.93	1.02	0.98	1.02	1.04	0.92	0.99	1.00
SUEM	6.95	6.96	7.08	6.89	6.86	7.69	6.38	6.88	6.85	7.20

SET 4										
Metric	V31&3 2	V32&3 3	V33&3 4	V34&3 5	V35&3 6	V36&3 7	V37&3 8	V38&3 9	V39&4 0	V40&4 1
SEM	273.90	266.85	266.77	292.42	263.61	264.53	293.25	282.22	286.93	251.20
SCEM m	7.53	7.22	6.12	6.81	6.98	7.35	7.01	6.92	7.44	6.84
SCEMo	4.95	6.24	6.51	6.54	6.35	6.71	6.12	5.84	5.94	6.28
SCEMt	0.98	0.93	0.92	0.99	0.94	1.06	1.01	0.91	0.96	1.00
SUEM	6.56	7.32	6.87	7.04	7.27	7.55	7.35	6.90	6.77	6.69
SET 5										
Metric	V41&4 2	V42&4 3	V43&4 4	V44&4 5	V45&4 6	V46&4 7	V47&4 8	V48&4 9	V49&5 0	V50&5 1
SEM	289.42	233.94	265.26	267.61	244.06	272.94	267.56	247.88	277.75	266.26
SCEM m	7.21	6.70	7.03	7.24	6.71	6.70	7.19	7.05	6.51	6.87
SCEMo	7.43	5.48	5.24	5.69	6.63	6.43	6.92	4.67	5.59	6.90
SCEMt	0.96	0.99	0.97	0.94	0.97	1.03	0.98	0.99	1.02	0.96
SUEM	7.39	6.29	7.33	7.20	6.60	6.44	7.44	6.65	6.96	7.24
SET 6										
Metric	V51&5 2	V52&5 3	V53&5 4	V54&5 5	V55&5 6	V56&5 7	V57&5 8	V58&5 9	V59&6 0	V60&6 1
SEM	280.08	267.26	274.20	239.65	289.18	299.06	261.26	258.28	268.26	282.93
SCEM m	7.15	7.48	6.71	6.61	7.56	7.51	6.61	6.80	7.46	7.02
SCEMo	6.77	5.94	5.85	3.68	5.19	4.95	7.63	6.15	5.49	5.73
SCEMt	1.00	1.00	0.98	0.97	1.00	1.03	1.06	0.98	0.92	0.93
SUEM	6.81	6.70	7.10	6.31	7.33	6.88	7.07	6.69	6.81	6.64
SET 7										
Metric	V61&6 2	V62&6 3	V63&6 4	V64&6 5	V65&6 6	V66&6 7	V67&6 8	V68&6 9	V69&7 0	V70&7 1
SEM	235.80	237.89	285.60	278.86	252.38	263.33	226.52	272.57	290.65	235.57
SCEM m	6.07	6.34	7.22	7.72	6.04	6.56	7.19	7.29	7.05	6.77
SCEMo	6.77	7.01	5.12	6.31	6.48	5.51	7.70	5.79	7.61	7.01
SCEMt	0.94	0.98	0.98	1.00	0.91	1.01	1.05	0.95	0.96	1.00
SUEM	6.72	6.96	6.83	7.19	6.63	6.47	7.29	6.59	7.44	7.18
SET 8										
Metric	V71&7 2	V72&7 3	V73&7 4	V74&7 5	V75&7 6	V76&7 7	V77&7 8	V78&7 9	V79&8 0	V80&8 1
SEM	287.43	296.60	281.55	239.66	238.91	297.76	241.74	259.26	293.87	262.89
SCEM m	6.16	7.61	7.66	5.82	6.28	7.08	7.23	7.21	7.50	6.38
SCEMo	6.13	6.11	5.07	6.13	7.47	5.78	5.55	6.12	6.70	6.38
SCEMt	1.02	1.00	1.04	0.97	0.93	0.99	1.03	0.87	0.94	1.07
SUEM	6.72	7.10	6.62	6.34	7.18	7.38	7.39	6.83	7.89	7.37
SET 9										
Metric	V81&8 2	V82&8 3	V83&8 4	V84&8 5	V85&8 6	V86&8 7	V87&8 8	V88&8 9	V89&9 0	V90&9 1
SEM	216.68	247.79	259.70	284.10	298.56	258.46	297.80	284.53	217.65	245.63
SCEM m	7.58	6.91	7.37	6.74	6.87	7.68	6.75	7.77	6.01	6.99
SCEMo	6.33	6.74	5.34	6.20	6.04	5.41	6.60	5.44	6.37	6.23
SCEMt	0.95	0.89	0.99	1.07	0.99	0.93	1.01	1.09	1.02	0.86
SUEM	7.10	6.55	7.15	7.02	7.28	6.86	7.11	6.92	6.31	6.27
SET 10										
Metric	V91&9 2	V92&9 3	V93&9 4	V94&9 5	V95&9 6	V96&9 7	V97&9 8	V98&9 9	V99&10 0	V100&10 1
SEM	232.96	256.57	289.93	271.58	238.22	243.10	292.60	237.09	264.94	264.57
SCEM m	6.70	6.46	6.63	6.88	7.39	6.68	7.39	7.11	7.02	6.89
SCEMo	6.01	5.65	6.61	6.62	5.19	5.85	6.30	5.75	6.35	5.75
SCEMt	0.99	0.96	1.00	1.06	0.97	1.05	0.95	0.88	1.02	0.98
SUEM	6.67	6.64	7.33	7.19	6.93	6.70	7.18	6.50	6.12	6.82

Figure 3.3 shows the correlation between SEM and SCEM<sub>O</sub>, and between SEM and SUEM and Figure 3.4 shows the correlation between SEM and SCEM<sub>T</sub>, and between SEM and SUEM. It may be noted that usefulness is positively correlated with the changes in both the cases. The graphs show the correlation between SEM and SCEM<sub>O</sub> and SEM and SCEM<sub>T</sub> as well. Since the client need not make any modifications to the code for SCEM<sub>O</sub> and SCEM<sub>T</sub>, the further study of correlation is not done.

The values are depicted in a line graph in Figure 3.2 (correlation between SEM and SCEM<sub>M</sub>, and between SEM and SUEM), Figure 3.3 (correlation between SEM and SCEM<sub>O</sub>, and between SEM and SUEM) and Figure 3.4 (correlation between SEM and SCEM<sub>T</sub>, and between SEM and SUEM) so as to have an insight into the correlation values of Table 3.6.

Table 3.6: Correlation among metrics

Metric	SET 1	SET 2	SET 3	SET 4	SET 5	SET 6	SET 7	SET 8	SET 9	SET 10
SEM and SCEM <sub>M</sub>	0.40	-0.08	0.71	0.13	0.28	0.66	0.53	0.53	0.10	0.06
SEM and SCEM <sub>O</sub>	0.50	0.36	-0.47	-0.19	0.47	0.11	-0.48	-.25	-0.19	0.71
SEM and SCEM <sub>T</sub>	-0.10	0.04	-0.09	-0.03	0.04	0.15	-0.24	0.17	0.46	0.20
SEM and SUEM	0.69	0.39	0.22	0.04	0.66	0.58	0.01	0.27	0.58	0.56

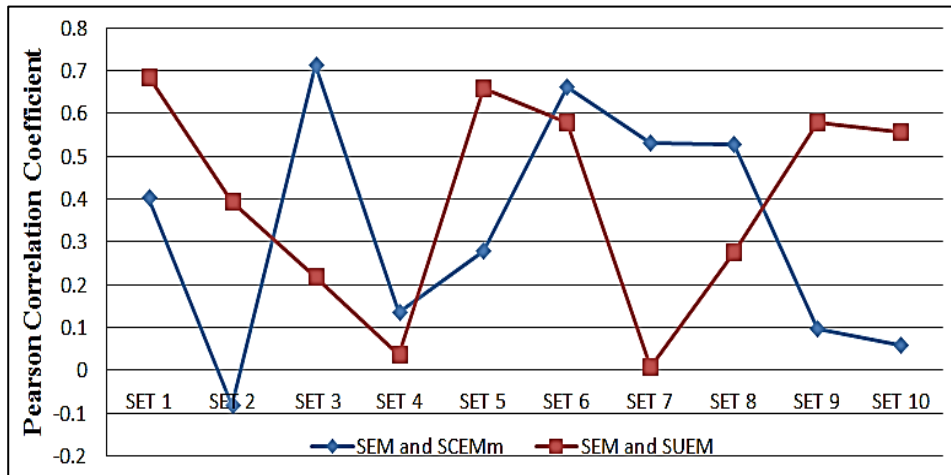


Figure 3.2: Graph showing correlation of SEM & SCEM<sub>M</sub> and SEM & SUEM

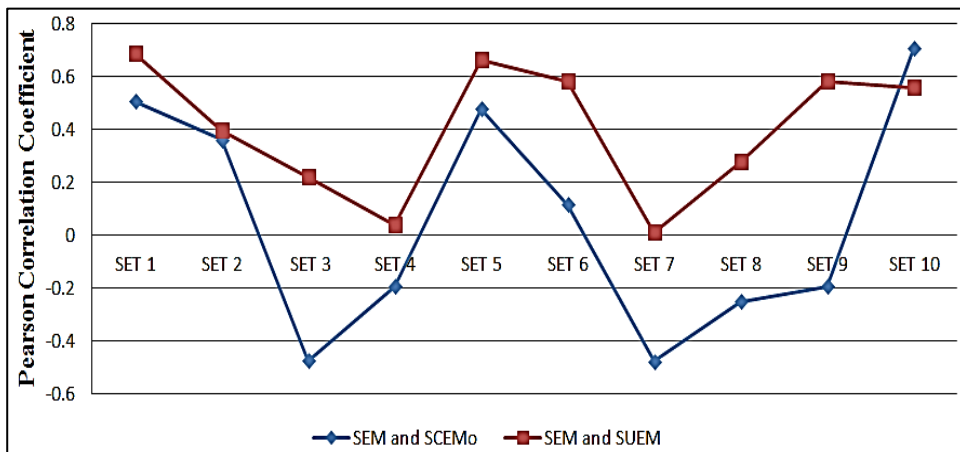


Figure 3.3: Graph showing correlation of SEM & SCEM<sub>0</sub> and SEM & SUEM

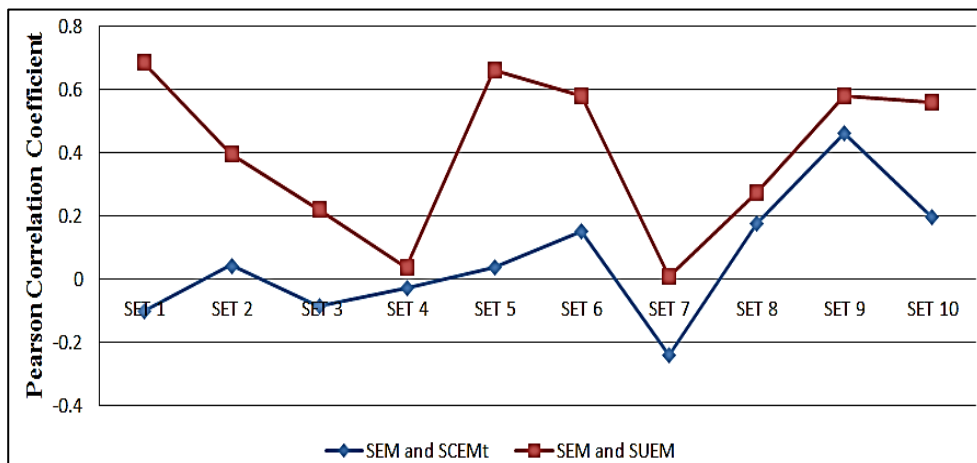


Figure 3.4: Graph showing correlation of SEM & SCEM<sub>T</sub> and SEM & SUEM

Now the analysis of Figure 3.2 is presented.

1. *Phases where the correlation of SEM & SCEM<sub>M</sub> are on a rise but of SEM & SUEM are on a fall:* These phases of evolution such as service evolution from Set 8 to Set 9, may be a concern for the service provider that needs to be looked-into as the client code has to change without much tangible benefits in terms of usefulness.
2. *Phases where the correlation of SEM & SCEM<sub>M</sub> are on a fall but of SEM & SUEM are on a rise :* These phases of evolution such as service evolution from Set 2 to Set 3 are a good indication of changes for the service provider as with very little code change the usefulness has increased. The service provider may analyze the corresponding versions of the service and use the same approach for further service evolution which he has used in these versions.
3. *Phases where the correlation of SEM & SCEM<sub>M</sub> and of SEM & SUEM both rise/fall:* These phases are not as good as the second phase but if usefulness increases albeit with an increase in client code modifications, it is, probably, worthwhile to make such changes.

### **3.5 Time Complexity**

Evolution data of all the changes that occur between two versions of a service is stored in a table in the database. There are five columns for each table. The column headers of the table are

1. Service versions (versions for which data is to be stored)
2. Element (lists the elements of a WSDL document of the service)
3. Depth of the element
4. Number of changes (number of changes for each of additions/deletions/modifications/split/merge of an element)
5. Change category (type of change for the client code and usefulness)

A row specifies the changes for each element of a WSDL document of the service. Whenever a new version of service is created, a new table is created. All the information of the changes that have occurred is inserted into the table.

Metrics proposed for the provider and the consumer uses different columns from the table. The data from the table is accessed sequentially for the metrics computation and computation of the metrics requires only the data which is stored in the table. Therefore, if  $n$  is the number of rows in a table then the time complexity for the metric's computation is  $O(n)$ . Therefore, the time complexity is linear of all the proposed metrics. An example entry is shown in Table 3.7 for operation element.

Table 3.7: Table showing evolution data for operation element

Service versions	Element	Depth of the element	Number of changes		Change category
Version 7&8	operation	2	Add	1	Optional, Favorable
			Delete	2	Mandatory, Unfavorable
			Modify	0	Trivial, Indifferent
			Split	0	Mandatory, Indifferent
			Merge	1	Mandatory, Indifferent

### 3.6 Metrics Formal Validation

In this section, the formal validation of metrics using Zuse's framework [67] is presented. It is a software measurement framework in which there are three structures to determine the scale of a metric using axiomatic approach. These structures are shown in Table 3.8.

A metric is characterized in a measurement scale based on the accomplished category. The scale of a metric helps in analyzing its values and empirical properties. There are four measurement scales which are nominal, ordinal, ratio and absolute. The nominal scale does not play a role in measurement because it just differentiates between the items based on their names. Ordinal scale provides a degree of difference between the values in terms of the order of values. Ratio between the values is allowed by the ratio scale. It also provides exact difference between the metric values. The absolute scale begins at a minimum point and extends in that direction only. This scale is used when,



with respect to zero point, precise values are required for comparison between the values.

Table 3.8: Summarized Zuse Framework

Structure	Axiom	Description
<b>MODIFIED EXTENSIVE STRUCTURE</b>	ME1: $(A, \cdot \succ=)$	Axiom for weak order
	ME2: $A1 \circ A2 \cdot \succ= A1$	Axiom for positivity
	ME3: $A1 \circ (A2 \circ A3) \approx (A1 \circ A2) \circ A3$	Axiom for weak associativity
	ME4: $A1 \circ A2 \approx A2 \circ A1$	Axiom for weak commutativity
	ME5: $A1 \cdot \succ= A2 \Rightarrow A1 \circ A \cdot \succ= A2 \circ A$	Axiom for weak monotonicity
	ME6: If $A3 \cdot \succ A4$ then for any $A1, A2$ , then there exists a natural number $n$ , such that $A1 \circ nA3 \cdot \succ A2 \circ nA4$	Axiom for Archimedean axiom
<b>INDEPENDENCE CONDITIONS</b>	IC1: $A1 \approx A2 \Rightarrow A1 \circ A \approx A2 \circ A$ and $A1 \approx A2 \Rightarrow A \circ A1 \approx A \circ A2$	Condition for weak homomorphism
	IC2: $A1 \approx A2 \Leftrightarrow A1 \circ A \approx A2 \circ A$ and $A1 \approx A2 \Leftrightarrow A \circ A1 \approx A \circ A2$	Condition for homomorphism
	IC3: $A1 \cdot \succ= A2 \Rightarrow A1 \circ A \cdot \succ= A2 \circ A$ , and $A1 \cdot \succ= A2 \Rightarrow A \circ A1 \cdot \succ= A \circ A2$	Condition for weak monotonicity
	IC4: $A1 \cdot \succ= A2 \Leftrightarrow A1 \circ A \cdot \succ= A2 \circ A$ , and $A1 \cdot \succ= A2 \Leftrightarrow A \circ A1 \cdot \succ= A \circ A2$	Condition for monotonicity
<b>MODIFIED RELATION OF BELIEF</b>	MR1: $\forall A, B \in \mathfrak{A}: A \cdot \succ= B$ or $B \cdot \succ= A$	Axiom for completeness
	MR2: $\forall A, B, C \in \mathfrak{A}: A \cdot \succ= B$ and $B \cdot \succ= C \Rightarrow A \cdot \succ= C$	Axiom for transitivity
	MR3: $\forall A \subseteq B \Rightarrow A = \langle B$	Axiom for dominance axiom
	MR4: $\forall (A \supseteq B, A \cap C = \emptyset) \Rightarrow (A \cdot \succ= B \Rightarrow A \cup C \cdot \succ B \cup C)$	Axiom for partial monotonicity
	MR5: $\forall A \in \mathfrak{A}: A \cdot \succ= 0$	Axiom for positivity

Next, we discuss the formal validation of SEM.

### SEM Metric Formal Validation

Let  $S_1, S_2, \dots, S_i, S_{i+1}$  be the versions of a service. The changes between any two versions, say  $S_x, S_{x+1}$  of a process are captured in a  $\text{Diff}_{x,x+1}$  table. Let  $\text{Diff}_{x,x+1}$  and  $\text{Diff}_{y,y+1}$  denotes the tables containing the information of all the changes between these versions. Let  $\text{Diff}$  be the set of all tables which store information of changes across different versions of a service.

The measure SEM is a mapping: SEM: Diff  $\rightarrow$  R such that the following holds for all tables Diff<sub>x,x+1</sub>, Diff<sub>y,y+1</sub>  $\in$  Diff: Diff<sub>x,x+1</sub>  $\succ=$  Diff<sub>y,y+1</sub>  $\Leftrightarrow$  SEM (Diff<sub>x,x+1</sub>)  $\succ=$  SEM (Diff<sub>y,y+1</sub>).

The combination rule of a metric determines the metric behavior when two values of the metric are combined using the concatenation operation. This behavior is required to validate the metric using different axioms of Zuse framework.

In the proposed metrics, the concatenation operation for combination rule is denoted as follows.

$$\text{SEM}(\text{Diff}_{x,x+1} \circ \text{Diff}_{y,y+1}) = \text{SEM}(\text{Diff}_{x,x+1} \cup \text{Diff}_{y,y+1})$$

where Diff<sub>x,x+1</sub>  $\cup$  Diff<sub>y,y+1</sub> is the table containing all the distinct changes in the two tables Diff<sub>x,x+1</sub> and Diff<sub>y,y+1</sub>. In other words, if a change is common to both the tables, then it appears only once in the concatenated table.

### **SEM and the Modified Extensive Structure**

ME1: The binary relation  $\bullet \succ=$  is known to be weak order when it is transitive and complete. Let Diff<sub>1,2</sub>, Diff<sub>3,4</sub> and Diff<sub>5,6</sub> be the three tables where Diff<sub>1,2</sub>, Diff<sub>3,4</sub>, Diff<sub>5,6</sub>  $\in$  Diff. It is true for SEM that either SEM (Diff<sub>1,2</sub>)  $\succ=$  SEM (Diff<sub>3,4</sub>) or SEM (Diff<sub>3,4</sub>)  $\succ=$  SEM (Diff<sub>1,2</sub>). Thus, property of completeness is fulfilled. Now, consider the transitivity property. If SEM (Diff<sub>1,2</sub>)  $\succ=$  SEM (Diff<sub>3,4</sub>) and SEM (Diff<sub>3,4</sub>)  $\succ=$  SEM (Diff<sub>5,6</sub>) then it is obvious that SEM (Diff<sub>1,2</sub>)  $\succ=$  SEM (Diff<sub>5,6</sub>). Thus, transitive property is also accomplished. Therefore, SEM fulfills ME1.

ME2: The positivity of the metric implies that the value of the metric when two tables are combined is bound to be greater than the metric for each individual table. Thus, SEM (Diff<sub>1,2</sub>  $\circ$  Diff<sub>3,4</sub>)  $\succ=$  SEM (Diff<sub>1,2</sub>). Therefore, ME2 is fulfilled.

ME3: Applying the weak associativity rule to the proposed metric, the formulation of the rule becomes, SEM (Diff<sub>1,2</sub>  $\circ$  (Diff<sub>3,4</sub>  $\circ$  Diff<sub>5,6</sub>)) = SEM ((Diff<sub>1,2</sub>  $\circ$  Diff<sub>3,4</sub>)  $\circ$  Diff<sub>5,6</sub>). This means that SEM (Diff<sub>1,2</sub>  $\cup$  (Diff<sub>3,4</sub>  $\cup$  Diff<sub>5,6</sub>)) = SEM ((Diff<sub>1,2</sub>  $\cup$

$\text{Diff}_{3,4}) \cup \text{Diff}_{5,6}$ ). It is obvious that this axiom is fulfilled because union operation is associative.

ME4: The weak commutative axiom is stated as  $\text{SEM}(\text{Diff}_{1,2} \circ \text{Diff}_{3,4}) = \text{SEM}(\text{Diff}_{3,4} \circ \text{Diff}_{1,2})$ . This means that  $\text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{3,4}) = \text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{1,2})$ . Therefore, this axiom is fulfilled because union operation is commutative.

ME5: The property of weak monotonicity is stated as  $\text{SEM}(\text{Diff}_{1,2}) \geq \text{SEM}(\text{Diff}_{3,4}) \Rightarrow \text{SEM}(\text{Diff}_{1,2} \circ \text{Diff}_{5,6}) \geq \text{SEM}(\text{Diff}_{3,4} \circ \text{Diff}_{5,6})$ . This means that  $\text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{5,6}) \geq \text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{5,6})$  (given  $\text{SEM}(\text{Diff}_{1,2}) \geq \text{SEM}(\text{Diff}_{3,4})$ ) needs to be proved. Suppose that the number of common changes between  $\text{Diff}_{3,4}$  and  $\text{Diff}_{5,6}$  are more than the ones between  $\text{Diff}_{1,2}$  and  $\text{Diff}_{5,6}$ . Since common identical changes appear only once in the concatenated table, it may well be the case that  $\text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{5,6}) \geq \text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{5,6})$ . Therefore, this axiom is not fulfilled.

ME6: To prove this axiom, the idempotent property needs to be considered. As per the definition of the concatenation operation, the metric is idempotent i.e.  $\text{SEM}(\text{Diff}_{1,2} \circ \text{Diff}_{1,2}) = \text{SEM}(\text{Diff}_{1,2})$ . Therefore, this axiom is not fulfilled.

It is concluded that SEM does not fulfill the modified extensive structure.

### **SEM and the Independence Conditions**

IC1: It has to be shown that  $\text{SEM}(\text{Diff}_{1,2} \circ \text{Diff}_{5,6}) = \text{SEM}(\text{Diff}_{3,4} \circ \text{Diff}_{5,6})$  and  $\text{SEM}(\text{Diff}_{5,6} \circ \text{Diff}_{1,2}) = \text{SEM}(\text{Diff}_{5,6} \circ \text{Diff}_{3,4})$  given  $\text{SEM}(\text{Diff}_{1,2}) = \text{SEM}(\text{Diff}_{3,4})$ .  $\text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{5,6})$  may be or may not be equal to  $\text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{5,6})$  as the common changes may not be the same between  $\text{Diff}_{1,2} \cup \text{Diff}_{5,6}$  and  $\text{Diff}_{3,4} \cup \text{Diff}_{5,6}$ . The same is true between  $\text{SEM}(\text{Diff}_{5,6} \cup \text{Diff}_{1,2})$  and  $\text{SEM}(\text{Diff}_{5,6} \cup \text{Diff}_{3,4})$ . Hence, this condition is not fulfilled.

IC2: SEM does not accomplish the first condition therefore, it will also not fulfill the second condition.

IC3: Due to non-accomplishment of the fifth axiom of the modified extensive structure, this condition is not fulfilled.

IC4: As IC3 is not fulfilled, thus, IC4 cannot be accomplished.

It can be concluded that SEM does not fulfill the independence conditions.

### **SEM and the Modified Relation of Belief**

MR1: SEM fulfills the weak order i.e. ME1 of modified extensive structure, therefore, this axiom is satisfied.

MR2: If the metric fulfills the weak order i.e. ME1 of modified extensive structure then, this axiom is also satisfied.

MR3: Suppose that all the changes of the table  $\text{Diff}_{3,4}$  are included in  $\text{Diff}_{1,2}$ , then  $\text{SEM}(\text{Diff}_{1,2}) \geq \text{SEM}(\text{Diff}_{3,4})$ . Thus, this axiom is satisfied.

MR4: Suppose that all the changes of the table  $\text{Diff}_{3,4}$  are included in  $\text{Diff}_{1,2}$  and  $\text{Diff}_{1,2} \cap \text{Diff}_{5,6} = \emptyset$ . Then,  $\text{SEM}(\text{Diff}_{3,4}) \geq \text{SEM}(\text{Diff}_{1,2}) \Rightarrow \text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{5,6}) \geq \text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{5,6})$  needs to be proved. Due to the fact that  $\text{SEM}(\text{Diff}_{3,4}) \geq \text{SEM}(\text{Diff}_{1,2})$  and that there are no common changes between  $\text{Diff}_{3,4}$  and  $\text{Diff}_{5,6}$ , the value of  $\text{SEM}(\text{Diff}_{3,4} \cup \text{Diff}_{5,6})$  will be more than  $\text{SEM}(\text{Diff}_{1,2} \cup \text{Diff}_{5,6})$ . Hence this axiom is satisfied.

MR5: This axiom is also satisfied because the changes in a service cannot be less than 0.

Therefore, SEM fulfills the modified relation of belief.

When a metric does not fulfill the axioms of the modified extensive structure and the independence conditions but fulfills the modified relation of belief, it can be characterized above the ordinal scale [87]. Therefore, SEM is a measure above the level of the ordinal scale.

We have validated other proposed metrics using Zuse framework. The results of applying the framework to all metrics are shown in Table 3.9.

Table 3.9: Summary of formal validation of metrics of a single service

Metrics/ Axioms	SEM	SCEM <sub>M</sub>	SCEM <sub>O</sub>	SCEM <sub>T</sub>	SUEM
ME1	Y	Y	Y	Y	Y
ME2	Y	Y	Y	Y	Y
ME3	Y	Y	Y	Y	Y
ME4	Y	Y	Y	Y	Y
ME5	N	N	N	N	N
ME6	N	N	N	N	N
IC1	N	N	N	N	N
IC2	N	N	N	N	N
IC3	N	N	N	N	N
IC4	N	N	N	N	N
MR1	Y	Y	Y	Y	Y
MR2	Y	Y	Y	Y	Y
MR3	Y	Y	Y	Y	Y
MR4	Y	Y	Y	Y	Y
MR5	Y	Y	Y	Y	Y
Scale	Above ordinal	Above ordinal	Above ordinal	Above ordinal	Above ordinal

### 3.7 Summary

In this chapter, a suite of evolution metrics is proposed for a web service for computing service evolution, its impact on client code and usefulness for its consumer. Different versions of the WSDL document of a service are used while proposing metrics. The proposed suite of metrics for evolving service, which is the first effort of its kind, is expected to benefit both the service provider and consumer. They are simple to compute. The proposed metrics have linear time complexity.

Experiments are conducted on the real world and simulated data to empirically validate the metrics. The correlation between the proposed metrics is computed using Pearson correlation coefficient. The experiments clearly demonstrated cases to the provider whether the changes benefitted the consumer and also those cases where the provider may have to re-consider the evolution. The metrics are validated theoretically using Zuse framework and all the metrics are found to be above the ordinal scale.

# Chapter 4

## Metrics for an Evolving Composite Service - Orchestration

---

### 4.1 Introduction

Orchestration is a process to compose services [79][80]. In chapter 3, we proposed metrics for a single service under evolution. In this chapter, we propose metrics for a composite service (through orchestration) when it evolves. Both perspectives of the provider and the consumer have been considered while defining metrics. WS-BPEL is considered to be a de facto language to compose services via orchestration [8]. Therefore, we use WS-BPEL process document to define the metrics.

A WS-BPEL process has a set of activities to compose different web services [9]. The changes that can occur in a process could be addition, deletion, modification, split and merge in its different activities [29]–[33]. All these changes may impact the provider as well as the consumer. First, consider the provider’s perspective. The provider may be interested in knowing the nature and quantum of evolution that has taken place in the process. The evolution in a process may involve changes in the interactions with the external services or may involve changes in the internal logic of the process. Therefore, changes in a process could be categorized, as per their nature, into External and Internal changes. These changes are measured for the provider. Now, consider the consumer’s perspective. Changes may affect the consumer in terms of the usefulness of the process. This impact on the process usefulness is measured for the consumer. Let us understand both of these perspectives with an example.

As an example, consider a Travel Booking process, TB version1 which coordinates with the consumer, Employee and the Airline web service and provides the functionality of booking the flight for an Employee. Activities in this version are ‘receive TB request’, ‘invoke Employee service’ (to retrieve Employee Travel

details), ‘receive Employee service’ (to receive travel status), ‘invoke Airline service (for travel booking), ‘receive Airline service’ (receive booking details), ‘invoke consumer’ (to reply travel booking confirmation)’. Let a Privilege functionality is added in the Airline service, which provides the Employee an opportunity to book a Hotel, rent a Car or subscribe for a magazine along with the discount given by the Airline. The process accommodates the newly offered functionality by the Airline service for the Employee. Therefore, in TB version2, ‘invoke Airline service’ (for availing Privileges) activity and ‘invoke consumer’ (for returning Privilege confirmation details) activity are added. These changes involve interactions with the external partner services. In this sense, these changes are external changes. Now, suppose a new version TB version3 is created in which a wait activity is added in the process to wait for some duration to perform functionalities of TB version2. This is an internal change in the process. Clearly, there are two types of process evolution, one is external in nature and other is internal in nature. This shows the nature of evolution which has taken place in the process for the provider. Now, let another version of the process, TB version4, be created in which the offered functionalities in TB version2 are deleted. Due to this change, there is an impact on the usefulness of the process. This is because the consumer was offered a new functionality in TB version2 but on the other hand, this functionality is removed in TB version4. This shows the impact on the usefulness of the process for the consumer.

Therefore, following questions arise.

- 1) “What” and “by how much” the process has changed?
- 2) How much is the process useful for the consumer?

To provide answers to the above questions, we propose BPEL process Evolution Metrics ( $BEM_I$  and  $BEM_E$ ) for the provider to measure evolution in the process. BPEL process usefulness metric is defined for the process consumer. BPEL Process Usefulness Metric under Evolution in a positive sense ( $BUME_P$ ) metric and BPEL Process Usefulness Metric under Evolution in a negative sense ( $BUME_N$ ) metric is used to measure the impact on the usefulness of the BPEL process when it evolves.

The layout of this chapter is as follows. Section 4.2 defines the BPEL Process. In section 4.3, BPEL evolution metrics for the provider and BPEL process usefulness metrics for the consumer are presented. The time complexity of all the metrics is discussed in section 4.4. Experiments and metrics analysis is given in section 4.5 for all the proposed metrics. The formal validation of metrics using Zuse framework is presented in section 4.6. Finally, the chapter is concluded in section 4.7.

## 4.2 WS-BPEL Process

WS-BPEL process is used to accomplish the orchestration of multiple web services. It specifies the business process behavior to perform the tasks to achieve a business goal. A process consists of two types of activities: basic and structured. We now briefly discuss these activities in Table 4.1.

Table 4.1: WS-BPEL process activities

Type	Definition	Activities
Basic	Used in performing basic steps of a BPEL process	invoke: Invoking a web service. receive: Waiting to receive a message. reply: Send a response in response of the request sent previously. assign: Manipulating data variables throw: Signaling fault explicitly wait: Specify a delay or wait until a deadline is reached empty: Do nothing exit: Immediately end process instance rethrow: Used in fault handlers to rethrow fault caught
Structured	Describe the order of execution of the activities	sequence: Contains activities that will be performed in a sequence flow: Defining a set of activities that will be executed in parallel. if: Implementing decisive behavior. pick: Selecting one of a number of alternative paths. while: Defining loops. repeatUntil: Executes a loop atleast once forEach: Executes a loop using counter



We use BPEL 2.0 standard to compute the metrics. In the next section, we present the metrics.

### 4.3 Proposed Metrics for a Composite Service – Orchestration

As discussed in section 4.1, metrics are proposed for both the provider and the consumer. First, we define metrics from the provider’s perspective.

#### 4.3.1 BPEL Process Evolution Metrics (BEM<sub>I</sub> and BEM<sub>E</sub>)

The evolution metrics are proposed to provide a measure of the nature and quantum of evolution that has occurred in a process for the provider. The evolution of a BPEL process is analyzed along internal and external changes. These changes are categorized as below.

1. *Internal Changes*: Changes in BPEL activities such as If, wait, while, assign etc. may occur. These changes are internal to the process itself i.e. they do not involve interactions with the external services. For example, addition of a wait activity is an internal change.
2. *External Changes*: A process uses external services to accomplish the required business functionalities. It interacts with these services via invoke, receive, reply activities. Any change in these interactions is classified as an external change. For example, addition of an invoke activity is an external change.

Table 4.2 provides a detailed list of the changes in the process activities.

Table 4.2: Category of changes for a WS-BPEL process

Category	Type of change: Activities in the WS-BPEL process
External	Add/Delete/Modify/Split/Merge: invoke, receive, reply
Internal	Add/Delete/Modify/Split/Merge: throw, rethrow, wait, sequence, if, while, repeatUntil, forEach, pick, flow, assign, exit, empty

Now, we define metrics for measuring the evolution of a process. Metrics are defined for both categories of changes.

When a process changes, a new version is created. The metrics are computed for changes in different activities across different versions of a process. Now, to define the metrics, let  $x$  and  $x+1$  be two versions of a process and let,  $T_{x,x+1}$  denote the table containing the changes between them. Let  $i$  be an activity and  $C_i (T_{x,x+1})$  be the number of changes for that activity stored in the table  $T_{x,x+1}$ . For example, a sequence activity can undergo a number of changes such as addition, deletion etc. Suppose 3 wait activities are added in the table  $T_{x,x+1}$ , then the value for  $C_i$  is 3. When the context of the changes is unambiguous  $T_{x,x+1}$  is not mentioned while defining the metrics.

1. Internal Evolution Metric ( $BEM_I$ ):

$$BEM_I = \frac{\sum_i^n w_i * C_i}{n} \quad (4.1)$$

where  $n$  is the count of the total number of types of changes in ‘Internal’ change category,  $C_i$  is the count of changes for  $i^{th}$  type of change within this category and  $w_i$  is the weight for the  $i^{th}$  type of change. The weight is computed as a proportion of the number of changes for each type of change within the ‘Internal’ change category to the total number of changes in the category. For example, if in a process, two wait activities are added ( $C_1$ ) and one throw activity is deleted ( $C_2$ ) then there are two types of changes i.e.  $n=2$ .  $C_1$  is 2 and  $w_1$  is  $2/3$ ;  $C_2$  is 1 and  $w_2$  is  $1/3$ . So,  $BEM_I = ((2/3)*2 + (1/3)*1)/2 = .83$

2. External Evolution Metric( $BEM_E$ ):

$$BEM_E = \frac{\sum_j^m w_j * C_j}{m} \quad (4.2)$$

where  $m$  is the count of the total number of types of changes in ‘External’ change category,  $C_j$  is the count of changes for  $j^{th}$  type of change within this category and  $w_j$

is the weight for the  $j^{\text{th}}$  type of change. For example, if invoke activities are added and modified then there are two types of changes i.e.  $n=2$ . If 3 invoke activities are added and 2 are modified then  $C_1$  is 3 &  $w_1$  is  $3/5$  and  $C_2$  is 2 &  $w_2$  is  $2/5$ . So,  $BEM_E = ((3/5)*3 + (2/5)*2)/2 = 1.3$

In the next section, we discuss the metrics defined for the consumer's perspective.

#### **4.3.2 BPEL Process Usefulness Metric under Evolution (BUME<sub>P</sub> and BUME<sub>N</sub>)**

In this section, we propose two metrics for the BPEL process that is, BPEL Process Usefulness Metric under Evolution in a positive sense (BUME<sub>P</sub>) and BPEL Process Usefulness Metric under Evolution in a negative sense (BUME<sub>N</sub>). Both the metrics are meant to provide a measure of the impact of process evolution on its usefulness for the consumer. Different changes in the process have a different impact on the usefulness for the consumer. Some changes make the process favorable to the consumer; some do not and some are neutral. We classify the changes into three categories namely Favorable, Unfavorable and Indifferent as per their impact on the usefulness. Then, we define metrics for each category. Finally, BUME<sub>P</sub> and BUME<sub>N</sub> are computed by combining the individual metrics for all the three categories.

We first define the categories.

1. *Favorable changes*: These changes make the process more useful to the consumer. For example, addition of invoke activity is a favorable change as new functionalities are added for consumer by invoking a service; addition of throw is a favorable change because it adds fault signaling activities in the process which makes it more advantageous for the consumer in case a fault occurs; addition of documentation makes the process more understandable and easy to use for consumer.
2. *Unfavorable changes*: These changes make the process less useful to the consumer. For example, deletion of invoke activity deprives consumer from the functionality that she/he may be using; deletion of If activity deprives the consumer from the choices that were available to her/him earlier.

3. *Indifferent changes*: These changes have a negligible impact on the consumer i.e. the usefulness of the process remains almost the same. For example, merging of invoke/sequence activities will not affect the consumer in the sense that the functionalities provided by the process will still be accessible to the consumer.

We now present the above classification in detail in Table 4.3. This table lists all the changes in each category for basic and structured activities of a process.

Table 4.3: Category of changes for usefulness of a WS-BPEL process

Category	Activity	Changes in Activities
Favorable	Basic	<i>Add</i> : invoke, receive, reply, throw, rethrow, wait, documentation
	Structured	<i>Add</i> : sequence, If, while, repeatUntil, forEach, pick, flow
Unfavorable	Basic	<i>Delete</i> : invoke, receive, reply, throw, wait, rethrow, documentation
	Structured	<i>Delete</i> : sequence, If, while, repeatUntil, forEach, pick, flow
Indifferent	Basic	<i>Modify/Merge/Split</i> : invoke, receive, reply, throw, wait, rethrow, assign, empty, exit <i>Add</i> : assign, empty, exit <i>Delete</i> : assign, empty, exit
	Structured	<i>Modify/Merge/Split</i> : sequence, If, while, repeatUntil, forEach, Pick, flow

Next, the metrics for each category are defined. We define each individual metric for basic and structured activities under each category and then combine both of them to compute the metric for the respective category.

1) *BPEL Usefulness Metric under Evolution for Favorable changes (BUME<sub>F</sub>)*:

First, we define the metric for basic activities.

Addition of different basic activities have a different degree of impact on the consumer i.e. how much favorable the process has become for the consumer after their addition. This degree of impact is used to classify these activities into different groups as shown in Table 4.4. Weights proportional to the impact are assigned to each group.

Table 4.4: Basic Activities for Favorable Changes

Group	Impact on consumer	Change in Basic Activity	Weight
1	High	Add: invoke, receive, reply	$w_{1=.6}$
2	Medium	Add: documentation, throw	$w_{2=.3}$
3	Low	Add: wait, rethrow	$w_{3=.1}$

The metric for basic activities for favorable changes is computed using Table 4.6 as

$$BUME_{FB} = \sum_{i=1}^3 w_i * G_i \quad (4.3)$$

where  $i$  is the group number,  $G_i$  is the total number of additions of activities in  $i^{\text{th}}$  group and  $w_i$  is the weight for  $i^{\text{th}}$  group.

Next we compute the metric for structured activities for favorable changes.

Here, we define four metrics: Choice Metric (corresponding to `if` activity), Iteration Metric (corresponding to `while`, `repeatUntil` and `forEach` activities), Selection Metric (corresponding to `pick` activity) and Sequence and Parallel Metric (corresponding to `sequence` and `flow` activity). These are combined to define the metric for structured activities for favorable changes. Now we discuss them in detail.

*Choice Metric* ( $F_{CM}$ ): Addition of `if` may give rise to a new choice for the consumer.  $F_{CM}$  gives a measure of how much more the process is after addition of choices.

$$F_{CM} = \begin{cases} \frac{F_{CA}}{F_{CA} + F_C}, & \text{when } F_{CA}, F_C > 0 \\ F_{CA}, & \text{when } F_{CA} > 0, F_C = 0 \\ 0, & \text{when } F_{CA} = 0, F_C > 0 \text{ or} \\ & F_{CA} = 0, F_C = 0 \end{cases} \quad (4.4)$$

where  $F_{CA}$  is the count of choices added and  $F_C$  is the count of choices in the process before additions.

*Iteration Metric* ( $F_{IM}$ ): The loops `while`, `repeatUntil` and `forEach` executes the contained activities based on the count of the specified iterations.  $F_{IM}$  is a measure for

the offered functionalities after addition of loops. We compute the metric by considering the number of activities in the added loops and the number of iterations added for those added loops.

$$F_{IM} = \begin{cases} \frac{F_{IA} * n}{F_{IA} * n + F_I * m}, & \text{when } F_{IA}, n, m, F_I > 0 \\ F_{IA} * n, & \text{when } F_{IA}, n > 0; F_I \text{ or } m > 0 \\ 0, & \text{when } F_{IA} \text{ or } n = 0; F_I, m > 0 \text{ or} \\ & F_{IA} \text{ or } n = 0, F_I \text{ or } m = 0; \end{cases} \quad (4.5)$$

where  $F_{IA}$  is the number of activities added in each added loop,  $n$  is the number of iterations specified for  $F_{IA}$ ,  $F_I$  is the number of activities in the loops before additions,  $m$  is the number of iterations for  $F_I$ .

*Selection Metric* ( $F_{SM}$ ): A pick activity is used to select one event on the basis of a particular message received or on the basis of an alarm. The metric is

$$F_{SM} = \begin{cases} \frac{\frac{1}{F_{M/AA}}}{\frac{1}{F_{M/AA}} + \frac{1}{F_{M/A}}}, & \text{when } F_{M/AA}, F_{M/A} > 0 \\ \frac{1}{F_{M/AA}}, & \text{when } F_{M/AA} > 0, F_{M/A} = 0 \\ 0, & \text{when } F_{M/AA} = 0, F_{M/A} > 0 \text{ or} \\ & F_{M/AA}, F_{M/A} = 0 \end{cases} \quad (4.6)$$

where  $F_{M/AA}$  is the number of onMessage/onAlarm activities added in the added pick activities and  $F_{M/A}$  is the count of onMessage/onAlarm activities in the process before additions in the process.

*Sequence and Flow Metric* ( $F_{SFM}$ ):  $F_{SFM}$  provides a measure of the functionalities in the form of a set of activities performed in a sequence or in parallel. The metric is

$$F_{SFM} = \begin{cases} \frac{F_{SA}}{F_{SA}+F_S} + \frac{F_{FA}}{F_{FA}+F_F}, & \text{when } F_{SA}, F_S, F_{FA}, F_F > 0 \\ F_{SA} + \frac{F_{FA}}{F_{FA}+F_F}, & \text{when } F_{SA} > 0, F_S = 0; F_{FA}, F_F > 0 \\ \frac{F_{FA}}{F_{FA}+F_F}, & \text{when } F_{SA} = 0, F_S > 0; F_{FA}, F_F > 0 \text{ or} \\ & F_{SA}, F_S = 0; F_{FA}, F_F > 0 \\ \frac{F_{SA}}{F_{SA}+F_S} + F_{FA}, & \text{when } F_{SA}, F_S > 0; F_{FA} > 0, F_F = 0 \\ \frac{F_{SA}}{F_{SA}+F_S}, & \text{when } F_{FA} = 0, F_F > 0 \text{ or } F_{FA}, F_F > 0; \\ & F_{SA}, F_S > 0 \\ F_{SA} + F_{FA}, & \text{when } F_{SA} > 0, F_S = 0; F_{FA} > 0, F_F = 0 \\ F_{SA}, & \text{when } F_{SA} > 0, F_S = 0; F_{FA} = 0, F_F > 0; \\ & \text{or } F_{FA} = 0, F_F = 0 \\ F_{FA}, & \text{when } F_{FA} > 0, F_F = 0; F_{SA} = 0, F_S > 0 \\ & \text{or } F_{SA} = 0, F_S = 0 \\ 0, & \text{when } F_{FA}, F_F, F_{SA}, F_S = 0 \end{cases} \quad (4.7)$$

where  $F_{SA}$  is the number of activities added in sequence,  $F_S$  is the total number of activities in sequence in the process before additions,  $F_{FA}$  is the number of activities added in flow and  $F_F$  is the total number of activities in flow in the process before additions.

Different types of computations are used in each of the individual metrics above. Therefore, to combine all these metric values, we use mean of these metrics to reflect the value appropriately. Therefore,

$$BUME_{FS} = \frac{F_{CM} + F_{IM} + F_{SM} + F_{SFM}}{4} \quad (4.8)$$

Next, the metric for the favorable changes is computed by combining metrics for Basic and Structured activities.

$$BUME_F = BUME_{FB} + BUME_{FS} \quad (4.9)$$

2) *BPEL Usefulness Evolution Metric for the Unfavorable changes (BUME<sub>UF</sub>):*

First, we define the metric for basic activities.

The deletion of different basic activities has a different degree of impact on the consumer in terms of how much unfavorable the process becomes for the consumer. We classify these activities into different groups as shown in Table 4.5. Weights proportional to the impact is assigned to each group.

Table 4.5: Basic Activities for Unfavorable Changes

Group	Impact on consumer	Change in Basic Activity	Weight
1	High	Delete: invoke, receive, reply	$w_1=.6$
2	Medium	Delete: documentation, throw	$w_2=.3$
3	Low	Delete: wait, rethrow	$w_3=.1$

The metric for basic activities for Unfavorable changes is computed using Table 4.7.

$$BUME_{UFB} = \sum_{j=1}^3 w_j * G_j \quad (4.10)$$

where  $j$  is the group number,  $G_j$  is the total number of deletions of activities in  $j^{\text{th}}$  group and  $w_j$  is the weight for each group.

Next, we compute the metric for structured activities for Unfavorable changes.

Here also, we define four metrics: Choice Metric (corresponding to If activity), Iteration Metric (corresponding to while, repeatUntil and forEach activities), Selection Metric (corresponding to pick activity) and Sequence and Parallel Metric (corresponding to sequence and flow activities). These are combined to define the metric for structured activities for Unfavorable changes. The metrics are defined below.

*Choice Metric* ( $UF_{CM}$ ):  $UF_{CM}$  gives a measure of how much the process has become less useful after deletion of choices.

$$UF_{CM} = \begin{cases} \frac{UF_{CD}}{UF_C}, & \text{when } UF_{CD}, UF_C > 0 \\ 0, & \text{when } UF_{CD} = 0, UF_C > 0 \text{ or} \\ & UF_{CD} = 0, UF_C = 0 \end{cases} \quad (4.11)$$



where  $UF_{CD}$  is the count of choices deleted and  $UF_C$  is the count of choices in the process before deletions.

*Iteration Metric* ( $UF_{IM}$ ):  $UF_{IM}$  is used to measure by how much the process has become less useful for the consumer after deletion of loops.

$$UF_{IM} = \begin{cases} \frac{UF_{ID} * n}{UF_I * m}, & \text{when } UF_{ID}, n, m, UF_I > 0 \\ 0, & \text{when } UF_{ID} \text{ or } n = 0, UF_I, m > 0 \text{ or} \\ & UF_{ID} \text{ or } n = 0, UF_I, m = 0 \end{cases} \quad (4.12)$$

where  $UF_{ID}$  is the number of activities deleted in the deleted loop,  $n$  is the number of deleted iterations in  $UF_{ID}$ ,  $UF_I$  is the number of count of activities before deletions,  $m$  is the number of iterations in  $UF_I$ .

*Selection Metric* ( $UF_{SM}$ ):  $UF_{SM}$  gives a measure of how much the process becomes less useful when selection activities are deleted from the process.

$$UF_{SM} = \begin{cases} \frac{\frac{1}{UF_{M/AD}}}{\frac{1}{UF_{M/A}}}, & \text{when } UF_{M/AD}, UF_{M/A} > 0 \\ 0, & \text{when } UF_{M/AD} = 0, UF_{M/A} > 0; \\ & UF_{M/AD}, UF_{M/A} = 0 \end{cases} \quad (4.13)$$

where  $UF_{M/AD}$  be the number of onMessage/onAlarm activities deleted and  $UF_{M/A}$  be the total number of onMessage/onAlarm activities in the process before deletions.

*Sequence and Flow Metric* ( $UF_{SFM}$ ):  $UF_{SFM}$  provides a measure of how many functionalities are reduced in the form of a set of activities performed in a sequence or in parallel. The metric is

$$UF_{SFM} = \begin{cases} \frac{UF_{SD}}{UF_S} + \frac{UF_{FD}}{UF_F}, & \text{when } UF_{SD}, UF_S, UF_{FD}, UF_F > 0 \\ \frac{UF_{FD}}{UF_F}, & \text{when } UF_{SD} > 0, UF_S = 0; UF_{SD} = 0, UF_S > 0; \\ & UF_{SD}, UF_S = 0; UF_{FD}, UF_F > 0 \\ \frac{UF_{SD}}{UF_S}, & \text{when } UF_{FD} > 0, UF_F = 0; UF_{FD} = 0, UF_F > 0 \\ & \text{or } UF_{FD}, UF_F = 0; UF_{SD}, UF_S > 0 \\ 0, & \text{when } UF_{FD}, UF_F, UF_{SD}, UF_S = 0 \end{cases} \quad (4.14)$$

where  $UF_{SD}$  be the number of activities deleted in the sequence,  $UF_S$  be the count of activities in the sequence before deletions,  $UF_{FD}$  be the number of activities deleted in the flow and  $UF_F$  be total number of activities in flow before deletions.

All the above calculated metric values are now combined to compute  $BUME_{UFS}$ .

$$BUME_{UFS} = \frac{UF_{CM} + UF_{IM} + UF_{SM} + UF_{SFM}}{4} \quad (4.15)$$

Next,  $BUME_{UF}$  is computed by combining metrics for basic and structured activities.

$$BUME_{UF} = BUME_{UFB} + BUME_{UFS} \quad (4.16)$$

### 3) *BPEL Usefulness Evolution Metric for the Indifferent changes ( $BUME_I$ ):*

The nature of the impact on the consumer for Indifferent changes in process activities makes the metric computation as a summation of all these changes.

$$BUME_I = \frac{1}{p} * \sum_{k=1}^p I_k \quad (4.17)$$

where  $p$  is the total number of activities for Indifferent changes,  $I_k$  is the number of changes in the  $k^{th}$  activity.

Now, we compute BPEL Process Usefulness Metric under Evolution in a positive sense ( $BUME_P$ ) and BPEL Process Usefulness Metric under Evolution in a negative sense ( $BUME_N$ ) by combining the metrics defined in Equation (4.9), Equation (4.16) and Equation (4.17). This is done because a consumer is always interested in knowing the impact on the overall usefulness when a process evolves. Weights are assigned to each category bearing in mind that each category has different significance in terms of the impact of changes on the usefulness of process for the consumer.

We denote  $w_{F^*}$ ,  $w_{UF^*}$ ,  $w_{I^*}$  as the weights for Favorable, Unfavorable and Indifferent changes respectively. We assign  $w_{F^*} = .4$ ,  $w_{UF^*} = .4$ ,  $w_{I^*} = .2$  so that  $w_{F^*} + w_{UF^*} + w_{I^*} = 1$ .

The usefulness metric in a positive sense is denoted as  $BUME_P$ . In this case,  $(w_{F^*} * BUME_F + w_{I^*} * BUME_I) \geq w_{UF^*} * BUME_{UF}$ .

$$BUME_P = (w_{F^*} * BUME_F + w_{I^*} * BUME_I) - w_{UF^*} * BUME_{UF} \quad (4.18)$$

The usefulness metric in a negative sense is denoted as  $BUME_N$ . In this case,  $(w_{F^*} * BUME_F + w_{I^*} * BUME_I) \leq w_{UF^*} * BUME_{UF}$ .

$$BUME_N = w_{UF^*} * BUME_{UF} - (w_{F^*} * BUME_F + w_{I^*} * BUME_I) \quad (4.19)$$

#### 4.4 Experiments and Analysis

First, we show the experiments for the BPEL Evolution metrics i.e.  $BEM_I$  and  $BEM_E$  metrics which are proposed for the provider.











A web service is invoked using client code. When a service undergoes changes, its corresponding client code may also undergo changes. In chapter 3, changes in a web service have been classified into three categories which are Mandatory changes, Optional changes and Trivial changes. Corresponding to these categories, metrics were proposed which are  $SCEM_M$  (for mandatory changes),  $SCEM_O$  (for optional changes) and  $SCEM_T$  (for trivial changes).

A WS-BPEL process is the consumer of web services. When a service changes, the process may have to accommodate the corresponding changes - depending upon the type of changes. The metrics proposed in this chapter and in chapter 3 are shown to be cohesive. For example, when service client code metrics reflect the mandatory changes in service, then  $BEM_I$  and  $BEM_E$  metrics of a process must exhibit a value indicating that a change has occurred for the successful execution of the process. This cohesiveness is demonstrated with the help of an example.

The example of Booking process from Oracle Technology Networks <http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html> is taken. It has two partner services: Airline and Employee services. Employee service is used to give travel status of employee to the process and then based on this status Airline service

returns airline booking details to process. Service and WS-BPEL process code which is taken from the reference cited above are modified. The modified versions are shown in Table 4.6.

Table 4.6: Description of the changes in the service and process

Versions	Changes in the Airline Service	Service Version	Changes in the BPEL Process	BPEL Version
1	Addition: Travel Update & Cancel and Refund functionality	 Airline WSDL Version 1.wsdl	Addition of activities for Travel Update & Cancel and Refund functionality	 Travel BPEL Version 1.bpel
2	Deletion: Travel Update & Cancel and Refund functionality	 Airline WSDL Version 2.wsdl	Deletion of activities for Travel Update functionality Deletion of activities for Cancel and Refund functionality	 Travel BPEL Version 2.bpel
3	Addition: Client Privilege functionality	 Airline WSDL Version 3.wsdl	Addition of activities for Client Privilege functionality	 Travel BPEL Version 3.bpel
4	Addition of Flight Schedule functionality	 Airline WSDL Version 4.wsdl	No change	 Travel BPEL Version 4.bpel
5	Addition of documentation	 Airline WSDL Version 5.wsdl	No change	 Travel BPEL Version 5.bpel

Service version  $V_i$  ( $i \geq 2$ ) is compared with service version  $V_1$  and the changes are stored in the table  $SV_{1,i}$  in the database. The second column of Table 4.4 lists these tables. Similarly, tables for WS-BPEL process are listed in the fourth column of Table 4.7.

Table 4.7: Metrics for the Airline service and the Travel booking process

S.No.	Service Version Table	Service metrics	Process Version Table	Process metrics
1	$SV_{1,2}$	$SCEM_M=2.44$ $SCEM_O=0.00$ $SCEM_T=0.00$	$BV_{1,2}$	$BEM_I=3.00$ $BEM_E=3.33$
2	$SV_{1,3}$	$SCEM_M=0.00$ $SCEM_O=3.31$ $SCEM_T=0.00$	$BV_{1,3}$	$BEM_I=1.80$ $BEM_E=0.80$
3	$SV_{1,4}$	$SCEM_M=0.00$ $SCEM_O=2.44$ $SCEM_T=0.00$	$BV_{1,4}$	$BEM_I=0.00$ $BEM_E=0.00$
4	$SV_{1,5}$	$SCEM_M=0.00$ $SCEM_O=0.00$ $SCEM_T=1.86$	$BV_{1,5}$	$BEM_I=0.00$ $BEM_E=0.00$

Metrics in chapter 3 and the metrics proposed in this chapter are shown in Table 4.4. Next, we analyze the metrics.

- 1) Mandatory changes:  $SCEM_M > 0$  and  $BEM_I$  and  $BEM_E$  have positive values for  $SV_{1,2}$  and  $BV_{1,2}$ . Therefore, there is a clear synchronization between mandatory changes in service client code vis-à-vis changes in the process.
- 2) Optional changes: Metrics for  $SV_{1,3}$  and  $SV_{1,4}$  show optional changes. The process may accommodate (as in  $BV_{1,3}$ ) or may not (as in  $BV_{1,4}$ ). Again, clearly, the changes are synchronized.
- 3) Trivial changes: The last row shows that the process is unaffected by the changes in  $SV_{1,5}$ .

Now, we show the experiments for the metrics i.e.  $BUME_P$  and  $BUME_N$  which are proposed for the consumer.

We simulate the changes (additions, deletions, change, split and merge) for the basic and structured activities of a process due to the non-availability of public WS-BPEL process versions. Starting with Version 1 of a process, changes are simulated to create Version 2. These changes between Version 1 and Version 2 are recorded in a database and denoted as Version 1&2. Next, Version 2 is picked up; changes are made and recorded as Version 2&3. Continuing in this way, we generate changes between versions till Version 10&11. We then compute the metrics. The results are shown in Table 4.8.

Table 4.8: Metric values for a WS-BPEL process

Version/ Metrics	1& 2	2&3	3& 4	4& 5	5&6	6& 7	7& 8	8& 9	9& 10	10& 11
$BUME_F$	5.48	5.48	5.48	5.48	10.46	17.94	23.17	23.15	23.14	23.12
$BUME_{UF}$	0.00	5.13	7.72	9.83	7.61	7.60	7.83	8.14	8.13	8.44
$BUME_I$	3.00	3.00	3.00	3.00	3.00	3.00	3.00	10.23	19.62	34.69
$BUME_P$	2.792	0.74	-	-	1.74	4.736	6.736	8.05	9.928	12.81
$BUME_N$	-	-	1.496	2.34	-	-	-	-	-	-

From the table, it is clearly seen that when  $BUME_P$  increases, the usefulness of the process also increases and vice-versa. On the other hand, usefulness is inversely proportional to the values of  $BUME_N$ . We now present a detailed analysis of the metrics.

1) *Constant  $BUME_F$  &  $BUME_I$ , Increase in  $BUME_{UF}$* : The value of  $BUME_P$  reflects that when unfavorable changes increases, the usefulness of the process for the consumer decreases. This is seen from the values of the metrics for Version1&2, Version2&3 in Table 4.8. Whereas, metrics for Version 3&4 and Version 4&5, shows that the higher the value of  $BUME_N$ , the lesser is the usefulness.

2) *Constant  $BUME_{UF}$  &  $BUME_I$ , Increase in  $BUME_F$* : The value of  $BUME_P$  reflects that when favorable changes increase, the usefulness of the process for the consumer also increases. This is seen in the Version5&6, Version6&7 and Version7&8 in Table 4.8.

3) *Constant  $BUME_F$  &  $BUME_{UF}$ , Increase in  $BUME_I$* : The value of  $BUME_P$  reflects that the increase in indifferent changes has very slight positive impact on the usefulness of the process for the consumer. This is seen in the Version8&9, Version9&10 and Version10&11 in Table 4.8.

From the above analysis, it can be seen that our proposed metrics  $BUME_P$  and  $BUME_N$  appropriately reflects the impact on the usefulness of a WS-BPEL process for the consumer when it evolves.

#### **4.5 Time Complexity**

Information about the evolution in a service is stored in a table. Each row of the table specifies changes for each activity of the process. The table has five columns. The column headers are

- 1) Process versions (evolution data between these two versions is stored)
- 2) Activity (lists the activities of a process)

- 3) Number of changes (the number additions/ deletions/ modifications/ split/merge for each activity)
- 4) Number of activities in previous version (count of each activity present in the previous version)
- 5) Change category (type of change for the process i.e. Internal, External, Favorable, Unfavorable or Indifferent)

Whenever a new version of the process is created, a new table is created to store the changes between this version and its previous version. Then, information of these changes is inserted into that table. Next, the proposed metrics are computed by using evolution data which is stored in the corresponding table. Different metrics use different columns to retrieve the data.

The data from the table is accessed sequentially for the metrics computation. Therefore, if  $n$  is the number of rows in a table, then the time complexity of the metric's computation is  $O(n)$ .

Table 4.9 shows an entry for the evolution data stored in the table for an invoke element.

Table 4.9: Table showing changes for invoke element

Process versions	Activity	Number of activities in previous version	Number of changes		Change category
2&3	invoke	7	Add	1	External, Favorable
			Delete	2	External, Unfavorable
			Modify	0	External, Indifferent
			Split	0	External, Indifferent
			Merge	1	External, Indifferent

#### 4.6 Metrics Formal Validation

All the proposed metrics are theoretically validated using Zuse framework as given in Table 3.8 in chapter 3. Now, we present formal validation of the  $BEM_I$ .

##### $BEM_I$ Metric Formal Validation

Let, there be a process  $P$  having  $n$  versions. Between any two process versions, evolution data is computed and stored in a table. This table is referred to as a

difference table. Let, there be two process versions i.e.  $x$  and  $x+1$ . Let, their difference table be denoted by  $\text{Diff}_{x,x+1}$ . Let  $F$  be the set of all difference tables.

Consider  $\text{BEM}_I$ . The measure  $\text{BEM}_I$  is a mapping:  $\text{BEM}_I: \text{Diff} \rightarrow \mathbb{R}$  such that the following holds for all tables  $\text{Diff}_{x,x+1}, \text{Diff}_{y,y+1} \in \text{Diff}$ :  $\text{Diff}_{x,x+1} \succ = \text{Diff}_{y,y+1} \Leftrightarrow \text{BEM}_I(\text{Diff}_{x,x+1}) \succ = \text{BEM}_I(\text{Diff}_{y,y+1})$ .

The concatenation operation for combination rule is denoted as follows.

$$\text{BEM}_I(\text{Diff}_{x,x+1} \circ \text{Diff}_{y,y+1}) = \text{BEM}_I(\text{Diff}_{x,x+1} \cup \text{Diff}_{y,y+1})$$

where  $\text{Diff}_{x,x+1} \cup \text{Diff}_{y,y+1}$  is the table which contains changes (distinct) in the two tables  $\text{Diff}_{x,x+1}$  and  $\text{Diff}_{y,y+1}$ .

### **$\text{BEM}_I$ and the Modified Extensive Structure**

ME1: The binary relation  $\bullet \succ =$  is known to be weak order when it is transitive and complete. Let  $\text{Diff}_{1,2}$ ,  $\text{Diff}_{3,4}$  and  $\text{Diff}_{5,6}$  be the three tables where  $\text{Diff}_{1,2}, \text{Diff}_{3,4}, \text{Diff}_{5,6} \in \text{Diff}$ . It must be true that either  $\text{BEM}_I(\text{Diff}_{1,2}) \succ = \text{BEM}_I(\text{Diff}_{3,4})$  or  $\text{BEM}_I(\text{Diff}_{3,4}) \succ = \text{BEM}_I(\text{Diff}_{1,2})$ . Thus, the property of completeness is fulfilled. Now, consider the transitivity property. If  $\text{BEM}_I(\text{Diff}_{1,2}) \succ = \text{BEM}_I(\text{Diff}_{3,4})$  and  $\text{BEM}_I(\text{Diff}_{3,4}) \succ = \text{BEM}_I(\text{Diff}_{5,6})$  then it is obvious that  $\text{BEM}_I(\text{Diff}_{1,2}) \succ = \text{BEM}_I(\text{Diff}_{5,6})$ . Thus, the transitive property is also accomplished. Therefore,  $\text{BEM}_I$  fulfills ME1.

ME2: It can be seen that when two tables are combined then the value of the metric  $\text{BEM}_I$  is larger than the value of the metric for each of those tables. Thus,  $\text{BEM}_I(\text{Diff}_{1,2} \circ \text{Diff}_{3,4}) \succ = \text{BEM}_I(\text{Diff}_{1,2})$ . This proves ME2 for  $\text{BEM}_I$ .

ME3: When weak associativity rule is applied to metric  $\text{BEM}_I$ , formulation of rule becomes,  $\text{BEM}_I(\text{Diff}_{1,2} \circ (\text{Diff}_{3,4} \circ \text{Diff}_{5,6})) = \text{BEM}_I((\text{Diff}_{1,2} \circ \text{Diff}_{3,4}) \circ \text{Diff}_{5,6})$ . The concatenation operation for the metric is Union operation. It is known that the union operation is associative, therefore,  $\text{BEM}_I(\text{Diff}_{1,2} \cup (\text{Diff}_{3,4} \cup \text{Diff}_{5,6})) = \text{BEM}_I((\text{Diff}_{1,2} \cup \text{Diff}_{3,4}) \cup \text{Diff}_{5,6})$ . ME3 is satisfied.



ME4: The weak commutative axiom for the metric  $BEM_I$  is stated as  $BEM_I (Diff_{1,2} \circ Diff_{3,4}) = BEM_I (Diff_{3,4} \circ Diff_{1,2})$ . This means that  $BEM_I (Diff_{1,2} \cup Diff_{3,4}) = BEM_I (Diff_{3,4} \cup Diff_{1,2})$ . It is known that the union operation is commutative. Hence,  $BEM_I$  fulfills ME4.

ME5: The property of weak monotonicity is stated as  $BEM_I (Diff_{1,2}) \geq BEM_I (Diff_{3,4}) \Rightarrow BEM_I (Diff_{1,2} \circ Diff_{5,6}) \geq BEM_I (Diff_{3,4} \circ Diff_{5,6})$ . To prove  $BEM_I (Diff_{1,2} \cup Diff_{5,6}) \geq BEM_I (Diff_{3,4} \cup Diff_{5,6})$  (given  $BEM_I (Diff_{1,2}) \geq BEM_I (Diff_{3,4})$ ), let the count of common changes between  $Diff_{3,4}$  and  $Diff_{5,6}$  be more than the count of common changes between  $Diff_{1,2}$  and  $Diff_{5,6}$ . Since these common changes appear once after applying concatenation operation, then the resultant metric computed based on their concatenate tables be  $BEM_I (Diff_{3,4} \cup Diff_{5,6}) \geq BEM_I (Diff_{1,2} \cup Diff_{5,6})$ . Therefore,  $BEM_I$  does not fulfill this axiom.

ME6: Idempotent property is considered here to prove this axiom. A metric is idempotent going by definition of concatenation operation i.e.  $BEM_I (Diff_{1,2} \circ Diff_{1,2}) = BEM_I (Diff_{1,2})$ . Therefore,  $BEM_I$  does not fulfill this axiom.

It is concluded that the modified extensive structure is not fulfilled by  $BEM_I$ .

### **$BEM_I$ and the independence conditions**

IC1: To prove this condition, it has to be shown that  $BEM_I (Diff_{1,2} \circ Diff_{5,6}) = BEM_I (Diff_{3,4} \circ Diff_{5,6})$  and  $BEM_I (Diff_{5,6} \circ Diff_{1,2}) = BEM_I (Diff_{5,6} \circ Diff_{3,4})$  given  $BEM_I (Diff_{1,2}) = BEM_I (Diff_{3,4})$ . Now,  $BEM_I (Diff_{1,2} \cup Diff_{5,6})$  may be or may not be equal to  $BEM_I (Diff_{3,4} \cup Diff_{5,6})$  because the changes (which are common) in  $Diff_{1,2} \cup Diff_{5,6}$  and  $Diff_{3,4} \cup Diff_{5,6}$  may not be the same. The same is true between  $BEM_I (Diff_{5,6} \cup Diff_{1,2})$  and  $BEM_I (Diff_{5,6} \cup Diff_{3,4})$ . Hence, this condition is not fulfilled.

IC2: When a metric does not accomplish IC1, it will also not fulfill IC2. The metric  $BEM_I$  does not fulfill IC1 and therefore does not fulfill IC2.

IC3: When a metric does not accomplish fifth axiom of the modified extensive structure, it will also not fulfill this condition which is the case with  $BEM_I$ .

IC4: A metric not satisfying the condition IC3 cannot accomplish the condition IC4. Hence,  $BEM_I$  does not accomplish IC4.

Therefore,  $BEM_I$  does not fulfill independence conditions.

### **$BEM_I$ and the modified relation of belief**

MR1: When ME1 of modified extensive structure is fulfilled by a metric, then it also satisfies MR1.  $BEM_I$  fulfills ME1 of modified extensive structure (proved above) and therefore,  $BEM_I$  satisfies MR1.

MR2: If ME1 of modified extensive structure is satisfied by a metric then that metric satisfies MR2.  $BEM_I$  fulfills ME1 of modified extensive structure and therefore, it satisfies MR2.

MR3: Suppose that all the changes of the table  $Diff_{3,4}$  are included in  $Diff_{31,2}$ , then  $BEM_I(Diff_{1,2}) \geq BEM_I(Diff_{33,4})$ . Thus, this axiom is satisfied.

MR4: In order to prove MR4, let all the changes of the table  $Diff_{3,4}$  are included in  $Diff_{1,2}$  and  $Diff_{1,2} \cap Diff_{5,6} = \emptyset$ . Then it needs to be proved that  $BEM_I(Diff_{3,4}) \geq BEM_I(Diff_{1,2}) \Rightarrow BEM_I(Diff_{3,4} \cup Diff_{5,6}) \geq BEM_I(Diff_{1,2} \cup Diff_{5,6})$  needs to be proved. Due to the fact that  $BEM_I(Diff_{3,4}) \geq BEM_I(Diff_{1,2})$  and that there are no common changes between  $Diff_{3,4}$  and  $Diff_{5,6}$ , the value of  $BEM_I(Diff_{3,4} \cup Diff_{5,6})$  will be more than  $BEM_I(Diff_{1,2} \cup Diff_{5,6})$ . This proves that the metric  $BEM_I$  satisfies MR4.

MR5: This axiom is also satisfied because changes in a process cannot be less than 0.

Therefore,  $BEM_I$  fulfills the modified relation of belief. Thus,  $BEM_I$  is a measure above the level of the ordinal scale.

Other metrics i.e.  $BEM_E$ ,  $BUMEP$  and  $BUMEN$  have also been validated using Zuse framework. All the metrics are found to be above the ordinal scale. The results of applying the framework to all metrics are shown in Table 4.10.

Table 4.10: Summary of formal validation of metrics of a WS-BPEL process

<b>Metrics/Axioms</b>	<b>BEM<sub>I</sub></b>	<b>BEM<sub>E</sub></b>	<b>BUME<sub>P</sub></b>	<b>BUME<sub>N</sub></b>
<b>ME1</b>	Y	Y	Y	Y
<b>ME2</b>	Y	Y	Y	Y
<b>ME3</b>	Y	Y	Y	Y
<b>ME4</b>	Y	Y	Y	Y
<b>ME5</b>	N	N	N	N
<b>ME6</b>	N	N	N	N
<b>IC1</b>	N	N	N	N
<b>IC2</b>	N	N	N	N
<b>IC3</b>	N	N	N	N
<b>IC4</b>	N	N	N	N
<b>MR1</b>	Y	Y	Y	Y
<b>MR2</b>	Y	Y	Y	Y
<b>MR3</b>	Y	Y	Y	Y
<b>MR4</b>	Y	Y	Y	Y
<b>MR5</b>	Y	Y	Y	Y
<b>Scale</b>	<b>Above ordinal</b>	<b>Above ordinal</b>	<b>Above ordinal</b>	<b>Above ordinal</b>

#### 4.7 Summary

In this chapter, we proposed metrics for an evolving process. Perspectives of both the provider as well as the consumer have been considered while proposing metrics.

For the provider, firstly, in order to understand what types of changes have occurred in a WS-BPEL process, two categories of changes are proposed: Internal and External changes. Subsequently, to estimate the amount of changes, metrics are defined for each of these categories. The corresponding metrics are Internal Evolution Metric (BEM<sub>I</sub>) and External Evolution Metric (BEM<sub>E</sub>). Also, these metrics truly reflect the cohesiveness of changes in a process vis-a-vis changes in services.

For the consumer of the process, BPEL Process Usefulness Metric under Evolution in a positive sense (BUME<sub>P</sub>) and BPEL Process Usefulness Metric under Evolution in a negative sense (BUME<sub>N</sub>) are proposed. They are defined for computing the impact on the usefulness for the consumer as a process evolves.

All the proposed metrics have linear time complexity. The metrics are validated theoretically using Zuse framework and are found to be above the ordinal scale.

# Chapter 5

## Metrics for an Evolving Composite Service - Choreography

---

### 5.1 Introduction

Service composition can be achieved through choreography [79][80]. Choreography refers to the collaborations between interacting services. Continuing with our approach of studying changes via metrics, in this chapter we propose metrics for an evolving composite service which is composed via choreography (WS-CDL process).

A choreography involves peer-to-peer interactions between participants (web services) having different roles [11]. The different kinds of changes that can occur in choreography are addition, deletion, modification, split or merge of the participants and interactions [34]–[37]. Among these changes, there are some changes which are additive in nature i.e. addition, modification and split and some are subtractive in nature i.e. deletion and merge. Both of these changes are considered while proposing metrics for a choreography as it evolves.

We propose metrics to measure changes in the entities (participant/role/interaction) of a choreography. These metrics take into account each kind of change in these entities i.e. additive changes and subtractive changes. To do so, two metrics are proposed, one is Additive Evolution Metric ( $AEM^+$ ) and the other one is Subtractive Evolution Metric ( $SEM^-$ ). The former metric is used to measure the changes which increase the number of entities that take part in the choreography and the latter gives a measure of the decrease in the number of entities that were participating. Evolution Metric (EM) is proposed to quantify the total evolution by taking into account both kinds of changes i.e. additive as well as subtractive changes. A case study is used to empirically show the applicability of the proposed metrics. To theoretically validate the metrics, Zuse Framework has been used.

The layout of this chapter is as follows. Section 5.2 defines the WS-CDL process. In section 5.3, evolution metrics are discussed. Experiments and analysis are shown using a case study are presented in section 5.4. Finally, the chapter is concluded in section 5.5.

## 5.2 WS-CDL Process

A WS-CDL process is used to compose different web services. Each web service is a participant which interacts with each other to attain a global goal of the choreography. WS-CDL is an XML based description language and not an executable language. Therefore, the aim of the WS-CDL process is to describe peer-to-peer collaborations (interactions) among the participants [11]. It represents the global perspective of the participant's interactions rather than from the perspective of a single participant. It aims to serve the purpose of defining abstract interactions among participants (services). The different entities of a WS-CDL process are listed in Table 5.1.

**Table 5.1:** WS-CDL process entities

S. No	Entity	Description
1	Interaction	a realization of the collaboration between two peers
2	Roles	Interactions takes place between different roles
3	Participants	a physical entity which realize the interaction
4	Relationships	declares the intention of the interaction
5	Information types	declares type of the variables to be used in the choreography
6	Tokens and Locators	Token is an alias for an information type and Locator is used to locate a particular token generally in an XPath query
7	Channels	a medium for the interaction to happen
8	Choreographies	define how the interaction would occur, in sequence/parallel/loop

Clearly, the above discussed entities could be divided into two parts: one part is of the entities from serial no. 1 to 3 which describes the collaborations and the other part is of entities from serial no. 4 to 8 which are used in describing on how the collaborations would take place. Therefore, our focus is on the first part as they are the principal components of the WS-CDL process.

In the next section, we propose metrics for a WS-CDL process under evolution.

### **5.3 Proposed Metrics for a Composite Service - Choreography**

Essentially, a choreography aims to accomplish multi-party interactions to achieve a global goal. WS-CDL document consists of many entities such as interaction, roleType, variable, token, informationType, while etc. From an external global perspective, a choreography can be viewed essentially as interactions between participants in a certain role. When a choreography undergoes changes, it can be viewed from two different perspectives. One is the structure of the WS-CDL document, the other is from the perspective of the interactions. In this work, changes in the interactions are addressed. These interactions do not happen in a vacuum but between participants having some roles. Thus, changes in the entities: participantType, roleType, interaction, are the focus of the study.

In a WS-CDL process, there are participants with roles which interact with each other. There could be changes in the existing participants such as deletion of their existing roles or interactions or there could be addition of new participants with new roles and interactions. In other words, changes which occur in these entities could be in the existing participants/roles/interaction or in newly added participants/roles/interactions.

As brought out in section 5.1, metrics are proposed for the above discussed changes. The changes themselves can be addition, modification, split, deletion and merge. The metrics take into account these changes for the entities under focus.

We use these metrics to measure the changes across different versions of a WS-CDL process. Let there be a version  $i$  of a process,  $CDL_i$ . When it changes, its new version  $CDL_{i+1}$  is created. A Difference Table,  $Diff_{i,i+1}$ , is maintained which contains all changes when  $CDL_i$  evolves to version  $CDL_{i+1}$ .

Here, the following terms and acronyms are used.

Peer: participant involved in an interaction

Already Interacting Peer (AIP): peer involved in interactions in current version of choreography

New Interacting Peer (NIP): peer involved in newly added interactions in current version of choreography

Old Participant (OP): Participant in the previous version of the choreography

New Participant (NP): Newly added participant in the current version of choreography

IA/RA: number of Interactions/Roles added

ID/RD: number of Interactions/Roles deleted

IMo<sup>+</sup>: number of Interactions/Roles modified in which an exchange is added

IMo<sup>-</sup>: number of Interactions/Roles modified in which an exchange is deleted

IS: number of split Interactions

IMe: number of merged Interactions

We use below notations throughout the metrics computation in this chapter.

Notation-(a): Peers which were already interacting with each other are denoted as AIP<sub>i</sub> and AIP<sub>j</sub>.

Notation-(b): Peers which were not interacting with each other are denoted as NIP<sub>i</sub> and NIP<sub>j</sub>.

Notation-(c): Newly added participant start interacting with another newly added participant - denoted as NP<sub>i</sub> and NP<sub>j</sub>.

Notation-(d): Newly added participant start interacting with an old participant - denoted as NP<sub>i</sub> and OP<sub>j</sub>.

Notation-(e): Old participant starts interacting with another old participant - denoted as OP<sub>i</sub> and OP<sub>j</sub>.

We propose below metrics pertaining to changes in the interaction entity, role entity and participant entity, in turn.

### 5.3.1 Metrics for interaction entity

Changes in interactions, when considered independently, occur in existing participants with their roles. An interaction could be added/deleted/modified/split/merged. Therefore, there are five components in the metrics computation for the interactions.

#### Additions

There are two ways in which interactions can be added. These can be between peers mentioned in Notation-(a) or between peers in Notation-(b) as defined in in section 5.3. Therefore, the metric for additions in interactions consists of two parts. The metric is defined below.

$$\Delta_{\text{Interaction}}^{\text{Addition}} = \frac{\{\sum_{i=1}^n \sum_{j=1}^m (IA_{AIP_i, AIP_j})\}}{|N_1|} + \frac{\{\sum_{i=1}^n \sum_{j=1}^m (IA_{NIP_i, NIP_j})\}}{|N_2|} \quad (5.1)$$

A peer can add several interactions with more than one peer. Thus, for each peer, these added interactions are counted with each interacting peer. The numerator of both parts in Equation (5.1) are  $IA_{AIP_i, AIP_j}$  and  $IA_{NIP_i, NIP_j}$  which are the number of interactions added between peers mentioned in Notation-(a) and Notation-(b) respectively.  $N_1$  and  $N_2$  are the number of peers which have participated in the interactions in the numerator.

#### Deletions

Deletions can occur only between the peers defined in Notation-(a) in section 5.3.

The metric for deletions, therefore, is defined as

$$\Delta_{\text{Interaction}}^{\text{Deletion}} = \frac{\{\sum_{i=1}^n \sum_{j=1}^m (ID_{AIP_i, AIP_j})\}}{|N_3|} \quad (5.2)$$

The numerator of Equation (5.2) contains the count of the deleted interactions between peers mentioned in Notation-(a).  $N_3$  is the number of peers (mentioned in Notation-(a)) which have participated in the interactions in the numerator.



## Splits

Interactions can be split among the peers defined in Notation-(a) in section 5.3. Metric for split interactions is as follows.

$$\Delta_{\text{Interaction}}^{\text{Split}} = \frac{\{ \sum_{i=1}^n \sum_{j=1}^m (\frac{M_1}{IS_{AIP_i, AIP_j}}) \}}{|N_4|} \quad (5.3)$$

An interaction can be split in more than one interaction between any two peers which are defined in Notation-(a). Therefore, the numerator of Equation (5.3) contains both the count of interactions which are split ( $IS_{AIP_i, AIP_j}$ ) as well as the count in which each of these interactions is split ( $M_1$ ).  $N_4$  is the number of peers as mentioned in Notation-(a) which have split interactions.

## Merge

Peers defined in Notation-(a) in section 5.3, can merge interactions between them. The metric for merged interactions is defined as follows.

$$\Delta_{\text{Interaction}}^{\text{Merge}} = \frac{\{ \sum_{i=1}^n \sum_{j=1}^m (IME_{AIP_i, AIP_j}) \}}{|N_5|} \quad (5.4)$$

A peer can merge several of its interactions with the peers with which it interacts. The numerator of the above equation contains  $IME_{AIP_i, AIP_j}$  as the number of interactions that are merged between peers as mentioned in (a).  $N_5$  is the number of peers mentioned in Notation-(a) which have merged their interactions.

## Modifications

Within an interaction, there is an exchange of information (send and/or receive) among peers. When an interaction is modified, the exchanges within an interaction are either added or deleted. The metric defined below for modifications of interactions considers these exchanges.

$$\Delta_{\text{Interaction}}^{\text{Modification}^+} = \frac{\{\sum_{i=1}^n \sum_{j=1}^m (\text{IMo}^+_{\text{AIP}_i, \text{AIP}_j})\}}{|\text{N}_6|} \quad (5.5)$$

$$\Delta_{\text{Interaction}}^{\text{Modification}^-} = \frac{\{\sum_{i=1}^n \sum_{j=1}^m (\text{IMo}^-_{\text{AIP}_i, \text{AIP}_j})\}}{|\text{N}_7|} \quad (5.6)$$

$\text{IMo}^+_{\text{AIP}_i, \text{AIP}_j}$  and  $\text{IMo}^-_{\text{AIP}_i, \text{AIP}_j}$  is the number of interactions that are modified for each peer (mentioned in Notation-(a)) by adding and deleting an exchange respectively.  $\text{N}_6$  and  $\text{N}_7$  are such respective number of peers.

Next, we present metrics for a WS-CDL process pertaining to evolution in role entity.

### 5.3.2 Metrics for role entity

A role could be added or deleted in a WS-CDL process but not split/merged. A role has a behavior which could be defined using references to a WSDL description binding or is optional in a process. A role can have a binding point to different service descriptions. Therefore, changes to the behavior within a roleType are possible and hence in a roleType. But this modification neither increases nor decreases the quantity of evolution in a process, thus modification of roleType is not taken into account while computing the metrics for roles. Hence, there are two components in the metrics computation for evolution in the roles i.e. one for additions and other for deletions. Both components consider changes that occur between existing participants with their roles.

Metrics are defined for both above discussed changes in roles.

#### Additions

Roles can be added between peers mentioned in Notation-(a) and Notation-(b) in section 5.3. In the metric defined below, the first part refers to roles added for Notation-(a) and second part refers to Notation-(b). The metric is computed as follows.

$$\Delta_{\text{Role}}^{\text{Addition}} = \frac{[\sum_{i=1}^n \sum_{j=1}^m \{ \text{RA}_{\text{AIP}_i, \text{AIP}_j} * (\sum_{k=1}^l \mathbf{M}_1) \}]}{|\mathbf{N}_8|} + \frac{[\sum_{i=1}^n \sum_{j=1}^m \{ \text{RA}_{\text{NIP}_i, \text{NIP}_j} * (\sum_{k=1}^l \mathbf{M}_2) \}]}{|\mathbf{N}_9|} \quad (5.7)$$

$\text{RA}_{\text{AIP}_i, \text{AIP}_j}$  is the number of roles that are added between each peer as mentioned in Notation-(a). For each role, thus added, one or more interactions are also added for each of these peers.  $\mathbf{M}_1$  is the total number of such interactions. A similar computation is expressed in the second part of the equation. However, the second part refers to roles added for peers as mentioned in Notation-(b).

### Deletions

Interacting peers, mentioned in Notation-(a), can delete their roles. The metric is defined as follows.

$$\Delta_{\text{Role}}^{\text{Deletion}} = \frac{[\sum_{i=1}^n \sum_{j=1}^m \{ \text{RD}_{\text{AIP}_i, \text{AIP}_j} * (\sum_{k=1}^l \mathbf{M}_3) \}]}{|\mathbf{N}_{10}|} \quad (5.8)$$

$\text{RD}_{\text{AIP}_i, \text{AIP}_j}$  is the number of roles that are deleted between peers mentioned in Notation-(a).  $\mathbf{M}_3$  is the number of interactions deleted for  $\text{RD}_{\text{AIP}_i, \text{AIP}_j}$ .  $\mathbf{N}_{10}$  is the number of such peers.

Now, consider the participant entity to propose metrics.

### 5.3.3 Metrics for participant entity

Participants are either added or deleted. A new participant is added with new roles and interactions and an old participant is deleted with old roles and interactions. A participant has a role which could be modified. But this modification neither increases nor decreases the quantity of evolution in a process, thus modification of participant is not taken into account. A participant can neither be split nor merged. Therefore, there are two components in the metrics computation for evolution in the participants i.e. for addition and deletion.

Metrics are defined for both additions as well as deletions in participants.

## Additions

The metric for additions in participants has two parts. The first part defines the case when newly added participants interact with each other with their newly added roles as mentioned in Notation-(c). The second part shows that when a newly added participant interacts with the old participant in the choreography mentioned in Notation-(d). The metric is computed as follows.

$$\Delta_{\text{Participant}}^{\text{Addition}} = \frac{(\sum_{i=1}^n (\text{RANP}_i * (\sum_{j=1}^m \mathbf{M}_4)))}{|\mathbf{N}_{11}|} + \frac{(\sum_{i=1}^n (\text{RAOP}_i * (\sum_{j=1}^m \mathbf{M}_5)))}{|\mathbf{N}_{12}|} \quad (5.9)$$

In the above equation,  $\text{RANP}_i$  and  $\text{RAOP}_i$  are the number of added roles for the new and old participant respectively.  $\mathbf{M}_4$  and  $\mathbf{M}_5$  are the number of interactions added for each role added for participants of Notation-(c) and Notation-(d) respectively.  $\mathbf{N}_{11}$  and  $\mathbf{N}_{12}$  are the number of participants in the numerators of each part of the above equation.

## Deletions

An old participant can be deleted in the choreography. Equation (5.10) defines the metric for deletions.

$$\Delta_{\text{Participant}}^{\text{Deletion}} = \frac{\{ (\sum_{i=1}^n (\text{RDOP}_i * (\sum_{j=1}^m \mathbf{M}_6)) \}}{|\mathbf{N}_{13}|} \quad (5.10)$$

$\text{RDOP}_i$  is the number of deleted roles for old participants as mentioned in Notation-(e).  $\mathbf{M}_6$  is the number of deleted interactions for these participants.  $\mathbf{N}_{13}$  is the number of old participants in the numerator.

Metrics proposed for each entity for each change is aggregated by using weights  $\gamma_I$  as .2,  $\gamma_R$  as .3 and  $\gamma_P$  as .5 as per their contribution to the amount of evolution in the choreography.

$$\Delta^{\text{Addition}} = \gamma_I * \Delta_{\text{Interaction}}^{\text{Addition}} + \gamma_R * \Delta_{\text{Role}}^{\text{Addition}} + \gamma_P * \Delta_{\text{Participant}}^{\text{Addition}} \quad (5.11)$$

$$\Delta^{\text{Split}} = \Delta_{\text{Interaction}}^{\text{Split}} \quad (5.12)$$

$$\Delta^{\text{Merge}} = \Delta_{\text{Interaction}}^{\text{Merge}} \quad (5.13)$$

$$\Delta^{\text{Deletion}} = \gamma_I * \Delta_{\text{Interaction}}^{\text{Deletion}} + \gamma_R * \Delta_{\text{Role}}^{\text{Deletion}} + \gamma_P * \Delta_{\text{Participant}}^{\text{Deletion}} \quad (5.14)$$

### 5.3.4 Additive/Subtractive Evolution Metric (AEM<sup>+</sup>/SEM<sup>-</sup>)

As brought out in section 5.1, some changes are additive in nature and some are subtractive. Therefore, the metrics are combined under two categories using Equation (5.5), Equation (5.6) and equations from Equation (5.11) to Equation (5.14).

Let CDL<sub>1</sub>, CDL<sub>2</sub>, ....., CDL<sub>i</sub>, CDL<sub>i+1</sub> be the versions of the WS-CDL process and the changes are maintained in the difference tables Diff<sub>1,2</sub>, Diff<sub>2,3</sub>, ....., Diff<sub>i,i+1</sub>.

a) Additive Evolution Metric:

$$\text{AEM}^+ (\text{Diff}_{i,i+1}) = \Delta^{\text{Addition}} + \Delta^{\text{Split}} + \Delta_{\text{Interaction}}^{\text{Modification}^+} \quad (5.15)$$

b) Subtractive Evolution Metric:

$$\text{SEM}^- (\text{Diff}_{i,i+1}) = \Delta^{\text{Deletion}} + \Delta^{\text{Merge}} + \Delta_{\text{Interaction}}^{\text{Modification}^-} \quad (5.16)$$

Whenever evolution occurs, some changes may increase the number of entities and some may decrease. The Equation (5.15) and Equation (5.16) help to determine the kind and quantity of changes with respect to additive or subtractive changes in the process.

### 5.3.5 Evolution Metric (EM)

Evolution Metric (EM): provides a measure for the net amount of evolution occurred in the process after it evolves from CDL<sub>i</sub> to CDL<sub>i+1</sub>. It is computed as

$$\begin{aligned}
EM(\text{Diff}_{i,i+1}) &= AEM^+(\text{Diff}_{i,i+1}) - SEM^-(\text{Diff}_{i,i+1}) && \text{when } AEM^+ > SEM^- \\
&= SEM^-(\text{Diff}_{i,i+1}) - AEM^+(\text{Diff}_{i,i+1}) && \text{when } SEM^- > AEM^+
\end{aligned}
\tag{5.17}$$

In the next section, we perform experiments on a case scenario for a WS-CDL process and analyze the resultant metric values computed for this process.

#### 5.4 Experiments and Analysis

The proposed metrics are evaluated using a case study of a WS-CDL process. Different versions of this process are made and then metrics are computed for each version of the process. The changes are made for additions and deletions as additions and deletions are common changes among all the three entities i.e. interaction, role and participant. Whenever a new version of the process is created, a new difference table is created. The information of the changes that have occurred is inserted into the table.

A process describes the ordering of the collaborations between the participants, therefore, UML diagrams are used to illustrate the case study. Consider a purchase order WS-CDL Process in which there are three participants: Seller, Buyer and Customer Support with buyer, seller and support as their respective roles. The interactions among these three are shown in Figure 5.1.

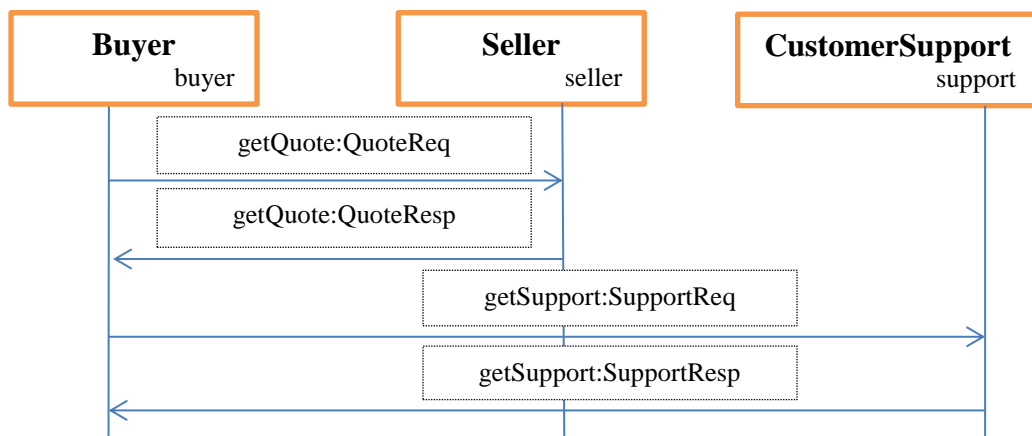


Figure 5.1: WS-CDL Process Version 1

Let the process Version 1 change to Version 2 after adding interactions between Buyer and Seller. The changed version is shown in Figure 5.2.

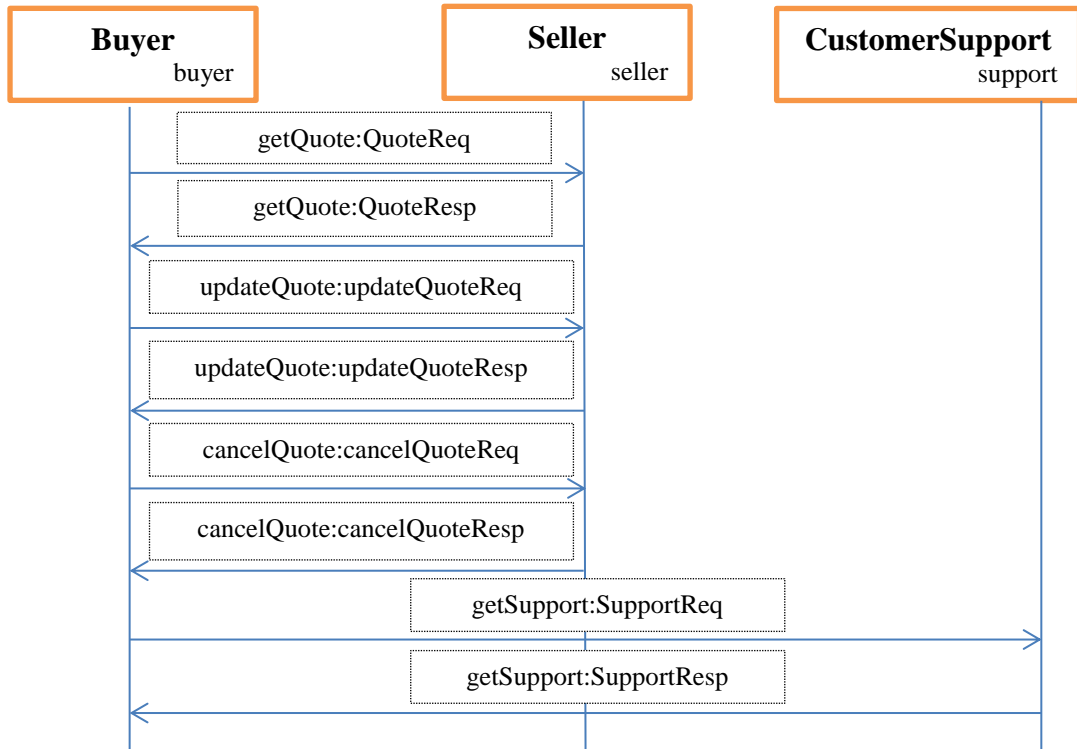


Figure 5.2: WS-CDL Process Version 2

Table 5.2 shows the evolution data.

Table 5.2: Evolution description of version 1 to version 2 of the process

Peers/ Participants (AIP/NIP/ NP/OP)	Addition ( $N_1/N_2/N_8/N_9/N_{11}/N_{12}$ )			Deletion ( $N_3/N_{10}/N_{13}$ )			Modification ( $N_6/N_7$ )	Split ( $N_4$ )	Merge ( $N_5$ )
	Interactions $IA_{AIP,AIP}/IA_{NIP,MIP}$	Roles $RA_{AIP,AIP}/RA_{NIP,NIP}$ $/DA_{AIP,AIP}/DA_{NIP,NIP}$	Role-Interactions ( $M_1/M_2/M_4/M_5$ )	Interactions $ID_{AIP,AIP}$	Roles $RD_{AIP,AIP}/RD_{OP}$	Role-Interactions $M_3/M_6$	Interactions $IMo^+_{AIP,AIP}/IMo^-_{AIP,AIP}$	Interactions $IS_{AIP,AIP}$	Interactions $IMe_{AIP,AIP}$
$AIP_{Buyer}$	2			0			0	0	0
$AIP_{Seller}$	2	0	0	0	0	0	0	0	

Let Version 3 be created after adding roles: sponser and advertiser for Seller and CustomerSupport respectively. The interactions are also added for these newly added roles. In addition, an interaction is also added between Buyer and CustomerSupport. All these changes are depicted in Figure 5.3. Table 5.3 shows the data of evolution from Version 2 to Version 3.

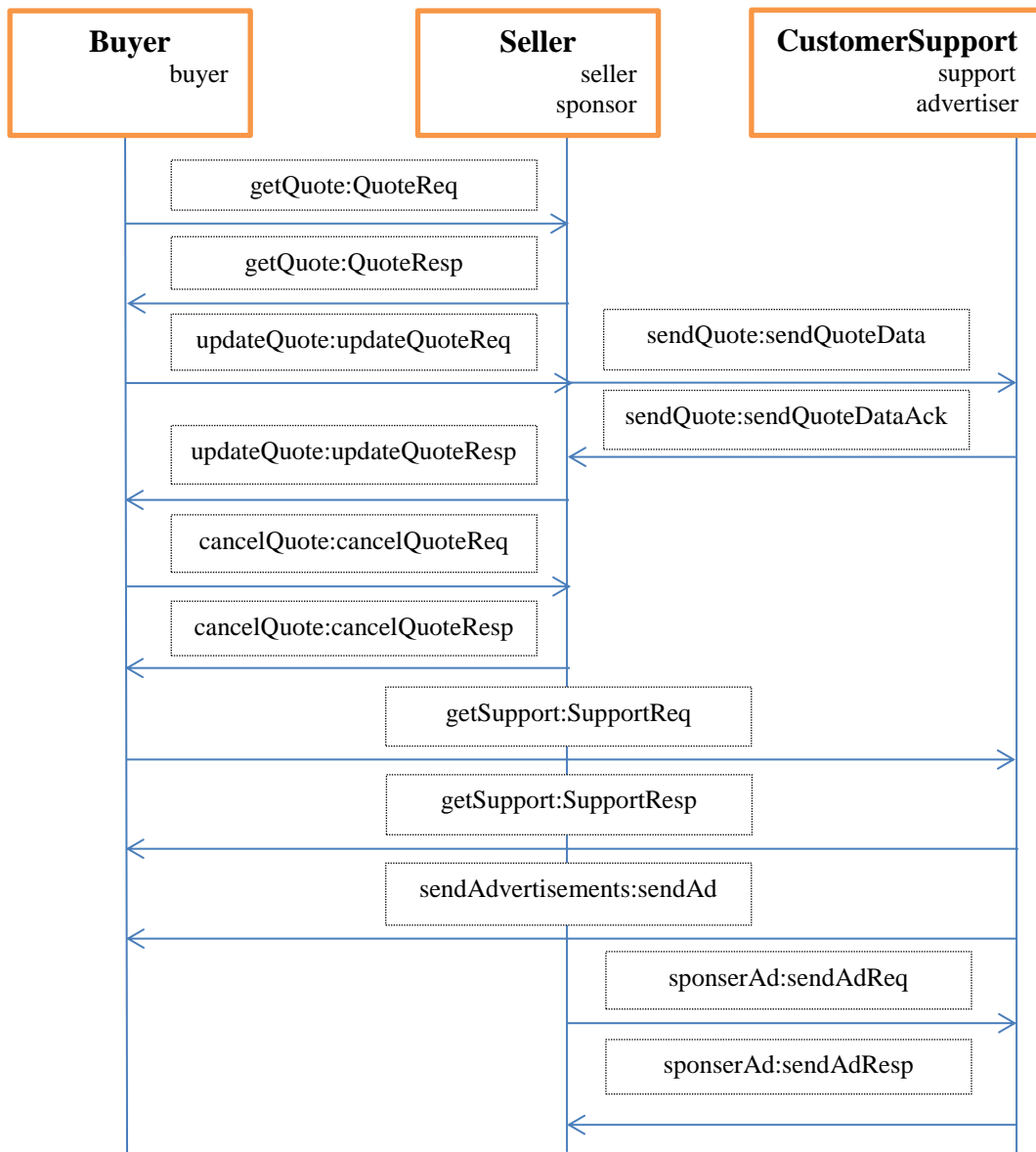


Figure 5.3: WS-CDL Process Version 3

Version 4 of the CDL process is created by adding a new participant (CreditAgency). In addition, a role (support) for CustomerSupport is deleted along with its interaction. Changed version is shown in Figure 5.4.



Table 5.3: Evolution description of version 2 to version 3 of the process

Peers/ Participants (AIP/NIP/ NP/OP)	Addition (N <sub>1</sub> / N <sub>2</sub> /N <sub>8</sub> /N <sub>9</sub> /N <sub>11</sub> / N <sub>12</sub> )			Deletion (N <sub>3</sub> /N <sub>10</sub> / N <sub>13</sub> )			Modification (N <sub>6</sub> /N <sub>7</sub> )	Split (N <sub>4</sub> )	Merge (N <sub>5</sub> )
	Interactions IA <sub>AIP,AIP</sub> /IA <sub>NIP,NIP</sub>	Roles RA <sub>AIP,AIP</sub>	Role-Interactions (M <sub>1</sub> /M <sub>2</sub> /M <sub>4</sub> /M <sub>5</sub> )	Interactions ID <sub>AIP,AIP</sub>	Roles RD <sub>AIP,AIP</sub>	Role-Interactions M <sub>3</sub> /M <sub>6</sub>	Interactions IM <sub>O<sub>AIP,AIP</sub><sup>+</sup></sub> /IM <sub>O<sub>AIP,AIP</sub><sup>-</sup></sub>	Interactions IS <sub>AIP,AIP</sub>	Interactions IM <sub>e<sub>AIP,AIP</sub></sub>
AIP <sub>Buyer</sub> ,	2			0			0	0	0
AIP <sub>CustomerSupport</sub>	0	1	1	0	0	0	0	0	0
NIP <sub>Seller</sub> ,	2			0			0	0	0
NIP <sub>CustomerSupport</sub>	1	1	1	0	0	0	0	0	0

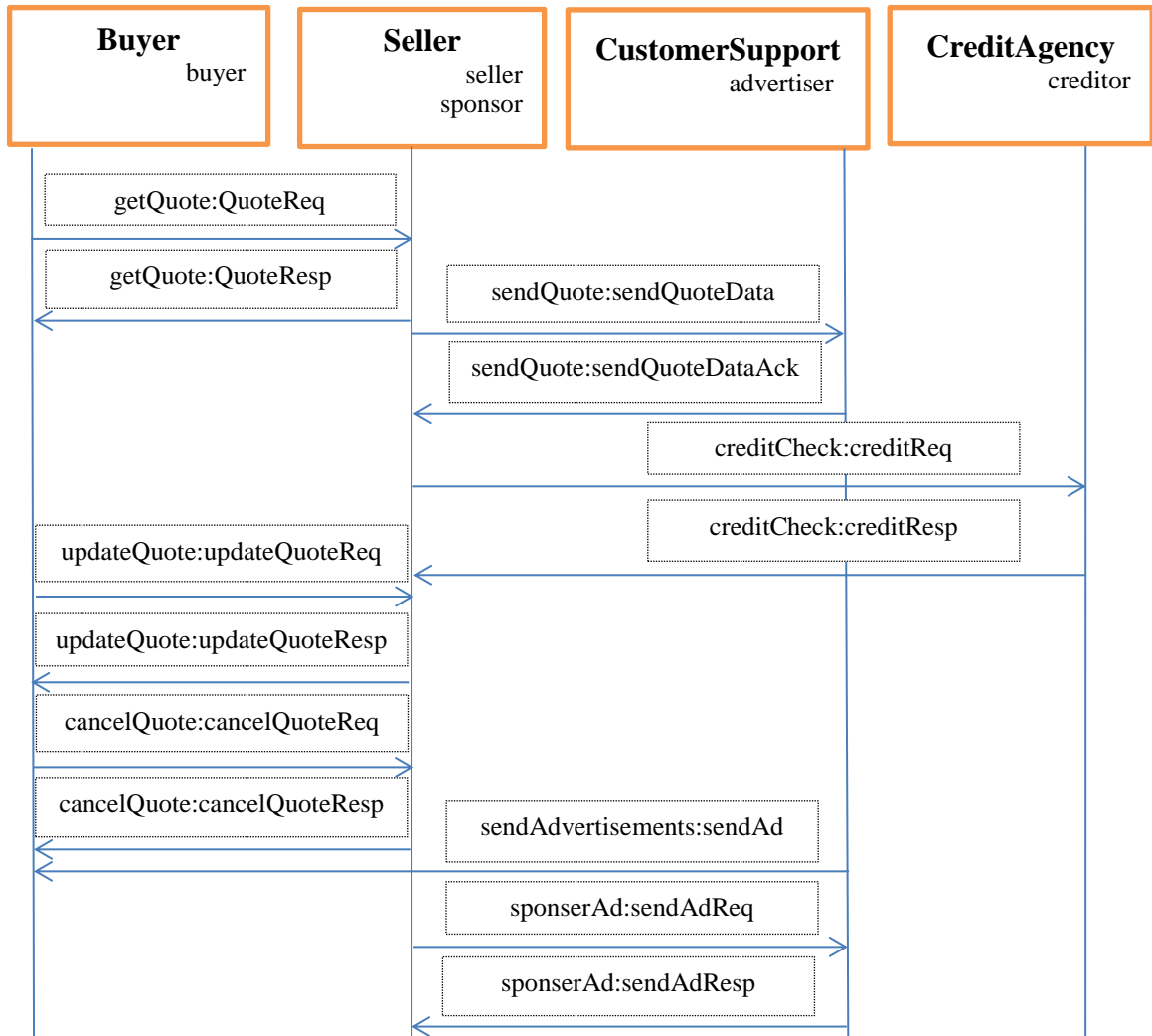


Figure 5.4: WS-CDL Process Version 4

The evolution data is shown in Table 5.4.

Table 5.4: Evolution description of version 3 to version 4 of the process

Peers/ Participants (AIP/NIP/ NP/OP)	Addition ( $N_1/N_2/N_8/N_9/N_{11}/N_{12}$ )			Deletion ( $N_3/N_{10}/N_{13}$ )			Modification ( $N_6/ N_7$ )	Split ( $N_4$ )	Merge ( $N_5$ )
	Interactions $IA_{AIP,AIP} / IA_{NIP,MIP}$	Roles $RA_{AIP,AIP} / RA_{NIP,NIP}$	Role-Interactions ( $M_1/M_2/M_4/M_5$ )	Interactions $ID_{AIP,AIP}$	Roles $RD_{AIP,AIP} / RD_{OP}$	Role-Interactions $M_3/M_6$	Interactions $IMO_{AIP,AIP}^+ / IMO_{AIP,AIP}^-$	Interactions $IS_{AIP,AIP}$	Interactions $IME_{AIP,AIP}$
CreditAgency, Seller	0	1	1	0	0	0	0	0	0
AIP <sub>Buyer</sub> ,	0			2			0	0	0
AIP <sub>CustomerSupport</sub>	0	0	0	0	1	1	0	0	0

In Figure 5.5, one participant (CreditAgency) is deleted. There are two additions: one is role (support) added between Buyer and CustomerSupport along with its corresponding interaction and the other one is the addition of two new participants (Accounts and Shipper). All these changes give rise to Version 5.



Figure 5.5: WS-CDL Process Version 5

The evolution data is shown in Table 5.5.

Table 5.5: Evolution description of version 4 to version 5 of the process

Peers/ Participants (AIP/NIP/NP/OP)	Addition ( $N_1/N_2/N_8/N_9/N_{11}/N_{12}$ )			Deletion ( $N_3/N_{10}/N_{13}$ )			Modification ( $N_6/N_7$ )	Split ( $N_4$ )	Merge ( $N_5$ )
	Interactions $IA_{AIP,AIP}/IA_{NIP,NIP}$	Roles $RA_{AIP,AIP}$	Role-Interactions ( $M_1/M_2/M_4/M_5$ )	Interactions $ID_{AIP,AIP}$	Roles $RD_{AIP,AIP}/RD_{OP}$	Role-Interactions $M_3/M_6$	Interactions $IMO_{AIP,AIP}/IMO_{AIP,AIP}$	Interactions $IS_{AIP,AIP}$	Interactions $IME_{AIP,AIP}$
CreditAgency, Seller	0			2			0	0	0
Seller, Accounts	0	0	0	0	1	1	0	0	0
Seller, Shipper	2			0			0	0	0
AIP <sub>Buyer</sub> , AIP <sub>CustomerSupport</sub>	0	1	1	0	0	0	0	0	0

The metrics for all the versions of the WS-CDL process, computed above, are tabulated in Table 5.6.

Table 5.6: Metric values of the WS-CDL process

Serial No.	Version	Evolution Description	AEM <sup>+</sup>	SEM <sup>+</sup>	EM
1	1,2	Interactions added between AIP, AIP	.20	0	.20
2	2,3	Roles added between AIP, AIP and NIP,NIP and Interactions added between NIP,NIP	.40	0	.40
3	3,4	Participant added Role deleted	.25	.15	.10
4	4,5	Participants added Participant deleted Role added	.48	.25	.23

Table 5.6 depicts the comprehensive analysis of the changes in choreography. AEM<sup>+</sup> values at Serial No. 1 and 2 contains the changes which are additive in nature. Serial No. 3 and 4 of the table contains both kinds of changes i.e. additive as well as

subtractive changes. This is shown by their  $AEM^+$  and  $SEM^-$  metric values. It can be seen that the metric EM at Serial No. 1 & 4 are approximately same. However, there are only additive changes at Serial No. 1 and 2. In other words, a clear picture of the exact evolution is brought out by  $AEM^+$  and  $SEM^-$  values. Therefore, the proposed metrics gives the total idea of the exact evolution that has taken place in the choreography.

### **5.5 Time Complexity**

A table is created in the database whenever a process evolves and results in the creation of its new version. All the evolution data between the process versions is then stored in the table. Each row of the table contains evolution data for each entity of the CDL process. The table contains six columns which are listed below.

- 1) CDL process versions (evolution data between these two versions is stored)
- 2) Entity (lists the entities of a process)
- 3) Peer/Participant (entity participating in the choreography)
- 4) Number of peer/participant (count of interacting peers/participants)
- 5) Change (kind of change in the entity)
- 6) Number of changes (the number additions/ deletions/ modifications/ split/merge for each entity)

Next, all the metrics are computed by sequentially accessing the information stored in the corresponding table. Thus, when there are n number of rows, then the metrics are computed in linear time i.e. $O(n)$ .

Tables from Table 5.2 to Table 5.5 shows the sample data stored for different process versions.

### **5.6 Metrics Formal Validation**

All the proposed metrics are theoretically validated using Zuse framework as given in Table 3.8 in chapter 3. We, now, present formal validation of  $AEM^+$ .

## AEM<sup>+</sup> Metric Formal Validation

Let  $CDL_1, CDL_2, \dots, CDL_i, CDL_{i+1}$  be the versions of the WS-CDL process. The changes between any two versions, say  $CDL_x, CDL_{x+1}$  of a process are captured in a  $Diff_{x,x+1}$  table. Let  $Diff_{x,x+1}$  and  $Diff_{y,y+1}$  denote the table containing the information of all the changes between these versions. Let  $Diff$  be the set of all tables for the process that store information of changes across its versions.

The measure  $AEM^+$  is a mapping:  $AEM^+: Diff \rightarrow \mathbb{R}$  such that the following holds for all tables  $Diff_{x,x+1}, Diff_{y,y+1} \in Diff$ :  $Diff_{x,x+1} \succ= Diff_{y,y+1} \Leftrightarrow AEM^+(Diff_{x,x+1}) \succ= AEM^+(Diff_{y,y+1})$ .

Here, the concatenation operation for combination rule is denoted as follows.

$$AEM^+(Diff_{x,x+1} \circ Diff_{y,y+1}) = AEM^+(Diff_{x,x+1} \cup Diff_{y,y+1})$$

where  $Diff_{x,x+1} \cup Diff_{y,y+1}$  is the table containing all the distinct changes in the two tables  $Diff_{x,x+1}$  and  $Diff_{y,y+1}$ .

## AEM<sup>+</sup> and the Modified Extensive Structure

ME1: The binary relation  $\succ=$  is known to be weak order when it is transitive and complete. Let  $Diff_{1,2}, Diff_{3,4}$  and  $Diff_{5,6}$  be the three tables where  $Diff_{1,2}, Diff_{3,4}, Diff_{5,6} \in Diff$ . It must be true that either  $AEM^+(Diff_{1,2}) \succ= AEM^+(Diff_{3,4})$  or  $AEM^+(Diff_{3,4}) \succ= AEM^+(Diff_{1,2})$ . Thus, property of completeness is fulfilled. Now, consider the transitivity property. If  $AEM^+(Diff_{1,2}) \succ= AEM^+(Diff_{3,4})$  and  $AEM^+(Diff_{3,4}) \succ= AEM^+(Diff_{5,6})$  then it is obvious that  $AEM^+(Diff_{1,2}) \succ= AEM^+(Diff_{5,6})$ . Thus, transitive property is also accomplished. Therefore,  $AEM^+$  fulfills ME1.

ME2: The positivity of the metric implies that the value of the metric when two tables are combined is bound to be greater than the metric for each individual table. Thus,  $AEM^+(Diff_{1,2} \circ Diff_{3,4}) \succ= AEM^+(Diff_{1,2})$ . Therefore, ME2 is fulfilled.

ME3: Applying the weak associativity rule to the proposed metric, the formulation of the rule becomes,  $AEM^+ (Diff_{1,2} \circ (Diff_{3,4} \circ Diff_{5,6})) = AEM^+ ((Diff_{1,2} \circ Diff_{3,4}) \circ Diff_{5,6})$ . This means that  $AEM^+ (Diff_{1,2} \cup (Diff_{3,4} \cup Diff_{5,6})) = AEM^+ ((Diff_{1,2} \cup Diff_{3,4}) \cup Diff_{5,6})$ . It is obvious that this axiom is fulfilled because union operation is associative.

ME4: The weak commutative axiom is stated as  $AEM^+ (Diff_{1,2} \circ Diff_{3,4}) = AEM^+ (Diff_{3,4} \circ Diff_{1,2})$ . This means that  $AEM^+ (Diff_{1,2} \cup Diff_{3,4}) = AEM^+ (Diff_{3,4} \cup Diff_{1,2})$ . Therefore, this axiom is fulfilled because union operation is commutative.

ME5: The property of weak monotonicity is stated as  $AEM^+ (Diff_{1,2}) \geq AEM^+ (Diff_{3,4}) \Rightarrow AEM^+ (Diff_{1,2} \circ Diff_{5,6}) \geq AEM^+ (Diff_{3,4} \circ Diff_{5,6})$ . This means that  $AEM^+ (Diff_{1,2} \cup Diff_{5,6}) \geq AEM^+ (Diff_{3,4} \cup Diff_{5,6})$  given  $AEM^+ (Diff_{1,2}) \geq AEM^+ (Diff_{3,4})$ , needs to be proved. Suppose that the number of common changes between  $Diff_{3,4}$  and  $Diff_{5,6}$  are more than the ones between  $Diff_{1,2}$  and  $Diff_{5,6}$ . Since common identical changes appear only once in the concatenated table, it may well be the case that  $AEM^+ (Diff_{3,4} \cup Diff_{5,6}) \geq AEM^+ (Diff_{1,2} \cup Diff_{5,6})$ . Therefore, this axiom is not fulfilled.

ME6: To prove this axiom, the idempotent property needs to be considered. As per the definition of the concatenation operation, the metric is idempotent i.e.  $AEM^+ (Diff_{1,2} \circ Diff_{1,2}) = AEM^+ (Diff_{1,2})$ . Therefore, this axiom is not fulfilled.

It is concluded that  $AEM^+$  does not fulfill the modified extensive structure.

### **$AEM^+$ and the Independence Conditions**

IC1: It has to be shown that  $AEM^+ (Diff_{1,2} \circ Diff_{5,6}) = AEM^+ (Diff_{3,4} \circ Diff_{5,6})$  and  $AEM^+ (Diff_{5,6} \circ Diff_{1,2}) = AEM^+ (Diff_{5,6} \circ Diff_{3,4})$  given  $AEM^+ (Diff_{1,2}) = AEM^+ (Diff_{3,4})$ .  $AEM^+ (Diff_{1,2} \cup Diff_{5,6})$  may be or may not be equal to  $AEM^+ (Diff_{3,4} \cup Diff_{5,6})$  as the common changes may not be the same between  $Diff_{1,2} \cup Diff_{5,6}$  and  $Diff_{3,4} \cup Diff_{5,6}$ . The same is true between  $AEM^+ (Diff_{5,6} \cup Diff_{1,2})$  and  $AEM^+ (Diff_{5,6} \cup Diff_{3,4})$ . Hence, this condition is not fulfilled.

IC2: If the metric does not accomplish the first condition, it will not fulfill the second condition.

IC3: Due to non-accomplishment of fifth axiom of the modified extensive structure, this condition is not fulfilled.

IC4: As IC3 is not fulfilled, thus, IC4 cannot be accomplished.

It can be concluded that  $AEM^+$  does not fulfill the independence conditions.

### **$AEM^+$ and the modified relation of belief**

MR1: If the metric fulfills the weak order i.e. ME1 of modified extensive structure then this axiom is satisfied.

MR2: If the metric fulfills the weak order i.e. ME1 of modified extensive structure then, this axiom is also satisfied.

MR3: Suppose that all the changes of the table  $Diff_{3,4}$  are included in  $T_{1,2}$ , then  $AEM^+(Diff_{1,2}) \geq AEM^+(T_{3,4})$ . Thus, this axiom is satisfied.

MR4: Suppose that all the changes of the table  $Diff_{3,4}$  are included in  $Diff_{1,2}$  and  $Diff_{1,2} \cap Diff_{5,6} = \emptyset$ . Then,  $AEM^+(Diff_{3,4}) \geq AEM^+(Diff_{1,2}) \Rightarrow AEM^+(Diff_{3,4} \cup Diff_{5,6}) \geq AEM^+(Diff_{1,2} \cup Diff_{5,6})$  needs to be proved. Due to the fact that  $AEM^+(Diff_{3,4}) \geq AEM^+(Diff_{1,2})$  and that there are no common changes between  $Diff_{3,4}$  and  $Diff_{5,6}$ , the value of  $AEM^+(Diff_{3,4} \cup Diff_{5,6})$  will be more than  $AEM^+(Diff_{1,2} \cup Diff_{5,6})$ . Hence this axiom is satisfied.

MR5: This axiom is also satisfied because the changes in a process cannot be less than 0.

Therefore,  $AEM^+$  fulfills the modified relation of belief. In summary,  $AEM^+$  is a measure above the level of the ordinal scale.



We have also validated  $SEM^-$  and EM using Zuse framework and both of them are found to be above the ordinal scale. Results are shown in Table 5.7.

Table 5.7: Summary of formal validation of metrics of a WS-CDL process

<b>Metrics/Axioms</b>	<b>AEM<sup>+</sup></b>	<b>SEM<sup>-</sup></b>	<b>EM</b>
<b>ME1</b>	Y	Y	Y
<b>ME2</b>	Y	Y	Y
<b>ME3</b>	Y	Y	Y
<b>ME4</b>	Y	Y	Y
<b>ME5</b>	N	N	N
<b>ME6</b>	N	N	N
<b>IC1</b>	N	N	N
<b>IC2</b>	N	N	N
<b>IC3</b>	N	N	N
<b>IC4</b>	N	N	N
<b>MR1</b>	Y	Y	Y
<b>MR2</b>	Y	Y	Y
<b>MR3</b>	Y	Y	Y
<b>MR4</b>	Y	Y	Y
<b>MR5</b>	Y	Y	Y
<b>Scale</b>	<b>Above ordinal</b>	<b>Above ordinal</b>	<b>Above ordinal</b>

## 5.7 Summary

In this chapter, three metrics (AEM<sup>+</sup>, SEM<sup>-</sup> and EM) are proposed for measuring the evolution of a WS-CDL process. AEM<sup>+</sup> and SEM<sup>-</sup> give an idea of what kinds of changes (additive/ subtractive in nature) are made and in what quantum. EM is a total sum of both kinds of changes to give an idea of the overall evolution. The metrics are empirically validated using a case scenario. They are theoretically validated using Zuse framework and found to be above the ordinal scale.

## Chapter 6 Implementation

---

This chapter explains the details of the Metrics Computation System (MCS) which implements computation of metrics for a single as well as a composite service in SOA.

MCS is implemented on a computer system having Intel(R) Core(TM) i7-3770 CPU@3.40GHz processor, 64-bit operating system and 10 GB RAM. Eclipse 4.6.0 (Neon) is used as the Integrated Development Environment (IDE) for building the user and database interactions and to build the code for metrics computation. We have used apache-tomcat-8.5.11 for the application server. SQL Server 12.0 is used as the database server. Java code is developed using Sun Java Development Kit (jdk1.8.0\_121).

Consider, now, the data needed for computing the metrics. For a single service, different versions of WSDL document of real world services (Amazon services) are used. Simulated data is also used for a single service. Due to the non-availability of real world data, only simulated data is used for the metrics computation for a composite service (orchestration) as well as a composite service (choreography).

### 6.1 Architecture of MCS

Figure 6.1 depicts MCS architecture. We have used the following abbreviations for the modules:

Evolution Data - ED

Compute and Store - CS

Metrics Computation - MC

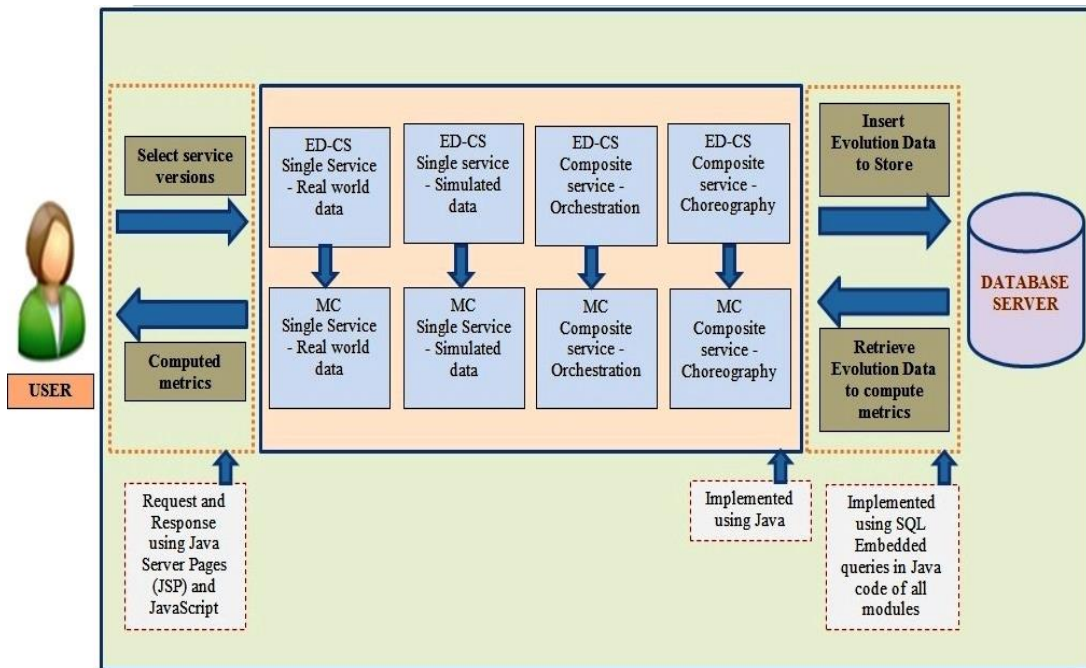


Figure 6.1: MCS Architecture

The core software modules are:

*Module 1:* ED-CS Single service - Real world data: calculates evolution data between the two versions of a service selected by the user and then creates a database table and insert the evolution data in the table.

*Module 2:* ED-CS Single service – Simulated data: generates simulated evolution data for the number of service versions selected by the user. This module then inserts this evolution data in the database tables which are created to store the data. Simulated data for all the WSDL elements of a service is generated.

*Module 3:* ED-CS Composite service - Orchestration: generates simulated evolution data for the number of composite service versions selected by the user and then creates database tables to insert the evolution data. Simulated data is generated for both basic as well as structured activities of a WS-BPEL process.

*Module 4:* ED-CS Composite service - Choreography: generates simulated evolution data for the number of composite service versions selected by the user and then

creates database tables to insert the evolution data. The WS-CDL entities are considered for which simulated evolution data is generated.

*Module 5: MC Single service - Real world data:* computes  $SEM$ ,  $SCEM_M$ ,  $SCEM_O$ ,  $SCEM_T$  and  $SUEM$  by using evolution data which is stored using the first module.

*Module 6: MC Single service – Simulated data:* computes  $SEM$ ,  $SCEM_M$ ,  $SCEM_O$ ,  $SCEM_T$  and  $SUEM$  using evolution data which is stored using the second module.

*Module 7: MC Composite service - Orchestration:* computes  $BEM_I$ ,  $BEM_E$ ,  $BUME_P$  and  $BUME_N$  using evolution data which is stored using the third module.

*Module 8: MC Composite service - Choreography:* computes  $AEM^+$ ,  $SEM^-$  and  $EM$  using evolution data which is stored using the fourth module.

Next, we show how to use MCS and its implementation for both a single service as well as a composite service.

## 6.2 MCS User Interface

User starts interacting with MCS via the user interface shown in Figure 6.2. This user interface gives two options for the user i.e. to compute metrics either for a single service or for a composite service using MCS. The first option is ‘Metrics Computation for Single Service’ and the second is ‘Metrics Computation for Composite Service’. User can select either option by clicking on the checkbox corresponding to that option. There is a ‘Submit’ button which the user clicks to further use MCS.

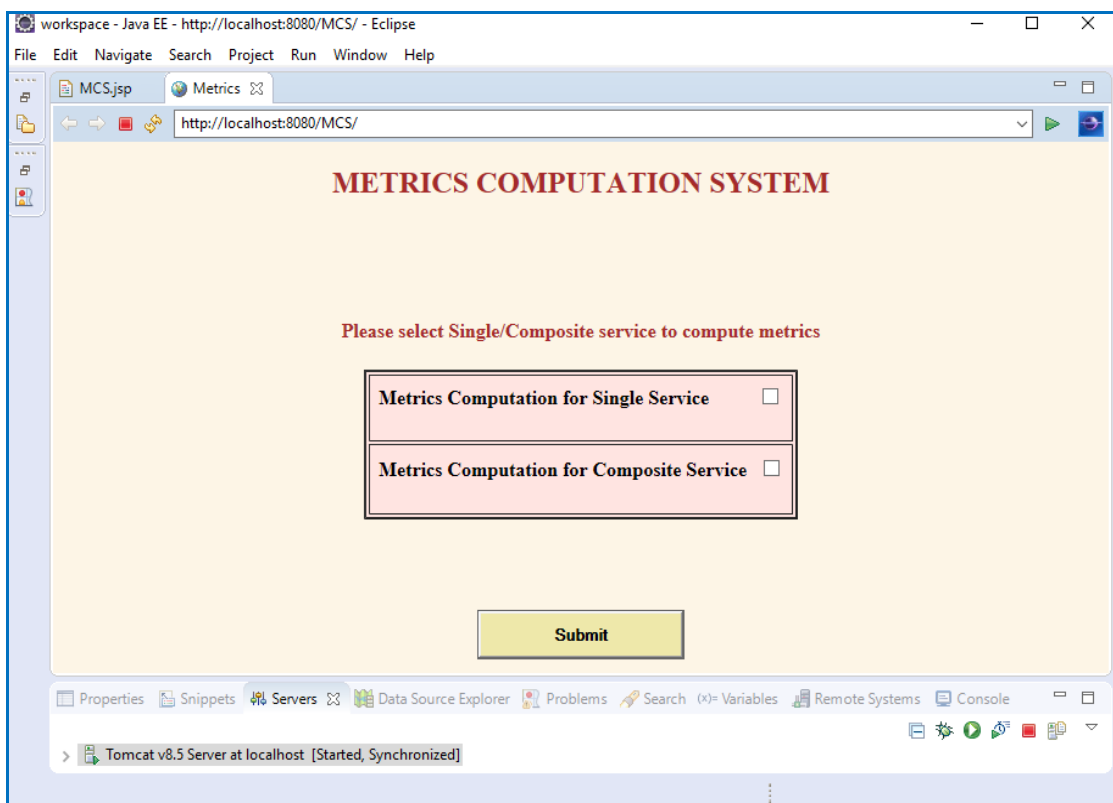


Figure 6.2: User Interface to initiate interaction with MCS

### 6.2.1 Using MCS for a Single Service

If the user chooses a single service in the previous user interface, the next user interface displayed to the user is shown in Figure 6.3. This interface gives two options to the user. The first option is to use MCS for metrics computation for single service for real world data. The second option is to use MCS for metrics computation for single service for simulated data.

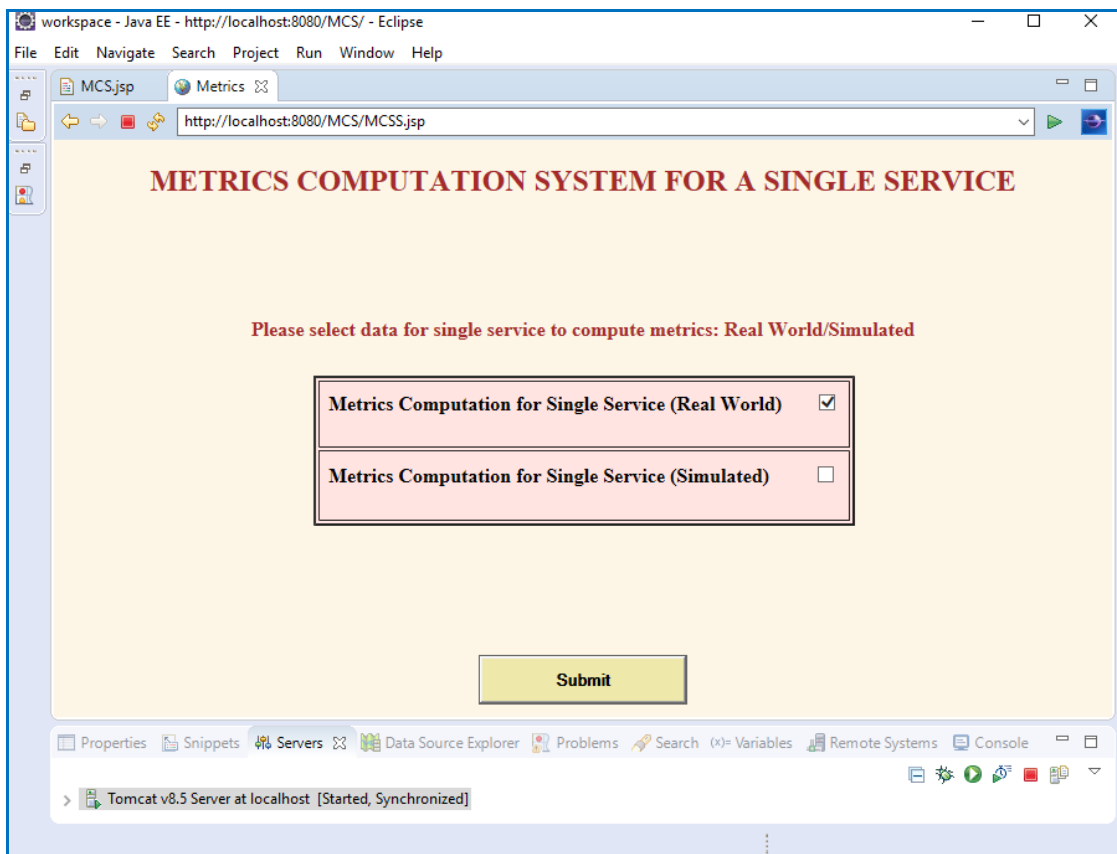


Figure 6.3: User Interface of MCS for a single service

The 'Submit' button has to be clicked after choosing one of the options.

### 6.2.1.1 Real world data

Figure 6.4 shows that this interface allows the user to choose any two service versions for which she/he wants to compute metrics. There are two rows in the table as shown in this figure. Firstly, user has to select a version of the service by clicking the browse button in the first row. Then, she/he needs to select another service version by clicking the browse button in the second row. User interface in Figure 6.4 shows that the user has selected two versions of an Airline service. User has to then click the ‘Compute’ button. MCS then computes the metrics for these selected versions of the service.

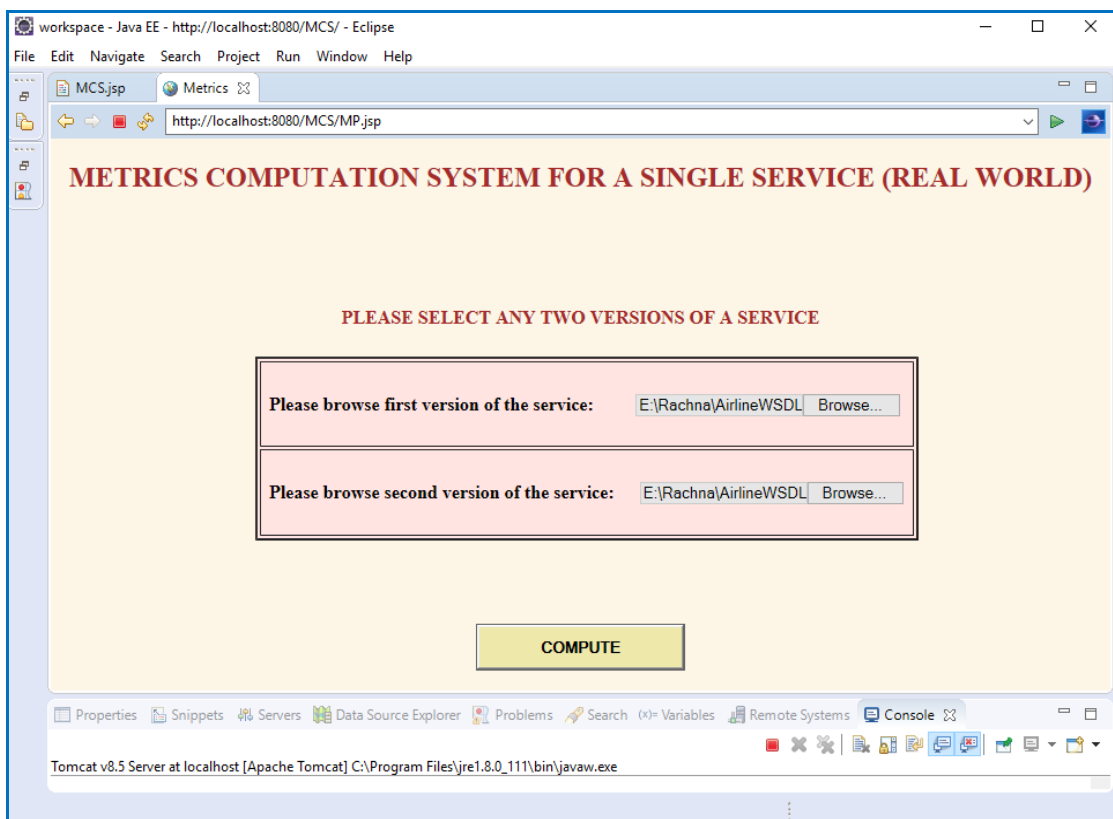


Figure 6.4: User Interface of MCS for a single service - real-world

As shown in Figure 6.5, all the computed metric values are displayed when the user clicks the ‘Display’ button. The figure displays the computed metric values for the selected versions of the Airline service by the user. Here, there are five rows corresponding to SEM, SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> and SUEM. Each row has a textbox in which the corresponding metric value is displayed.

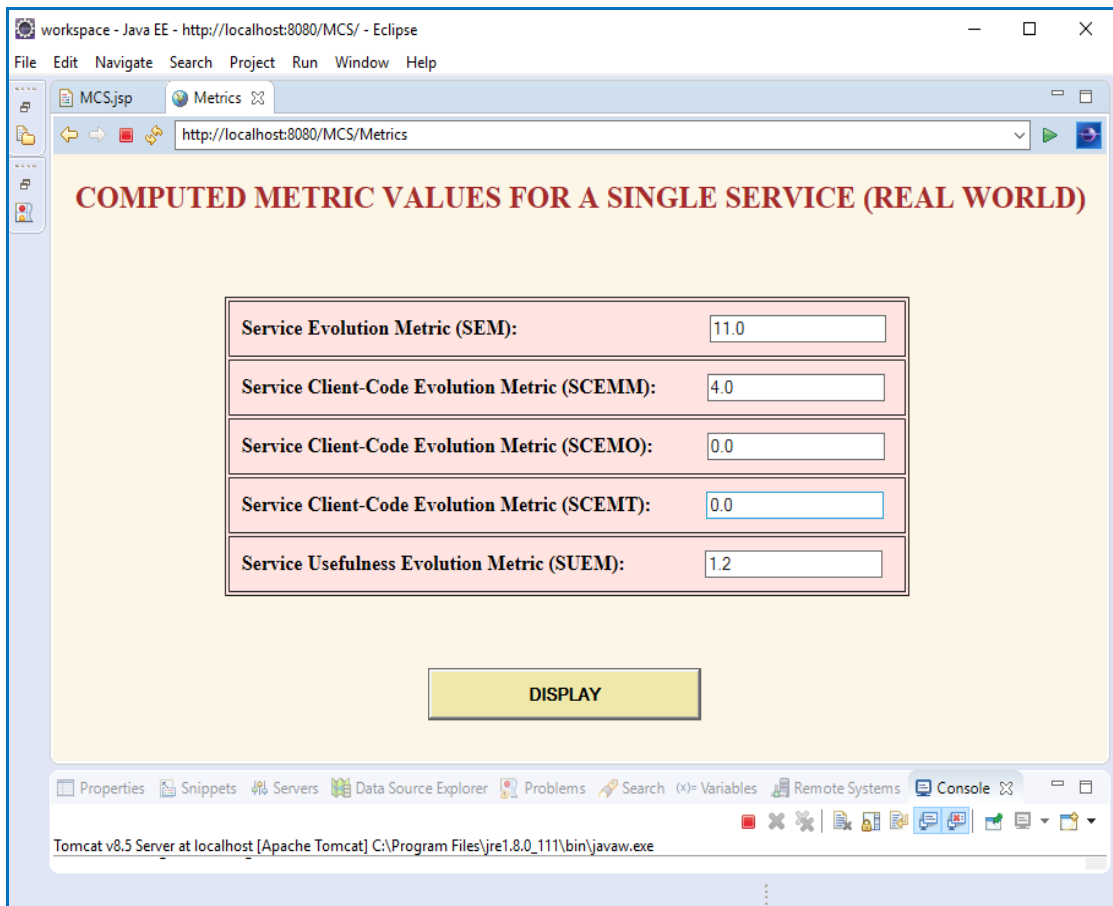


Figure 6.5: Computed metrics for a single service – real world



### 6.2.1.2 Simulated data

The user interface shown in Figure 6.6 is displayed when she/he clicks on the checkbox in the second row of the table and then on the ‘Submit’ button in Figure 6.4. Figure 6.6 is the user interface of MCS for a single service for simulated data. Here, there is a drop-down box which is used to select for how many service versions the user wants to compute the metrics. There is a ‘Compute’ button to navigate to the next user interface. In Figure 6.6, user selects 20 service versions for metrics computation.

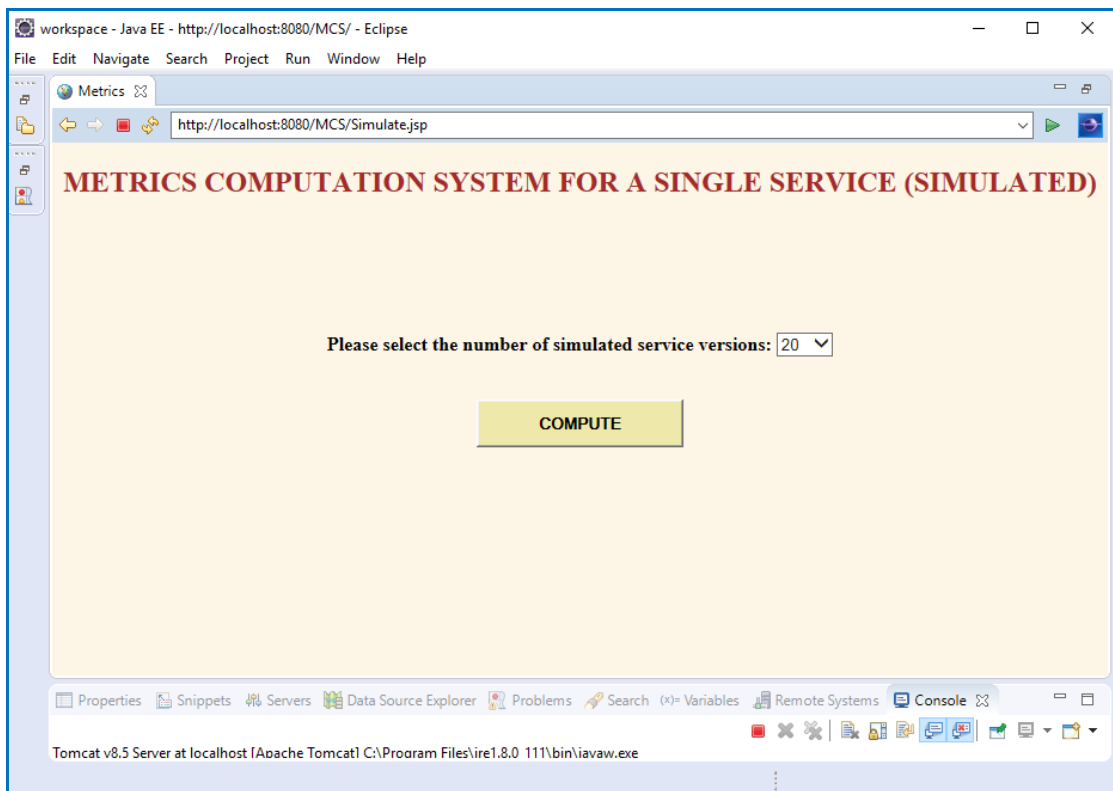


Figure 6.6: User Interface of MCS for a single service – simulated

When the user clicks the ‘Compute’ button, all the metrics that are computed for the simulated data for the selected number of service versions are displayed. Figure 6.7 shows the computed metric values for the user who has selected 20 service versions.

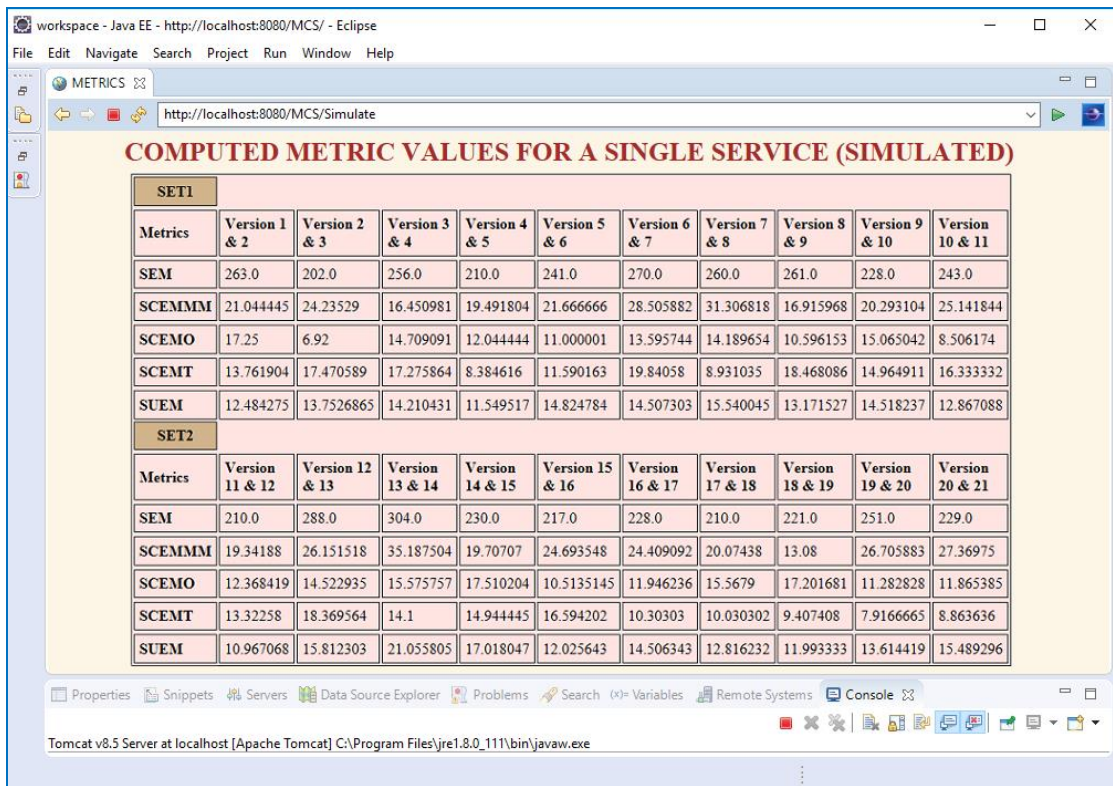


Figure 6.7: Computed metrics for a single service – simulated

## 6.2.2 Using MCS for a Composite Service

If the user chooses a composite service in Figure 6.2, the user interface shown in Figure 6.8 is displayed. This interface gives two options to the user. The first option is to use MCS for metrics computation for a composite service (orchestration). The second option is to use MCS for metrics computation for a composite service (choreography).

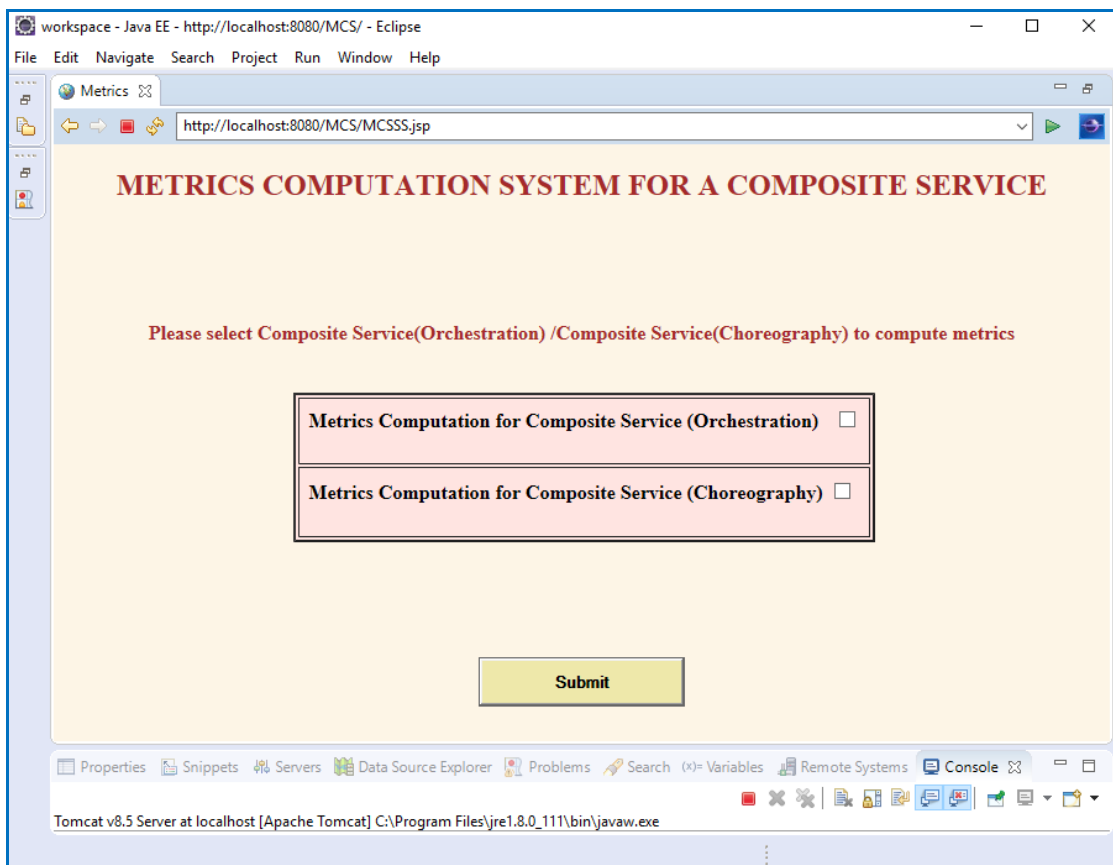


Figure 6.8: User Interface of MCS for a composite service

The 'Submit' button has to be clicked after choosing one of the options.

### 6.2.2.1 Orchestration

When the user selects the first option of Figure 6.8, the user interface as shown in Figure 6.9 is displayed. User has to select the number of versions. MCS computes metrics for simulated data of composite service (orchestration). In Figure 6.9, user has selected 20 versions.

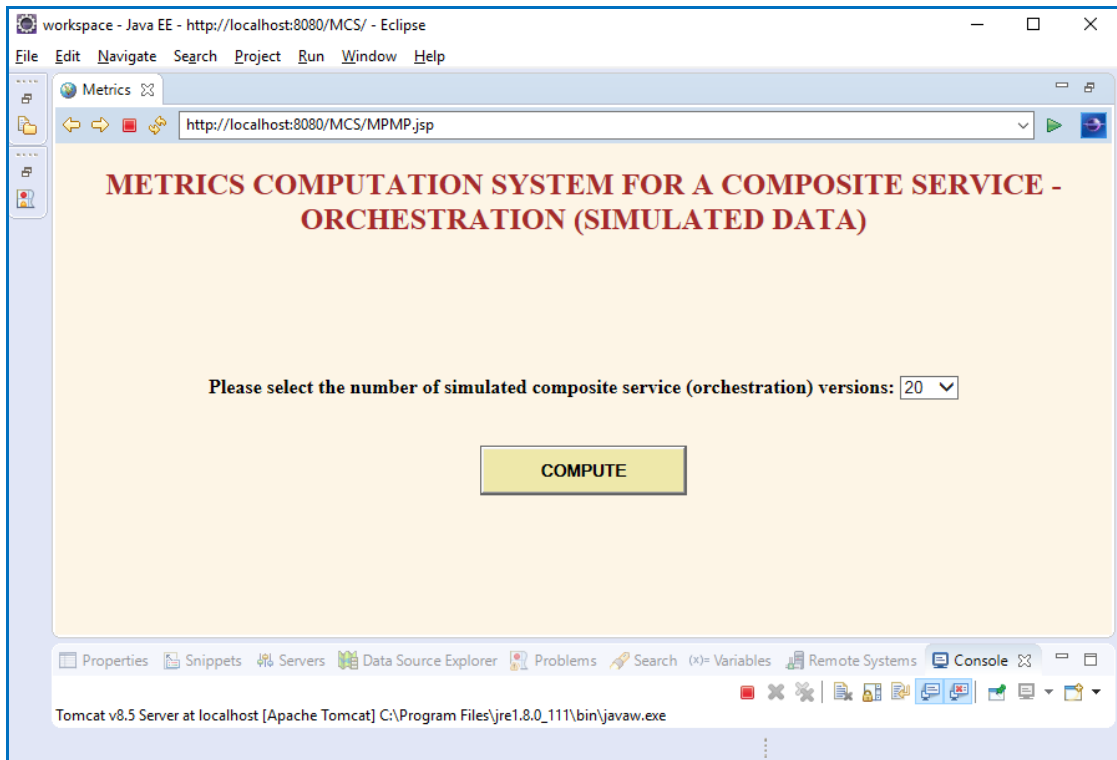


Figure 6.9: User Interface for a composite service – Orchestration

After selecting the number of versions in Figure 6.9, user clicks the ‘Compute’ button. The computed metrics for the selected number of versions are displayed. Figure 6.10 shows computed metrics for 20 selected versions of the composite service.

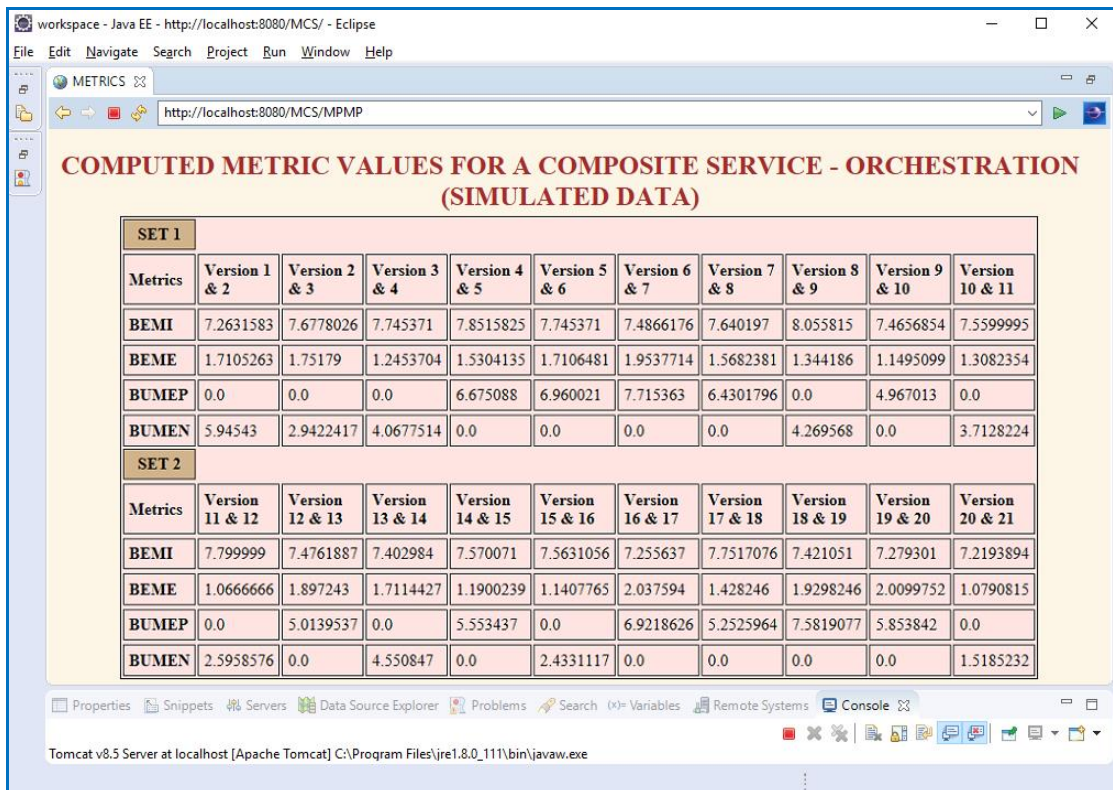


Figure 6.10: Computed metrics of a composite service – Orchestration

### 6.2.2.2 Choreography

To use MCS for composite service (choreography), user has to select the checkbox in the second row in Figure 6.8. The user interface as shown in Figure 6.11 is then displayed. The number of versions to compute the metrics needs to be selected. Then, MCS computes metrics for composite service (choreography). User has selected 20 versions as shown in Figure 6.11.

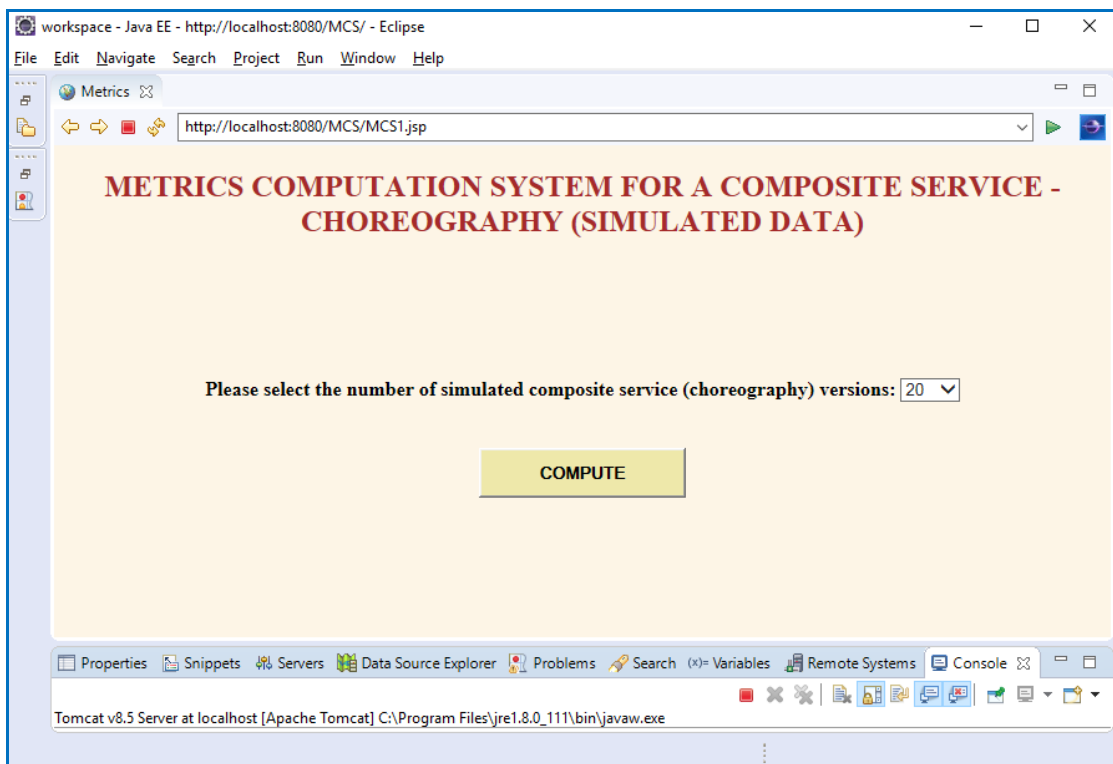


Figure 6.11: User Interface of MCS for a composite service – Choreography

User clicks the ‘Compute’ button in Figure 6.11 after selecting the number of versions. The computed metrics are displayed. Figure 6.12 shows computed metrics for 20 selected versions of the composite service (choreography).

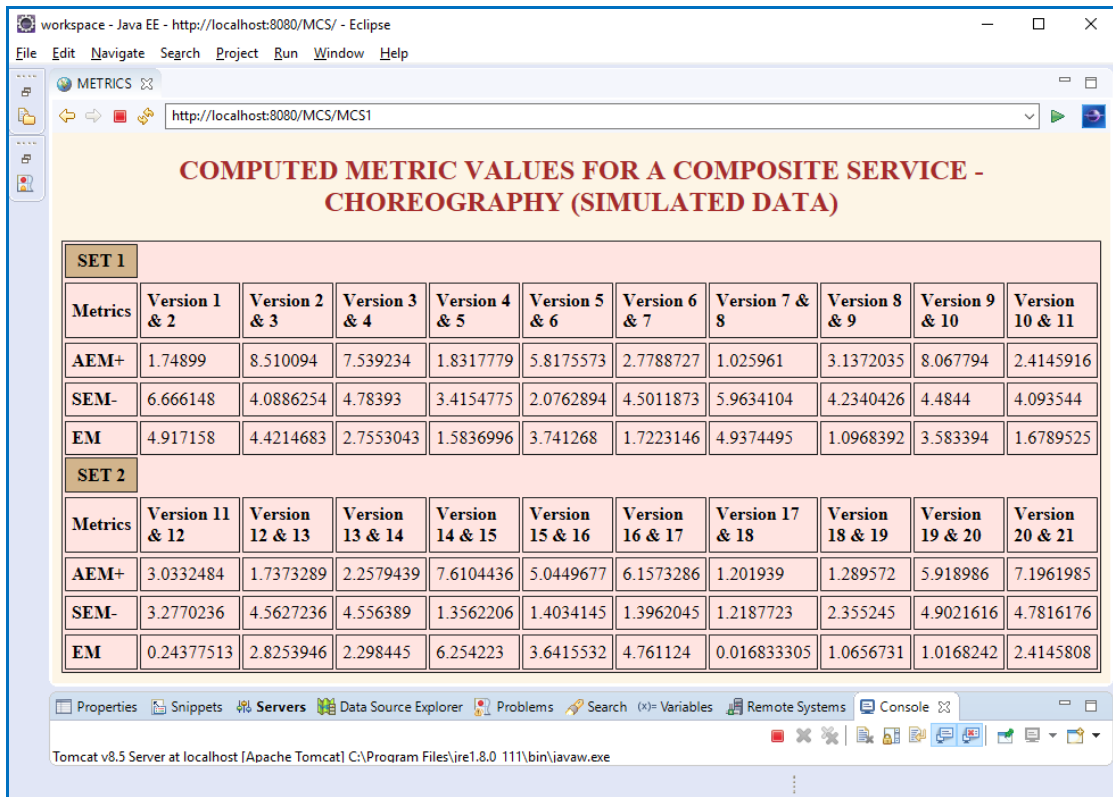


Figure 6.12: Computed metrics for a composite service – Choreography

Now, we have seen the different user interfaces provided by MCS to know how it is used by the user. Next, we show how the core modules mentioned in section 6.1 are implemented to know how MCS internally works.

## 6.3 Implementation of MCS

In this section, we discuss the implementation of MCS for a single service as well as composite service.

### 6.3.1 Single Service - Real World Data

ED-CS Single service - Real world data and MC Single service - Real world data are the two modules which realize the computation of the metrics.

ED-CS Single service - Real world data is implemented as given in Algorithm 1. In this algorithm, firstly database table is created to store the evolution data between the selected two versions of a single real world service. Then, the selected versions are parsed to compute the evolution data for each WSDL element of the service versions. After this, the evolution data to compute the metrics for the selected versions is stored in the table created before.

**Algorithm 1 : ED-CS Single service - Real world data**

**Input:** WSDL File versions

**Output:** Evolution data stored in database for Input

1. Create table with columns: Service versions, Element, Depth of the element, Number of changes and Change category
2. Parse selected versions of the WSDL files of service.
3. Compute each change (evolution data) between the selected versions i.e. addition, deletion, modification, merge and split in each element.
4. Store result in table.

MC Single service - Real world data is implemented as given in Algorithm 2. Data stored in tables (using Algorithm 1) is read. The data is used to compute the metrics for a single service i.e. SEM, SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> and SUEM.



**Algorithm 2: MC Single service - Real world data****Input:** WSDL File versions**Output:** Computed values for SEM, SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> and SUEM

1. Read tables to compute SEM using SQL statements which uses columns: Service versions, Depth of the element and Number of changes
2. Use data from step 1 to compute SEM
3. Read tables to compute SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> using SQL statements which uses columns: Service versions, Element, Number of changes and Change category
4. Use data from step 3 to compute SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub>
5. Read tables to compute SUEM using SQL statements which uses columns: Service versions, Element, Number of changes and Change category
6. Use data from step 5 to compute SUEM

**6.3.2 Single Service – Simulated Data**

The computation of the metrics is realized through ED-CS Single service - Simulated data and MC Single service – Simulated data.

ED-CS Single service - Simulated data is implemented using Algorithm 3. At first, database table is created to store the evolution data for the selected number of versions for a single service (simulated). If a user selects 20 number of versions, then the evolution data is generated randomly for the subsequent versions i.e. between Version 1&2, Version 2&3,....., Version 20&21. Evolution data for each WSDL element is computed. After this, the evolution data to compute the metrics for the selected number of versions is stored in the table.

**Algorithm 3: ED-CS Single service - Simulated data****Input:** Number of service versions for the simulated data**Output:** Evolution data stored in database for selected Input

1. Create tables with columns : Service versions, Element, Depth of the element, Number of Changes and Change category
2. Generate randomly changes for each WSDL element for the number of versions given in Input
3. Store data in table

Algorithm 4 is used to implement MC Single service – Simulated data. Data stored in tables (using Algorithm 3) is read. The data is used to compute the metrics for a single service i.e. SEM, SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> and SUEM.

**Algorithm 4: MC Single service – Simulated data**

**Input:** Number of selected service versions to compute metrics

**Output:** Computed values for SEM, SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> and SUEM

1. Read tables to compute SEM using SQL statements which uses columns: Service versions, Depth of the element, Number of changes and Change category
2. Use data from step 1 to compute SEM
3. Read tables to compute SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub> using SQL statements which uses columns: Service versions, Element, Number of changes and Change category
4. Use data from step 3 to compute SCEM<sub>M</sub>, SCEM<sub>O</sub>, SCEM<sub>T</sub>
5. Read tables to compute SUEM using SQL statements which uses columns: Service versions, Element, Number of changes and Change category
6. Use data from step 5 to compute SUEM

### 6.3.3 Composite Service - Orchestration

For the computation of metrics proposed for a composite service (orchestration), two modules i.e. ED-CS Composite service - Orchestration and MC Composite service - Orchestration are used.

The ED-CS Composite service - Orchestration module is implemented using Algorithm 5. The first step in the algorithm is to create the database table to store the evolution data for the selected number of versions for a composite service (orchestration-simulated). The evolution data is generated randomly for the selected number of versions for each WS-BPEL activity (basic and structured). After this, the evolution data is stored in the table.

**Algorithm 5: ED-CS Composite service - Orchestration**

**Input:** Number of composite service versions for the simulated data

**Output:** Evolution data stored in database for selected Input

1. Create tables with columns : Process versions, Activity, Number of activities in previous version, Number of changes and Change category
2. Generate randomly, changes for each WS-BPEL activity for the number of versions given in Input.
3. Store data in tables.

The MC Composite service - Orchestration module is implemented using Algorithm 6. Data stored in tables (using Algorithm 5) is read. The data is used to compute the metrics for a single service i.e.  $BEM_E$ ,  $BEM_I$ ,  $BUME_P$  and  $BUME_N$ .

**Algorithm 6: MC Composite service - Orchestration**

**Input:** Number of selected composite service versions to compute metrics

**Output:** Computed values for  $BEM_E$ ,  $BEM_I$ ,  $BUME_P$  and  $BUME_N$

1. Read tables to compute  $BEM_E$  using SQL statements which uses columns: Process versions, Activity, Number of changes and Change category
2. Use data from step 1 to compute  $BEM_E$
3. Read tables to compute  $BEM_I$  using SQL statements which uses columns: Process versions, Activity, Number of changes and Change category
4. Use data from step 3 to compute  $BEM_I$
5. Read tables to compute  $BUME_P$  using SQL statements which uses columns: Process versions, Activity, Number of activities in previous version, Number of changes and Change category
6. Use data from step 5 to compute  $BUME_P$
7. Read tables to compute  $BUME_N$  using SQL statements which uses columns: Process versions, Activity, Number of activities in previous version, Number of changes and Change category
8. Use data from step 7 to compute  $BUME_N$

### 6.3.4 Composite Service - Choreography

MCS for composite service (choreography) is implemented using ED-CS Composite service - Choreography and MC Composite service - Choreography modules.

ED-CS Composite service - Choreography is implemented using Algorithm 7. Firstly, database table is created to store the evolution data for the selected number of versions for a composite service (choreography-simulated). The evolution data is generated randomly for the selected number of versions for WS-CDL entities (participant/role/interaction). After this, the evolution data is stored in the created table.

**Algorithm 7: ED-CS Composite service - Choreography****Input:** Number of composite service versions for the simulated data**Output:** Evolution data stored in database for selected Input

1. Create tables with columns: CDL process versions, Entity, Peer/Participant, Number of peer/participant, Change, Number of changes
2. Generate randomly, changes for WS-CDL entities (participant, role, and interaction) for the number of versions given in Input.
3. Store data in tables.

MC Composite service - Choreography module is implemented using Algorithm 8. Data stored in tables (using Algorithm 7) is read. The data is used to compute the metrics for a single service i.e.  $AEM^+$ ,  $SEM^-$  and EM.

**Algorithm 8: MC Composite service - Choreography****Input:** Number of composite selected service versions to compute metrics**Output:** Computed values for  $AEM^+$ ,  $SEM^-$  and EM

1. Read tables to compute  $AEM^+$  using SQL statements which uses columns: CDL process versions, Entity, Peer/Participant, Number of peer/participant, Change, Number of changes
2. Use data from step 1 to compute  $AEM^+$
3. Read tables to compute  $SEM^-$  using SQL statements which uses columns: CDL process versions, Entity, Peer/Participant, Number of peer/participant, Change, Number of changes
4. Use data from step 3 to compute  $SEM^-$
5. Read tables to compute EM using values of  $AEM^+$  and  $SEM^-$
6. Use data from step 5 to compute EM

All algorithms are implemented using Java, Embedded SQL queries in Java code and wsdl4j-1\_6\_3 parser (used as a plug-in in Eclipse).

## 6.4 Summary

In this chapter, we have presented Metrics Computation System (MCS) which is developed to compute the metrics proposed in this thesis. MCS contains modules to compute and store changes and also to compute the metrics for different versions of a single as well as a composite service.

## Chapter 7 Conclusion

---

In this thesis, we have proposed metrics for the evolving services in SOA for both single and composite service. The service provider's as well as the service consumer's perspective has been considered while proposing metrics. The metrics provide a measure of the quantum of change in a service to the provider. They also measure the impact on the consumer. This impact has been studied from different perspectives i.e. impact on the service client code and impact on the usefulness of the service.

For a single service, to provide a measure of the overall evolution for the service provider, we have proposed SEM metric. In order to measure the impact of service evolution in the service client code,  $SCEM_M$ ,  $SCEM_O$  and  $SCEM_T$  are proposed. These metrics are a measure of the amount of the changes (mandatory, optional, trivial) for the client code to adapt. The proposed metric SUEM measures the impact on the usefulness for the consumer as a service evolves. The correlation analysis of the metrics helps the service provider to identify phases of the service evolution which are beneficial to the consumer and which are not.

For a composite service, metrics for both orchestration and choreography are considered. WS-BPEL document structure is used while defining the metrics for service composition through orchestration. Metrics are proposed both for the provider as well as the consumer. Two WS-BPEL Evolution Metrics are proposed for the provider. One is for external evolution ( $BEM_E$ ) and the other is for internal evolution ( $BEM_I$ ). The changes which involve interactions with the external partner services are measured by  $BEM_E$ . The changes which are confined only to the internal logic of the process are measured by  $BEM_I$ . For the consumer, two metrics are proposed i.e.  $BUME_P$  to measure the impact of evolution on process usefulness in a positive sense and  $BUME_N$  as a measure in a negative sense. It is used to give an idea by how much the process is useful for her/him across its different versions. Here, we have defined

metric for favorable/ unfavorable/indifferent changes. The metric analysis shows the degree of variance of usefulness for all these different kinds of changes.

Using versions of WS-CDL document, the evolution in choreography has also been studied. Three metrics are proposed considering from the provider's perspective. The first metric is AEM<sup>+</sup> which is a measure of all the changes which are additive in nature. The second metric, SEM<sup>-</sup>, measures those changes which are subtractive in nature. Finally, EM, measures the total sum of both the kinds of changes in the choreography to give an idea of the overall evolution.

We have performed theoretical validation of all the proposed metrics using Zuse framework. All the metrics are found to be above the ordinal scale level. The metrics were validated empirically using real time data for a single service. Simulated data has been used for a composite service as well as a single service because of the non-availability or insufficiency of real time data.

Metrics Computation System (MCS) has been developed to implement the computation of the metrics. MCS was implemented using jdk1.8.0\_121, SQL Server 12.0, Apache Tomcat 8.5.11, and Eclipse 4.6.0 (Neon).

## **7.1 Future Work**

The future directions related to the proposed work in the thesis are as follows.

a) Metrics can be proposed for composite services (orchestration) using languages other than WS-BPEL such as Web Services Flow Language (WSFL), Xlang etc. For composite services (choreography), metrics can be proposed using languages such as Web Service Conversation Language (WSCL), Web Service Choreography Interface (WSCI) etc.

b) The metrics of different languages can, then, be compared. Further, the metrics can be studied to determine whether any of the languages is better from the point of evolution.

## References

- [1] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.
- [2] T. Erl, *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice hall, 2004.
- [3] L.-J. Zhang, "SOA and Web services," in *Services Computing, 2006. SCC'06. IEEE International Conference on*, 2006, pp. xxxvi--xxxvi.
- [4] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 1, no. 1, pp. 101–105, 2009.
- [5] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005.
- [6] E. Newcomer and G. Lomow, *Understanding SOA with Web services*. Addison-Wesley, 2005.
- [7] T. Unger, F. Leymann, S. Mauchart, and T. Scheibler, "Aggregation of service level agreements in the context of business processes," in *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE*, 2008, pp. 43–52.
- [8] A. Barker and J. Van Hemert, "Scientific workflow: a survey and research directions," *Parallel Process. Appl. Math.*, pp. 746–753, 2008.
- [9] C. Barreto *et al.*, "Web Services Business Process Execution Language Version 2.0," *OASIS*, 2007. [Online]. Available: <https://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm>. [Accessed: 01-Jan-2013].
- [10] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based analysis of obligations in web service choreography," in *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, 2006, p. 149.
- [11] S. Ross and T. Fletcher, "Web Services Choreography Description Language: Primer," *W3C Working Draft*, 2006. [Online]. Available: <https://www.w3.org/TR/ws-cdl-10-primer/>. [Accessed: 01-Jan-2014].

- [12] M. Rani, A. K. Chawla, and S. Batra, “Web service choreography description language (WS-CDL): Goals and benefits,” *COIT*. [Online]. Available <http://www.rimtenqq.com/coit2007/proceedings/pdfs/49.pdf>, 2006.
- [13] E. A. Marks and M. Bell, *Service Oriented Architecture (SOA): a planning and implementation guide for business and technology*. John Wiley & Sons, 2008.
- [14] J. P. Lawler and H. Howell-Barber, *Service-oriented architecture: SOA strategy, methodology, and technology*. CRC Press, 2007.
- [15] T. Erl, *Soa: principles of service design*. Prentice Hall Press, 2007.
- [16] S. Carter, *The New Language of Business: SOA & Web 2.0 (Adobe Reader)*. Pearson Education, 2007.
- [17] M. N. Huhns and M. P. Singh, “Service-oriented computing: Key concepts and principles,” *IEEE Internet Comput.*, vol. 9, no. 1, pp. 75–81, 2005.
- [18] D. Booth *et al.*, “Web Services Architecture,” *W3C Working Group*, 2004. [Online]. Available: <https://www.w3.org/TR/ws-arch/>. [Accessed: 01-Jan-2013].
- [19] D. Booth and C. Kevin Liu, “Web Services Description Language (WSDL) Version 2.0 Part 0: Primer,” *W3C*, 2007. [Online]. Available: <https://www.w3.org/TR/wsd120-primer/>. [Accessed: 01-Jan-2013].
- [20] G. Mein *et al.*, “Simple object access protocol.” Google Patents, 2002.
- [21] M. Gudgin *et al.*, “SOAP Version 1.2 Part 1: Messaging Framework,” *W3C*, 2007. [Online]. Available: <https://www.w3.org/TR/soap12/>. [Accessed: 01-Jan-2013].
- [22] N. Milanovic and M. Malek, “Current solutions for web service composition,” *IEEE Internet Comput.*, vol. 8, no. 6, pp. 51–59, 2004.
- [23] A. Barker, C. D. Walton, and D. Robertson, “Choreographing web services,” *IEEE Trans. Serv. Comput.*, vol. 2, no. 2, pp. 152–166, 2009.
- [24] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, “Service-oriented computing: a research roadmap,” *Int. J. Coop. Inf. Syst.*, vol. 17, no. 2, pp. 223–255, 2008.
- [25] M. P. Papazoglou, V. Andrikopoulos, and S. Benbernou, “Managing evolving services,” *IEEE Softw.*, vol. 28, no. 3, pp. 49–55, 2011.



- [26] M. B. Juric, A. Sasa, B. Brumen, and I. Rozman, “WSDL and UDDI extensions for version support in web services,” *J. Syst. Softw.*, vol. 82, no. 8, pp. 1326–1343, 2009.
- [27] S. Akram, A. Bouguettaya, X. Liu, A. Haller, and F. Rosenberg, “A Change Management Framework for Service Oriented Enterprises.,” *Int. J. Next-Generation Comput.*, vol. 1, no. 1, 2010.
- [28] Y. Wang, J. Yang, W. Zhao, and J. Su, “Change impact analysis in service-based business processes,” *Serv. Oriented Comput. Appl.*, vol. 6, no. 2, pp. 131–149, 2012.
- [29] D. Kim, M. Kim, and H. Kim, “Dynamic business process management based on process change patterns,” in *Convergence information technology, 2007. international conference on*, 2007, pp. 1154–1161.
- [30] M. Koning, C. Sun, M. Sinnema, and P. Avgeriou, “VxBPEL: Supporting variability for Web services in BPEL,” *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 258–269, 2009.
- [31] W. Fdhila, A. Baouab, K. Dahman, C. Godart, O. Perrin, and F. Charoy, “Change propagation in decentralized composite web services,” in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, 2011, pp. 508–511.
- [32] H. Liu, Z. Li, J. Zhu, and H. Tan, “Business process regression testing,” in *ICSOC, 2007*, vol. 7, pp. 157–168.
- [33] A. Slominski, “Adapting BPEL to scientific workflows,” *Work. e-Science*, pp. 208–226, 2007.
- [34] M. Hiel, H. Aldewereld, and F. Dignum, “Ensuring conformance in an evolving choreography,” in *Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on*, 2010, pp. 1–4.
- [35] W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert, “Dealing with change in process choreographies: Design and implementation of propagation algorithms,” *Inf. Syst.*, vol. 49, pp. 1–24, 2015.
- [36] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, “Requirements-driven collaborative choreography customization,” *Serv. Comput.*, pp. 144–158, 2009.

- [37] S. Rinderle, A. Wombacher, and M. Reichert, “Evolution of process choreographies in DYCHOR,” *Move to Meaningful Internet Syst. 2006 CoopIS, DOA, GADA, ODBASE*, pp. 273–290, 2006.
- [38] J. Su, T. Bultan, X. Fu, and X. Zhao, “Towards a theory of web service choreographies,” *Lect. Notes Comput. Sci.*, vol. 4937, pp. 1–16, 2008.
- [39] B. Curtis, “Measurement and experimentation in software engineering,” *Proc. IEEE*, vol. 68, no. 9, pp. 1144–1157, 1980.
- [40] B. A. Kitchenham, *Software metrics: measurement for software process improvement*. Blackwell Publishers, Inc., 1996.
- [41] J. E. Gaffney Jr, “Metrics in software quality assurance,” in *Proceedings of the ACM’81 conference*, 1981, pp. 126–130.
- [42] N. Fenton and J. Bieman, *Software metrics: a rigorous and practical approach*. CRC Press, 2014.
- [43] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski, “Metrics and laws of software evolution-the nineties view,” in *Software Metrics Symposium, 1997. Proceedings., Fourth International*, 1997, pp. 20–32.
- [44] N. Drouin, M. Badri, and F. Touré, “Metrics and software quality evolution: A case study on open source software,” *Int. J. Comput. Theory Eng.*, vol. 5, no. 3, p. 523, 2013.
- [45] C. Gerlec and M. Hericko, “Analyzing Structural Software Changes: A Case Study,” in *BCI (Local)*, 2012, pp. 117–120.
- [46] T. Mens and S. Demeyer, “Future trends in software evolution metrics,” in *Proceedings of the 4th international workshop on Principles of software evolution*, 2001, pp. 83–86.
- [47] A. Mockus and L. G. Votta, “Identifying Reasons for Software Changes using Historic Databases,” in *icsm*, 2000, pp. 120–130.
- [48] Y. Lee, J. Yang, and K. H. Chang, “Metrics and evolution in open source software,” in *Quality Software, 2007. QSIC’07. Seventh International Conference on*, 2007, pp. 191–197.
- [49] J. Van Gurp and J. Bosch, “Design erosion: problems and causes,” *J. Syst.*

*Softw.*, vol. 61, no. 2, pp. 105–119, 2002.

- [50] Z. Balfagih and M. F. Hassan, “Quality model for web services from multi-stakeholders’ perspective,” in *Information Management and Engineering, 2009. ICIME’09. International Conference on*, 2009, pp. 287–291.
- [51] S. W. Choi, J. S. Her, and S. D. Kim, “Modeling QoS attributes and metrics for evaluating services in SOA considering consumers’ perspective as the first class requirement,” in *Asia-Pacific Service Computing Conference, The 2nd IEEE*, 2007, pp. 398–405.
- [52] D. Rud, A. Schmietendorf, and R. Dumke, “Resource metrics for service-oriented infrastructures,” *Proc. SEMSOA 2007*, pp. 90–98, 2007.
- [53] Y. Lee, “QoS metrics for service level measurement for SOA environment,” in *Advanced Information Management and Service (IMS), 2010 6th International Conference on*, 2010, pp. 509–514.
- [54] M. Pereplechikov, C. Ryan, and K. Frampton, “Cohesion metrics for predicting maintainability of service-oriented software,” in *Quality Software, 2007. QSIC’07. Seventh International Conference on*, 2007, pp. 328–335.
- [55] M. Gebhart, “Measuring design quality of service-oriented architectures based on web services,” in *Eighth International Conference on Software Engineering Advances (ICSEA 2013), Venice, Italy*, 2013, pp. 504–509.
- [56] D. Athanopoulos and A. V Zarras, “Fine-grained metrics of cohesion lack for service interfaces,” in *Web Services (ICWS), 2011 IEEE International Conference on*, 2011, pp. 588–595.
- [57] M. Pereplechikov, C. Ryan, K. Frampton, and Z. Tari, “Coupling metrics for predicting maintainability in service-oriented designs,” in *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, 2007, pp. 329–340.
- [58] S. Kalepu, S. Krishnaswamy, and S. W. Loke, “Verity: a QoS metric for selecting Web services and providers,” in *Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on*, 2003, pp. 131–139.
- [59] S. W. Choi and S. D. Kim, “A quality model for evaluating reusability of

- services in soa,” in *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, 2008, pp. 293–298.
- [60] D. Dyachuk and R. Deters, “Using sla context to ensure quality of service for composite services,” in *Pervasive Services, IEEE International Conference on*, 2007, pp. 64–67.
- [61] A. Khoshkbarforoushha, R. Tabein, P. Jamshidi, and F. Shams, “Towards a metrics suite for measuring composite service granularity level appropriateness,” in *Services (SERVICES-1), 2010 6th World Congress on*, 2010, pp. 245–252.
- [62] B. Wetzstein, S. Strauch, and F. Leymann, “Measuring performance metrics of WS-BPEL service compositions,” in *Networking and Services, 2009. ICNS’09. Fifth International Conference on*, 2009, pp. 49–56.
- [63] K. Qian, J. Liu, and F. Tsui, “Decoupling metrics for services composition,” in *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COM SAR 2006. 5th IEEE/ACIS International Conference on*, 2006, pp. 44–47.
- [64] P. T. Quynh and H. Q. Thang, “Dynamic coupling metrics for service-oriented software,” *World Acad. Sci. Eng. Technol. Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 3, no. 3, pp. 795–800, 2009.
- [65] X. Wang, “Metrics for evaluating coupling and service granularity in service oriented architecture,” in *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, 2009, pp. 1–4.
- [66] B. Wetzstein, D. Karastoyanova, and F. Leymann, “Towards management of SLA-aware business processes based on key performance indicators,” in *9th Workshop on Business Process Modeling, Development and Support (BPMDS’08)-Business Process Life-Cycle: Design, Deployment, Operation & Evaluation*, 2008.
- [67] H. Zuse, *A framework of software measurement*. Walter de Gruyter, 1998.
- [68] I. Mistr’\ik, R. M. Soley, N. Ali, J. Grundy, and B. Tekinerdogan, *Software*

*quality assurance: in large scale and complex software-intensive systems.*

Morgan Kaufmann, 2015.

- [69] L. Finkelstein, "Theory and philosophy of measurement," *Handb. Meas. Sci.*, vol. 1, pp. 1–30, 1982.
- [70] L. Finkelstein and M. S. Leaning, "A review of the fundamental concepts of measurement," *Measurement*, vol. 2, no. 1, pp. 25–34, 1984.
- [71] S. L. Pfleeger and J. M. Atlee, *Software engineering: theory and practice*. Pearson Education India, 1998.
- [72] A. L. Baker, J. M. Bieman, N. Fenton, D. A. Gustafson, A. Melton, and R. Whitty, "A philosophy for software measurement," *J. Syst. Softw.*, vol. 12, no. 3, pp. 277–281, 1990.
- [73] M. E. Bush and N. E. Fenton, "Software measurement: a conceptual framework," *J. Syst. Softw.*, vol. 12, no. 3, pp. 223–231, 1990.
- [74] V. R. Basili, "Software modeling and measurement: the Goal/Question/Metric paradigm," 1992.
- [75] N. Simpkins, "T320 E-business technologies: foundations and practice," 2008. [Online]. Available: [http://www.eclipse.org/webtools/community/education/web/t320/Generating\\_a\\_client\\_from\\_WSDL.pdf](http://www.eclipse.org/webtools/community/education/web/t320/Generating_a_client_from_WSDL.pdf).
- [76] Oracle, "Creating a Simple Web Service and Client with JAX-WS - The Java EE 5 Tutorial," 2010. [Online]. Available: <http://docs.oracle.com/javase/5/tutorial/doc/bnayn.html>.
- [77] Pivotal, "Consuming a SOAP web service," 2016. [Online]. Available: <https://spring.io/%0Aguides/gs/consuming-web-service/>.
- [78] Z. M, "Create a web service client for a SOAP based web service," 2012. [Online]. Available: <http://java.boot.by/ocewsd6-guide/ch06.html>.
- [79] C. Peltz, "Web services orchestration and choreography," *Computer (Long Beach. Calif.)*, vol. 36, no. 10, pp. 46–52, 2003.
- [80] F. Daniel and B. Pernici, "Insights into web service orchestration and choreography," *Int. J. E-bus. Res.*, vol. 2, no. 1, pp. 58–77, 2006.