A STATIC ANALYSIS APPROACH FOR SERVICE ORIENTED SOFTWARE ENGINEERING (SOSE) DESIGNS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CAN ÇERMİKLİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

A STATIC ANALYSIS APPROACH FOR SERVICE ORIENTED SOFTWARE ENGINEERING (SOSE) DESIGNS

submitted by **CAN ÇERMİKLİ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Ali Hikmet Doğru _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Assoc. Prof. Ahmet Coşar
Computer Engineering Department, METU _____

Assoc. Prof. Ali Hikmet Doğru
Computer Engineering Department, METU _____

Dr. Cevat Şener
Computer Engineering Department, METU _____

Dr. Kıvanç Dinçer
TÜBİTAK - UEKAE _____

Dr. Ahmet Tümay
TÜBİTAK - UEKAE _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: CAN ÇERMİKLİ

Signature :

ABSTRACT

A STATIC ANALYSIS APPROACH FOR SERVICE ORIENTED
SOFTWARE ENGINEERING (SOSE) DESIGNS

Çermikli, Can
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Ali Hikmet Doğru

September 2010, 61 pages

In this thesis, a static analysis approach is introduced in order to develop correct business processes according to the Web Service Business Process Execution Language (WS-BPEL) specification. The modeling of the business processes are conducted with Business Process Execution Language (BPEL) which is a popular orchestrator of Service Oriented Architectures (SOA) based system through the description of workflow. This approach is also integrated to the Service Oriented Software Engineering (SOSE) design technique. The integration of this approach aims the development of complex business processes more robust and implementation of them more accurate and low error prone. Moreover, this approach will also decrease the development cost of the system and rework in the implementation phase. The implementation of the approach is also conducted and it is integrated to the existing service oriented architecture based tool named Service Oriented Software Engineering Tool (SOSECASE). This integration forwards the SOSECASE a step forward to an all-in-one service oriented architecure software development tool.

Keywords: Service Oriented Architecure, Service Oriented Software Engineering, Web Services Business Process Execution Language, Validation of Business Process

ÖZ

SERVİS YÖNELİMLİ YAZILIM MÜHENDİSLİĞİ TASARIMLARINA
STATİK ANALİZ YAKLAŞIMI

Çermikli, Can
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ali Hikmet Doğru

Eylül 2010, 61 sayfa

Bu tezde, web servisleri iş yürütme süreci dili (WS-BPEL) teknik özelliklerine göre doğru iş süreçleri geliştirmek amacıyla statik bir analiz yaklaşımı tanıtıldı. İş süreçlerinin modellemesi servis yönelimli mimarilerin (SOA) herkes tarafından bilinen bir orkestra yöneticisi olan iş süreci yürütme dili (BPEL) kullanılarak iş akışının tanımı yoluyla gerçekleştirildi. Bu yaklaşım servis yönelimli yazılım mühendisliği (SOSE) tasarım tekniğine entegre de edildi. Bu yaklaşımın entegre edilmesi kompleks iş süreçlerinin geliştirilmesini daha sağlam ve bu iş süreçlerinin gerçekleştirilmesini daha doğru ve daha az hataya açık hale getirmeyi amaçlar. Bunun yanısıra, bu yaklaşım sistemin geliştirilme maliyetini ve gerçekleştirme fazındaki iş tekrarını da azaltacaktır. Bu yaklaşımın gerçekleştirilmesi de yapıldı ve hali hazırda var olan, servis yönelimli mimari tabanlı servis yönelimli yazılım mühendisliği aracına (SOSE) da entegre edildi. Bu entegrasyon SOSECASE'i hepsi bir arada olan bir servis tabanlı mimarı yazılım geliştirme aracı olmaya da bir adım yaklaştırdı.

Anahtar Kelimeler: Servis Yönelimli Mimari, Servis Yönelimli Yazılım Mühendisliği, İş

Süreci Yürütme Dili, İş Sürecinin Geçerlemesi

To My Family,

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and profound respect to my supervisor Assoc. Prof. Ali Hikmet Doğru for his expert guidance and suggestions, positive approach throughout my master study and his efforts and patience during supervision of the thesis.

I would like to express my thanks to the jury members, Assoc. Prof. Ahmet Coşar, Dr. Cevat Şener, Dr. Kıvanç Dinçer and Dr. Ahmet Tümay for reviewing and evaluating my thesis.

I would like to thank to TÜBİTAK UEKAE / G222 for supporting my academic studies.

Finally special thanks to my wife for her love, understanding and every kind of support throughout my thesis and also to my son who has grown up along with my thesis.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xiii

LIST OF FIGURES xiv

CHAPTERS

1 INTRODUCTION 1

1.1 SOSE Methodology 2

1.2 Motivation 2

1.3 Thesis Organization 3

2 BACKGROUND 4

2.1 Service-Oriented Architecture (SOA) 4

2.1.1 Structure of SOA 4

2.1.2 A Closer Look at SOA 5

2.1.3 Benefits of SOA 8

2.2 Web Services 9

2.2.1 Web Services and SOA 10

2.3 Business Process Execution Language 11

2.3.1 A Brief History of BPEL 11

2.3.2 Features of BPEL 12

2.4 SOSE Approach 14

2.4.1 SOSEML in Detail 14

2.4.2 Design Using SOSECASE 17

2.4.2.1 Construction of Hierarchical Decomposition Tree 17
2.4.2.2 Modeling Business Processes 18
3 STATIC ANALYSIS APPROACH 20
3.1 Philosophy of the Approach 20
3.2 Apache ODE 21
3.2.1 WS-BPEL 2.0 Specification 21
3.2.2 Architectural Overview 21
3.2.2.1 Components 22
3.2.3 WS-BPEL Compliance and Divergence 23
3.3 Design of the System 26
3.3.1 Class Diagram 26
3.3.2 Sequence Diagram 27
3.3.3 Studies 28
3.3.3.1 Commercial Tools 30
4 ADAPTING THE APPROACH TO THE SOSECASE 31
4.1 SOSECASE Overview 31
4.2 Extensions to the SOSECASE 32
5 A CASE STUDY: VALIDATION OF A MILITARY DEPLOYMENT PLANNING SYSTEM 37
5.1 Description of the Military Deployment Planning Software 37
5.2 Modeling the System 38
5.3 Validation of the System 39
5.3.1 Inventory Procurance Process Model 39
5.3.2 Weapons Deployment Process Model 41
5.3.3 Sensors Deployment Process Model 44
5.3.4 Unit Deployment Process Model 46
5.3.5 PTL Decisions Process Model 47
5.3.6 SRS Decisions Process Model 49
5.3.7 Task Orders Decision Process Model 51
5.3.8 Deployment Decision Support Process Model 53

5.4 Case Study Results 55
6 CONCLUSION 57
6.1 Future Work 58
REFERENCES 59
APPENDICES
A STATIC ANALYSIS FAULTS 61

LIST OF TABLES

TABLES

Table A.1 Static Analysis Faults 61

LIST OF FIGURES

FIGURES

Figure 2.1 Service Oriented Application 5
Figure 2.2 Elements of SOA (Adapted from [9]) 6
Figure 2.3 Collaborations in SOA (Adapted from [9]) 7
Figure 2.4 Roles in SOA 7
Figure 2.5 Operations in SOA 8
Figure 2.6 Artifacts in SOA 8
Figure 2.7 Activities in WS-BPEL 13
Figure 2.8 Graphical Modeling Elements in SOSEML 14
Figure 2.9 BPEL Symbols Used in SOSEML 16
Figure 2.10 General Structure of Decomposition Tree in SOSEML 18
Figure 2.11 Difference between Decomposition and Modeling in SOSE Methodology . 19

Figure 3.1 Components of Apache ODE (Adapted from [13]) 22
Figure 3.2 Class Diagram of SOSE Business Process Validation System 26
Figure 3.3 Sequence Diagram of SOSE Business Process Validation System 28

Figure 4.1 General View of the SOSECASE Main Window 32
Figure 4.2 The Current Location of the Menu Item `Project` in SOSECASE 33
Figure 4.3 The Child Elements of the Menu Item `Project` 34
Figure 4.4 Validation Result Dialog 34
Figure 4.5 Validation Messages Table 35
Figure 4.6 Validation Result Status Panel 36

Figure 5.1 Inputs and Output of Military Deployment Planning Software (Adapted from [4]) 38
Figure 5.2 Hierarchical Decomposition Tree of the Entire Military Deployment Planning Software 38
Figure 5.3 BPEL Model for the Inventory Procurance Process 40
Figure 5.4 Validation Result of the Inventory Procurance Process 40
Figure 5.5 Conditions Added for While Activities 41
Figure 5.6 BPEL Model for the Weapons Deployment Process 42
Figure 5.7 Validation Result of the Weapons Deployment Process 42
Figure 5.8 Modifying `Receive` Activity in BPEL Designer 43
Figure 5.9 Second Validation Result of the Weapons Deployment Process 43
Figure 5.10 Third Validation Result of the Weapons Deployment Process 44
Figure 5.11 BPEL Model for the Sensor Deployment Process 45
Figure 5.12 Validation Result of the Sensors Deployment Process 45
Figure 5.13 BPEL Model for the Unit Deployment Process 46
Figure 5.14 Validation Result of the Unit Deployment Process 47
Figure 5.15 BPEL Model for the PTL Decisions Process 48
Figure 5.16 Validation Result of the PTL Decisions Process 49
Figure 5.17 BPEL Model for the SRS Decisions Process 50
Figure 5.18 Validation Result of the SRS Decisions Process 51
Figure 5.19 BPEL Model for the Task Orders Decisions Process 52
Figure 5.20 Validation Result of the Task Orders Decisions Process 53
Figure 5.21 BPEL Model for the Deployment Decision Support Process 54
Figure 5.22 Validation Result of the Deployment Decision Support Process 55

CHAPTER 1

INTRODUCTION

Almost every decade, a new ”silver bullet” appears in the software development area which declares to solve the known problems that afflicted the software development in the past. The lengthened development cycles, inadequate solutions to the expectations, high maintenance costs, and the fearful cost overruns are some of those problems.

”*The quest is to find a solution that simplifies development and implementation, supports effective reuse of software assets, and leverages the enormous and low-cost computing power now at our fingertips. While some might claim that service-oriented architecture (SOA) is just the latest fad in this illusive quest, tangible results have been achieved by those able to successfully implement its principles [1].*”

An article in the Harvard Business Journal states that companies that have embraced SOA ”have eliminated huge amounts of redundant software, reaped major cost savings from simplifying and automating manual processes, and realized big increases in productivity” [2]. Moreover, SOA has attained more attractive and lasting staying power than its previous alternatives in the industry [1].

As it is stated that, in every part of the industry Service-Oriented Architecture (SOA) is attracting a lot of people. Since SOA is primarily based on standard-based technologies like XML, web services, and SOAP, its usage is moving rapidly from pilot projects to critical business projects. There is no doubt that the key standard that makes the usage of SOA in critical projects faster is Business Process Execution Language for web services (BPEL). One of the main reasons of the creation of BPEL is to address the requirements of composition of web services in a service-oriented environment. Moreover, BPEL has a capability of orchestrating the current business processes’ composed services [3].

The basic artifacts of a service-oriented application are XML, SOAP, web services and BPEL.

1.1 SOSE Methodology

Service Oriented Software Engineering (SOSE) methodology is introduced by Eren Kocak Akbiyik in his thesis study [4]. This methodology is highly based on service oriented architecture and it aims to compose a complex business system into smaller parts in order to make the design of such a complex business system design easier. In this methodology, initially a hierarchical decomposition tree is constructed in a top-down manner, and then the system is modeled by starting the leaf levels to upper levels. The construction and modeling are done in Service Oriented Software Engineering Modeling Language (SOSEML) syntax which is like Unified Modeling Language (UML). All of these construction and modeling is done on the tool called Service Oriented Software Engineering Tool (SOSECASE). This tool has a complete graphical user interface to construct the decomposition tree and model the business processes. Business Process Execution Language (BPEL) graphical designer is used to model the business processes.

1.2 Motivation

SOSE methodology is successful at composing an existing complex business process. On the other hand, since there is no compilation or validation mechanism in the tool, the designed project may not work properly in the production environment. There can be so much compilation and runtime errors which will cause so much rework and high development cost. In this thesis, in order to decrease the development cost and rework, and to make such kinds of errors available in the development time, a static analysis approach is going to be adapted to the SOSE methodology. By this approach, the business processes in the current SOSE project are going to be compiled and validated according to WS-BPEL 2.0 specification [5]. In this approach, in addition to the schema validation of the business process files which have .bpel file extension, lots of logical validations are done.

1.3 Thesis Organization

This thesis work includes six chapters. In Chapter 2, the necessary background information about service-oriented architecture, web service, business process execution language and an overview of SOSE methodology is included. Chapter 3 defines the proposed static analysis approach and its philosophy. Moreover, it describes how this approach is implemented and which technologies and APIs are used. In Chapter 4 how the proposed approach is integrated to the SOSECASE stated and also general overview of the SOSECASE is presented. Chapter 5 includes an extensive case study to represent how the proposed approach and its implementation in SOSECASE works. Finally, Chapter 6 concludes the thesis work and states the future work.

CHAPTER 2

BACKGROUND

2.1 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is one of the hot topics in software engineering in the last decade and it is commonly used in enterprise projects in the industry. Before going further, let give the definition of this architecture.

"*"Service-oriented architecture" is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic. Individually, these units can be distributed [6].*"

There are several definition of SOA, but the above one is making a clear understanding of it. Main goal of the SOA is making an architectural model which develops efficiency, agility and productivity of enterprise [7].

2.1.1 Structure of SOA

In an SOA implementation, there can be any combination of technologies, APIs, products and some other parts. But, a service-oriented application which is based on an service-oriented architecture mainly consists of services [8]. These services are generally web services and the terminology web services and services are generally used interchangeably in this context. The following is a service-oriented application and the services are invoked in a hierarchy.

In this application there are three levels which are integration services, business services and data-access services. Integration services control a flow activities and calls the necessary

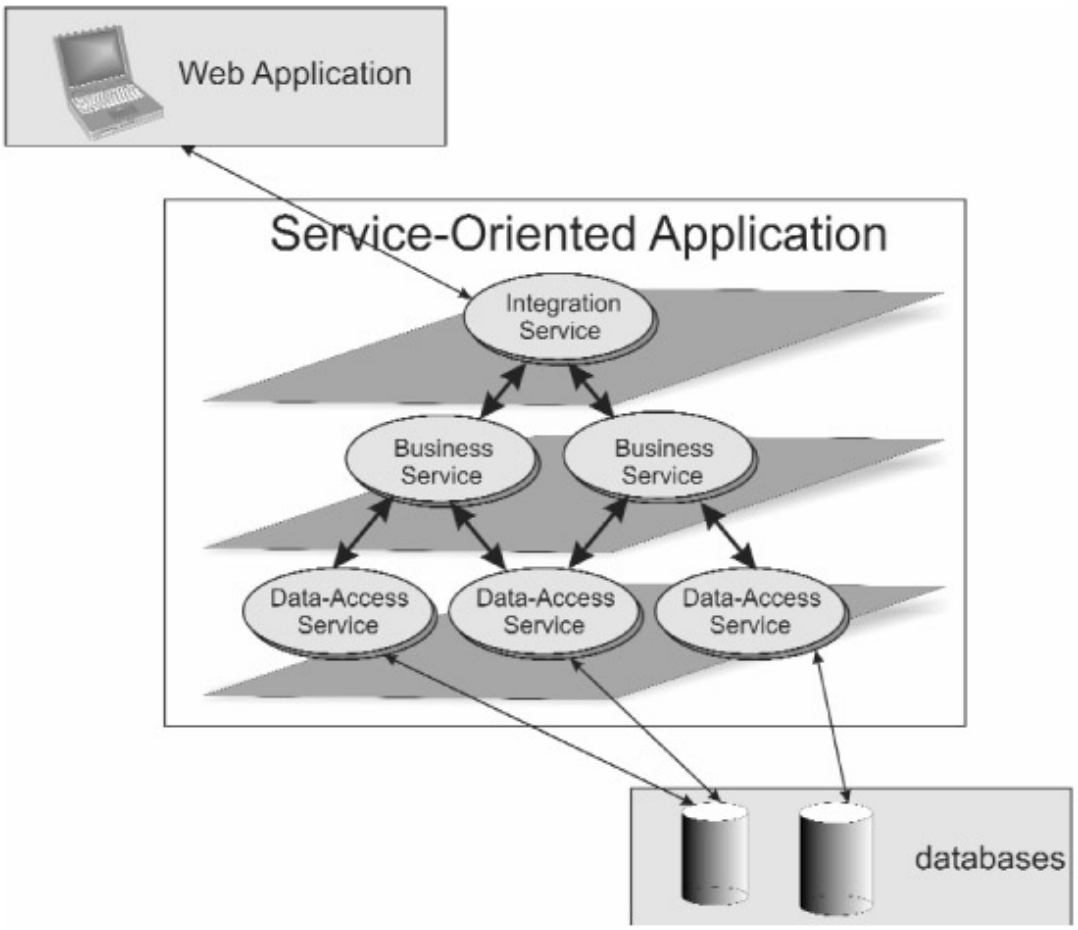

Figure 2.1: Service Oriented Application

business services. Business services are responsible for execution of low level taks. These are generally web services. The lowest level which is data-access services responsible for reading from and writing to data storage areas (e.g. databases, message queues).

2.1.2 A Closer Look at SOA

Service oriented architecture can be categorized by two elements: functional and quality of service.

As the figure shows that the left part points to functional aspects and the right part points to the quality of service aspects of the architecture:

- Functional aspects:

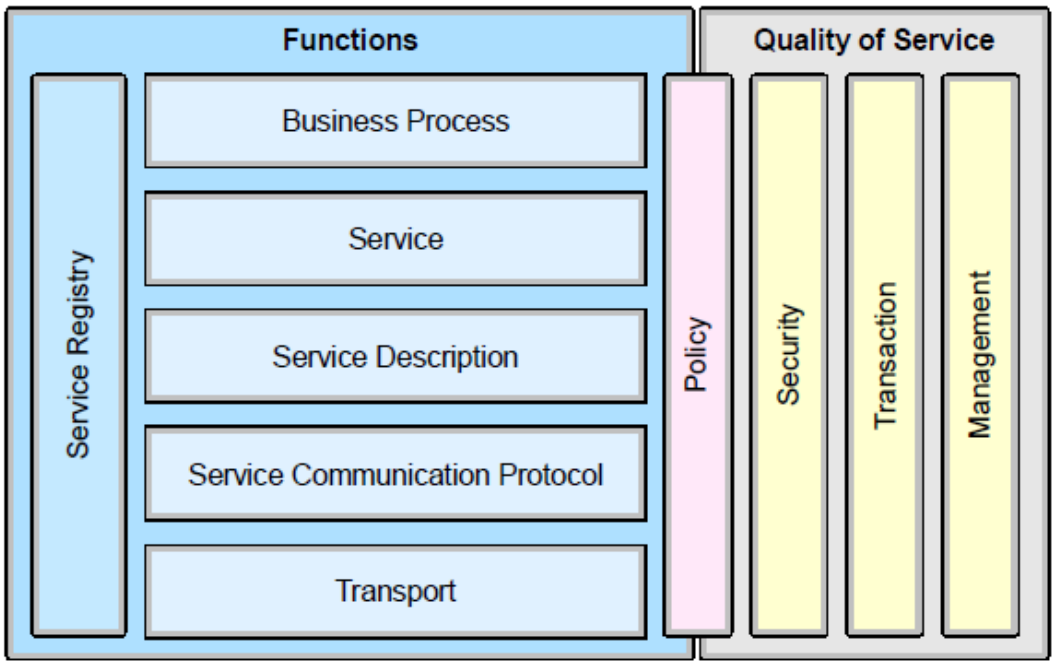

Figure 2.2: Elements of SOA (Adapted from [9])

- **Transport:** It forwards the service requests from the service consumer to service provider and vice versa.
- **Service Communication Protocol:** It is a protocol between service consumer and service provider.
- **Service Description:** It is a usable actual service.
- **Business Process:** It is a combination of services written to meet a business requirement.
- **Service Registry:** It is repository of service and data descriptions.

- Quality of service aspects:
  - **Policy:** It is a kind of aggreement which makes the service available to consumers.
  - **Security:** It is a set of rules that can be applied to authentication, authorization and access control.
  - **Transaction:** It is used to make services to return consistent result.
  - **Management:** It is responsible for managing the service provided and consumed.

The following figure shows the entities of the collaborations in the service-oriented architecture. They collaborates according to the ”find, bind and invoke” paradigm. In this paradigm service consumers query the desired service according to its criteria from the service registry. Then, if the service registry finds the requested service it returns the available service interface with its service description.

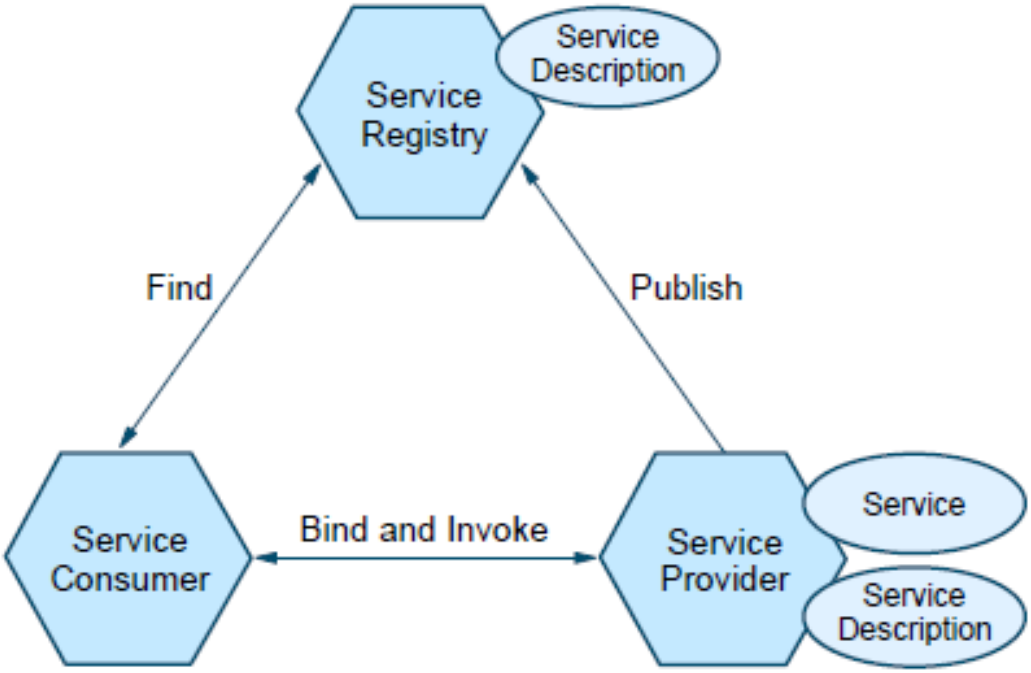

Figure 2.3: Collaborations in SOA (Adapted from [9])

The roles in the service-oriented architecture is described in the following figure. There are three roles and each entity in SOA can have one or more of these roles.

Roles in Service Oriented Architecture	
Service Consumer	It is an application, module or another service that requires a service.
Service Provider	It is an entity that accepts and executes request from consumers.
Service Registry	It contains the list of available services and enables them to be looked up.

Figure 2.4: Roles in SOA

Besides the roles, the following expresses the operations that can be done by entities.

Operations in Service Oriented Architecture	
Publish	It is the operation that is needed to make services available so that other services can invoke them.
Find	It is the operation that makes the requested services to be found according to the requestor criteria.
Bind and Invoke	It is the operation that is after the service description is retrieved, invoking the service according to the retrieved information.

Figure 2.5: Operations in SOA

The important two artifacts in SOA is:

Artifacts in Service Oriented Architecture	
Service	A service that has a callable interface in the published context by service consumers.
Service Description	It is the way of describing how service provider and service consumer will interact each other. It consists of the formats of request and response from the service.

Figure 2.6: Artifacts in SOA

2.1.3 Benefits of SOA

Service-oriented architecture provides many benefits to today's industry and help them to be able to cope with rapidly changing technology, customer requirements and other factors. The followings are the most recognizable benefits of it:

- **Leverage existing assets:** Since SOA provides a level of abstraction, it enables the organizations to implement their business functions as services. By the help of these services, a new service can be composed of them instead of rewriting all the services from scratch.

- **Easier to integrate and manage complexity:** Since SOA encapsulates all the implementation details of the services, the other service consumers just need to know the requested service interface. Moreover, it increases the flexibility of adding a new service

to the existing system. Another point is the effect of a change in the implementation and resource of a service is minimized since all of these details are hidden from the client.

- **More responsive and faster time-to-market:** The capability of composing a new service from existing services decreases the software development time. Furthermore, since an existing service which is already implemented, tested and is being used in production environment is used in the new service creation, there will also no need to test its all details just the integration point tests might be enough.

- **Reduce cost and increase reuse:** With the composition of the business services, the duplication of resources will decrease which will lower the costs.

- **Be ready for what lies ahead:** With the ease of new service creation, the organizations will be more ready to the future. Addition to the ease of service creation, the ease of changing the implementation and meeting the requirements are the other key points [9].

2.2 Web Services

It is difficult to give a single definition for web service. The following two definitions complete each other and make a good understanding.

"A Web service is an interface that describes a collection of operations that are network-accessible through standardized XML messaging. A Web service performs a specific task or a set of tasks [10]."

"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 [11]."

Web service specifications are completely independent of programming language. operations system and hardware which enables loose coupling between service providers and service consumers. The web service technology is primarily based on:

- eXtensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)

Web services combine the power of two omnipresent technologies: XML, that is universal data description language; and HTTP transport protocol.

```
Web services = XML + transport protocol (such as HTTP)
```

The followings are some of the key benefits of Web services:

- Web services are self-contained.
- Web services are self-describing.
- Web services are modular.
- Web services can be published, located, and invoked across the Web.
- Web services are language independent and interoperable.
- Web services are inherently open and standards based.
- Web services are dynamic.
- Web services are composable [9].

2.2.1 Web Services and SOA

Web services are one of the best suited technology for implementing a service-oriented architecture. Web services are self-describing and can be used over the internet easily. Web services can be developed by using any programming language, any protocol or any platform based on XML standards. Hence, these properties enables the services to be available to users at anytime, at any location.

It is important to note that web services are not the only technology that can be used implementing a service-oriented architecture. Furthermore, its reverse is also true, that is web services can be used to implement any technology other than service-oriented architecture. But, in general, in the implementation of service-oriented architecture web services is used [9].

2.3 Business Process Execution Language

"Business Process Execution Language (BPEL) is an XML-based language for creating a process, which is a set of logical steps (called activities) that guide a workflow [8]."
The following is a sample workflow:

1. Accept a request for a credit card application.

2. If the sent details are valid and appropriate, then forward the request to credit card department.

3. Otherwise, say "No" and specify what exactly is the problem.

2.3.1 A Brief History of BPEL

The first version of Business Process Execution Language for Web Services was released in July, 2002 with the effort of IBM, Microsoft and BEA. This first version had lots of inspirations from IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG specification. After one year later, a new version of BPEL4WS was released in May, 2003, and this version was more attractive and received more vendor support. By this version, the BPEL4WS specification was submitted to Organization for the Advancement of Structured Information Standards (OASIS) technical committee to make the development of specification official and open standard. After four years later, the current and still being used version was realesed in April, 2007. by OASIS.

2.3.2 Features of BPEL

WS-BPEL specifies a model and grammar for describing the behavior of a business process which interactions between the process and its partners are key parts of it. Web services provide the interfaces for those interactions and partnerLink encapsulates the structure of the relationship at the interface level. Furthermore, WS-BPEL specifies the coordination between these partners, and also the state and logic necessary for this coordination [5].

BPEL provides standards-based and platform independent solutions. Loosely coupled BPEL process extinguishes vendor dependency, decreases integration costs and also supplies interoperability. Moreover, it provides security management, logging and exception management. Another important point is that by using BPEL companies can use their existing infrastructure, service-oriented it and orchestrate it [3].

WS-BPEL uses lots of XML specifications such as WSDL 1.1, XML Schema 1.0 and XSLT 1.0.

The following is the set of available activities in BPEL.

Activity	Purpose
receive	Allows the business process to wait for a matching message to arrive.
reply	Allows the business process to send a message in reply to a message that was received by an inbound message activity (IMA).
invoke	Allows the business process to invoke a one-way or request-response operation on a `portType` offered by a partner.
assign	Used to update the values of variables with new data.
throw	Used to generate a fault from inside the business process.
validate	Used to validate the values of variables against their associated XML and WSDL data definition.
wait	Used to wait for a given time period or until a certain point in time has been reached.
empty	"no-op" in a business process.
sequence	Used to define a collection of activities to be performed sequentially in lexical order.
if	Used to select exactly one activity for execution from a set of choices.
while	Used to define that the child activity is to be repeated as long as the specified `<condition>` is true.
repeatUntil	Used to define that the child activity is to be repeated until the specified `<condition>` becomes true.
forEach	Iterates its child scope activity exactly N+1 times where N equals the `<finalCounterValue>` minus the `<startCounterValue>`.
pick	Used to wait for one of several possible messages to arrive or for a time-out to occur.
flow	Used to specify one or more activities to be performed concurrently.
scope	Used to define a nested activity with its own associated `<partnerLinks>`, `<messageExchanges>`, `<variables>`, `<correlationSets>`, `<faultHandlers>`, `<compensationHandler>`, `<terminationHandler>`, and `<eventHandlers>`.
compensate	Used to start compensation on all inner scopes that have already completed successfully, in default order.
compensateScope	Used to start compensation on a specified inner scope that has already completed successfully.
exit	Used to immediately end a business process instance within which the `<exit>` activity is contained.
rethrow	Used to rethrow the fault that was originally caught by the immediately enclosing fault handler.
extensionActivity	Used to extend WS-BPEL by introducing a new activity type.

Figure 2.7: Activities in WS-BPEL

2.4 SOSE Approach

The static analysis approach in this thesis is contructed upon Service Oriented Software Engineering (SOSE) modeling technique, which is introduced by Eren Kocak Akbiyik in his thesis study. This technique also has a modeling language which is called Service Oriented Software Engineering Modeling Language (SOSEML).

SOSEML, the graphical modeling language is based on decomposition of the system in a top down manner. It aims to simplify the design of complex business processes [4].

2.4.1 SOSEML in Detail

SOSEML is completely graphical modeling language which aims to construct a hierarachical composition tree by using four basic graphical modeling elements. The following figure shows the graphical modeling elements in SOSEML.

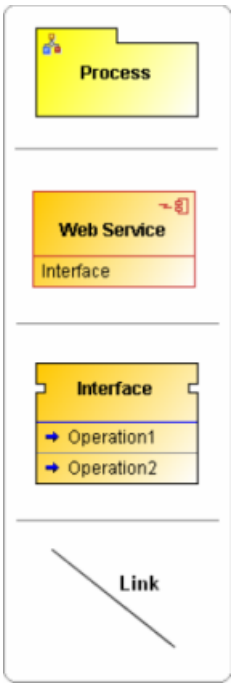

Figure 2.8: Graphical Modeling Elements in SOSEML

As it is shown in the figure Process, Web Service, Interface and Link symbols are the basic graphical modeling elements in SOSEML.

A process is represented by a yellow package symbol with a small process icon on the left upper corner. Processes are the main building blocks in a decomposition tree. The whole system and all sub processes are shown by process symbols in the model.

Web services are represented by orange boxes. The small icon on the right corner of the box implies that a web service is a remote component. A web service can publish various methods or operations in its interface where some of those methods can be used in different processes. Therefore, in SOSEML, a web service can have more than one web service interface. Names of all interfaces belong to the web service are also shown in the box symbol.

Web service interfaces are also shown in SOSE models and represented by orange boxes similar to web service symbols. An interface symbol contains the names of the operations which can be called by the requesters through this interface.

Links are represented by standard black lines in the model and are used to connect other graphical elements. A link symbol can be used between a process and another process or between a process and a web service.

After creating the decomposition tree, the next step to design the business processes. In this step, BPEL is used for modeling the business processes. Although BPEL is an XML based language, in SOSEML all the modeling can be done by using BPEL graphical editor. It produces BPEL business processes in the correct WS-BPEL syntax. The following figure shows the BPEL symbols that are available and used in SOSEML.

Name	Symbol	Description
Invoke		The <invoke> activity is used to invoke the web service operations provided by partners.
Receive		A <receive> activity is used to receive requests in a BPEL business process to provide services to its partners.
Reply		A <reply> activity is used to send a response to a request previously accepted through a <receive> activity.
Assign		The <assign> activity is used to copy data from one variable to another and construct and insert new data using expressions and literal values.
Empty		An activity that does nothing is defined by the <empty> tag.
If		The <if> activity expresses a conditional behavior.
Pick		The <pick> activity is used to wait for the occurrence of one of a set of events and then perform an activity associated with the event.
While		A <while> activity is used to define an iterative activity. The iterative activity is performed until the specified Boolean condition no longer holds true.
For Each		A <For Each> activity is also used to define iterative activities over a counter value for a group of items.
Repeat Until		A <Repeat Until> activity is also used to define an iterative activity similar to <while> activity.
Wait		A <wait> activity is used to specify a delay for a certain period of time or until a certain deadline is reached.
Sequence		A <sequence> activity is used to define activities that need to be performed in a sequential order.
Scope		A <scope> defines behavior contexts for activities. They provide fault handlers, event handlers, compensation handlers, data variables, and correlation sets for activities.
Flow		The <flow> activity provides concurrent execution of enclosed activities and their synchronization.

Figure 2.9: BPEL Symbols Used in SOSEML

2.4.2 Design Using SOSECASE

There are two main steps to design a whole system in SOSE approach.

1. Creating the hierarchical decomposition tree.
2. Creating the business process models for business processes in the tree.

2.4.2.1 Construction of Hierarchical Decomposition Tree

In SOSE modeling, creation of the hierarchical decomposition tree starts with the top down decomposition of the system. Initially, the whole system is accepted as a complex business system which retrieves the input and after some operations returns the output. In other words, the whole system starts with the root node. Since the root and the initial node corresponds to a complex business system, modeling it with all its details is very difficult. Hence, the system process is decomposed into high level sub processes. Generally, one decomposition level will not be enough so that the system processes are decomposed into such processes that can be easily understood. So, this decomposition process can be done recursively to a point that the leaf business processes are understandable enough and composition of existing web services.

The following shows the general structure of a hierarchical decomposition tree.

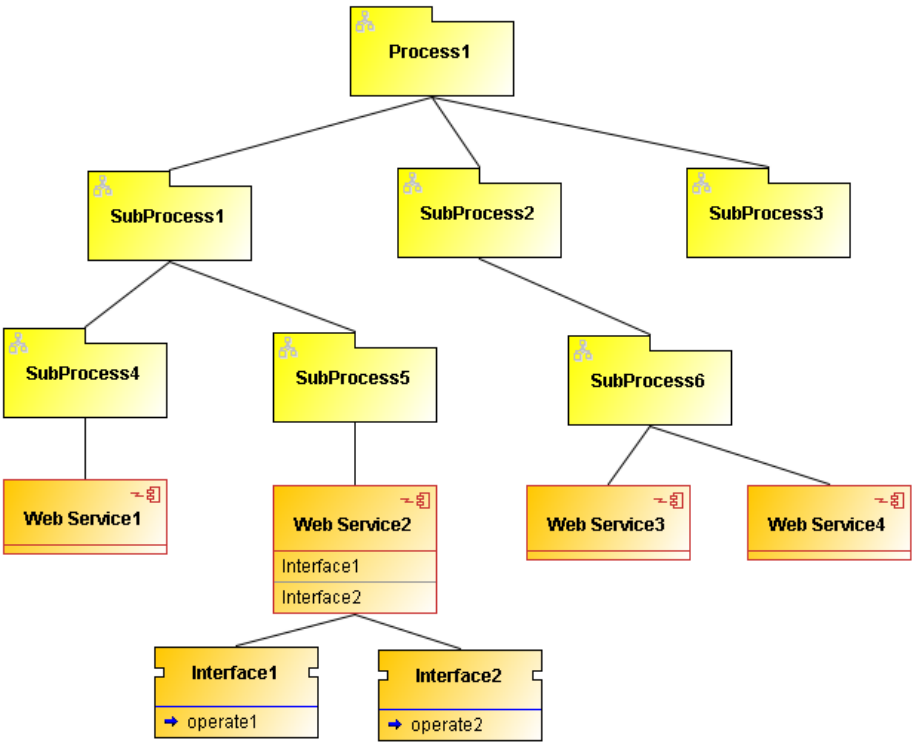

Figure 2.10: General Structure of Decomposition Tree in SOSEML

2.4.2.2 Modeling Business Processes

In SOSE methodology, the root and intermediate levels in the tree are the business processes and the leaf levels are the web services. Moreover, web services do not contain any other sub processes.

As it is stated in the previous sections, decomposition of the system is done in a top down manner. After the construction of the decomposition tree, in contrast to the decomposition approach, SOSE methodology recommends to model the system in a bottom up manner.

The following shows the difference between the decomposition and modeling approach in SOSE methodology.

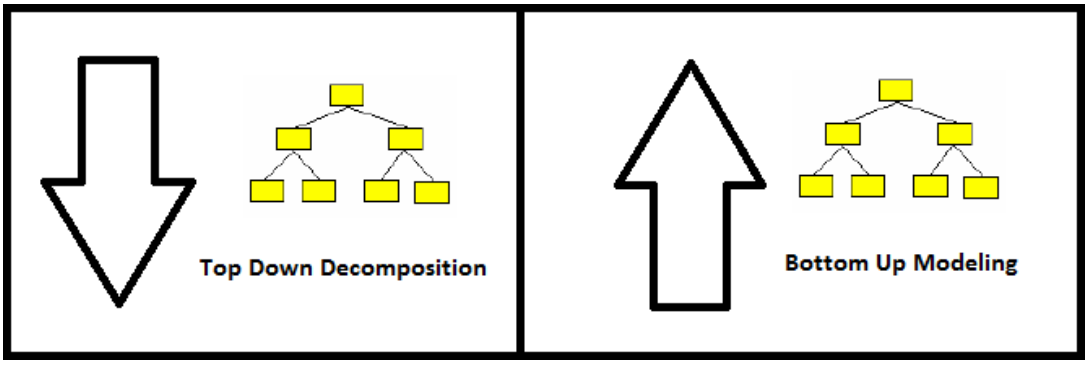

Figure 2.11: Difference between Decomposition and Modeling in SOSE Methodology

CHAPTER 3

STATIC ANALYSIS APPROACH

3.1 Philosophy of the Approach

SOSE provides an acceptable design technique for service-oriented architecture based software development. As stated in the previous chapter, it provides a way of simplifying the design phase of a complex business process by the help of decomposing the whole system in a top down manner. Furthermore, it brings many abstraction levels to the system which helps the system to be modeled easily. After creating the hierarchical decomposition tree, it starts modeling the business processes from the lowest tree level to root node.

SOSE completes its functionality after the modeling of business processes finish. But, the actual world problem starts at this point and the following question will raise inevitably:

The whole system is designed, but are the processes actually designed correctly and will execute in the real environment?

The philosophy of the approach starts with the answer of this question. The whole system is designed by using the SOSEML modeling language and its graphical editor. However, there can be still errors in the business processes which will appear when the designed system is compiled or executed. Since these errors are not seen at the design time in SOSE, after compilation or at the time to be executed, the compilation and runtime errors will raise which are not compatible with WS-BPEL specifications [5]. These errors will prevent the system to be developed in a short time and will result lots of rework. The business processes which cause the error(s) will be checked again. After the check, when the system is going to be executed or compiled, the same errors or the new errors because of the newly changed parts

can be seen. As it is seen, there is no end to this process, and its development cost is really high.

In the static analysis, the designed system is validated according to the WS-BPEL 2.0 specifications. In this analysis period, in addition to the XML namespace schema validation, some logical validations are done. A system designed with SOSECASE and then validated statically will be ready to be executed which will prevent lots of reworks.

In the development of this approach and integration of it to the SOSECASE, the Apache ODE (Orchestration Director Engine) library is used [12].

3.2 Apache ODE

Apache ODE (Orchestration Director Engine) executes the business process written in WS-BPEL standard. In addition to the execution, it provides a compilation tool before it produces executable business processes. Moreover, it can communicate with Web Services, send and receive messages, handles data manipulation and error recovery as it is stated in the business process design [12].

The Apache Software Foundation is a non-profit organization and Apache ODE is an open-source project.

3.2.1 WS-BPEL 2.0 Specification

As it is described in previous chapter, WS-BPEL specification is released by OASIS technical committee and the current version of this specification is 2.0 which is released in April, 2007.

3.2.2 Architectural Overview

The most important reason which leds to the development of ODE was to create a reliable, compact and embeddable component which has the capability of managing the long and complex business processes which are implemented using the WS-BPEL process description language. It is primary focus is developing full featured Business Process Management System (BPMS) using loosely coupled small modules.

3.2.2.1 Components

The following figure shows the current components of Apache ODE.

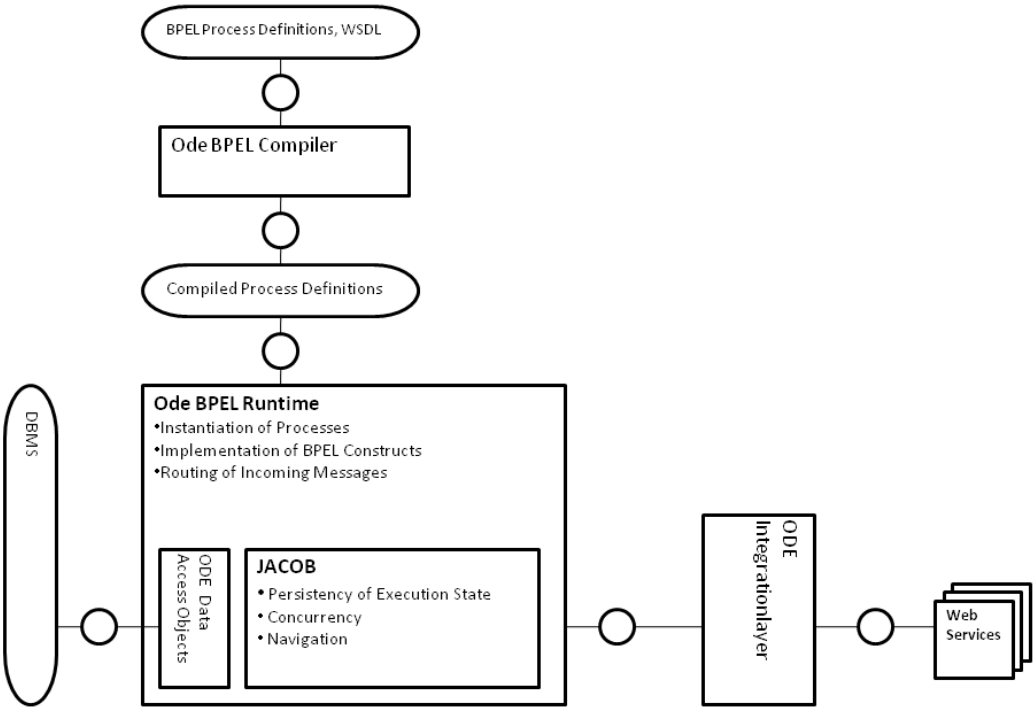

Figure 3.1: Components of Apache ODE (Adapted from [13])

The most important components in this architecture are:

- ODE BPEL Compiler,
- ODE BPEL Engine Runtime,
- ODE Data Access Objects (DAOs),
- ODE Integration Layers (ILs), and
- User Tooling.

In this thesis, ODE BPEL Compiler is used and integrated to SOSECASE. So, just ODE BPEL Compiler will be explained in detail here.

ODE BPEL Compiler

The primary task of the BPEL Compiler is to convert BPEL artifacts such as BPEL process files, WSDL documents, and schemas into an executable file format which is compatible with BPEL Engine Runtime. The compilation can result two outputs; either a successful compilation or a list of compilation errors which are related to the BPEL artifacts.

The object model of the output executable BPEL representation is similar to the underlying BPEL process documents. In spite of the high similarity, the compiled executable file has lots of additions. It has resolved the various named references in BPEL document such as variable names, the required WSDL files are internalized, and variour constructs are generated. As it is stated above, this resulted artifact is the main input of the BPEL Engine Runtime. The extension of executable file is *.cbp*.

3.2.3 WS-BPEL Compliance and Divergence

Apache ODE almost fully supports the WS-BPEL 2.0 specification. There is few activities and parts that Apache ODE diverge from the original specification. The following activities are fully compliant with the specification:

- throw
- exit
- wait
- empty
- sequence
- if
- while
- repeateUntil
- forEach
- flow
- compensateScope

- rethrow
- extensionActivity

On the other hand, the following activities have some fully support problem and diverge from the specification:

receive

There are many support problems with `receive` activity.

- `fromPart` syntax is not supported. Because of that, `variable` attribute must be used. Moreover, `variable` can reference only message variables.
- Multiple start activities are not supported. But this does not include the use of `initiate="join"`.
- Ordering guarantee does not provided as it is described in section 10.4 of specification [5].
- The `validate` attribute - if exist - is ignored. Apache ODE does not support variable validation.

reply

There are minor differences with the specification:

- `toPart` syntax is not supported.
- `variable` attributes must reference message-typed variables.

invoke

The following differences with the specification exist:

- `toPart` and `fromPart` syntax is not supported.
- `inputVariable` and `outputVariable` attributes must reference message-typed variables.

- `validate` attribute -if exist - is ignored.

assign

The differences with the specification:

- The specification requires `assign` activity to be atomic, but in ODE each `copy` is atomic.
- The specification allows the variables to be validated at the end of the assignment using `validate` attribute, but ODE does not support it.
- At the variable declaration, inline assignment is not supported.
- The specification uses `queryLanguage` attribute to determine the language used in assignments, but ODE uses `expressionLanguage` attribute.

pick

The `pick` activity also has the same compatibilty problems with `receive` activity.

scope

In the version 1.2/2.0 of ODE, the `scope` activity is fully supported. But in ODE 1.0/1.1 does not support isolated scopes. So, `isolated` and `exitOnStandardFault` attributes on `scope` elements are interpreted as if they do not exist.

compensate

This activity is not fully compliant with the specification and has same effect and syntax as `compensateScope`.

validate

This activity is implemented in ODE yet. So, the business process which has `validate` activity will cause compilation failure.

3.3 Design of the System

In this section, the class diagram and sequence diagram of the system will be described.

3.3.1 Class Diagram

The following class diagram shows the most important classes in the SOSE business process validation system.

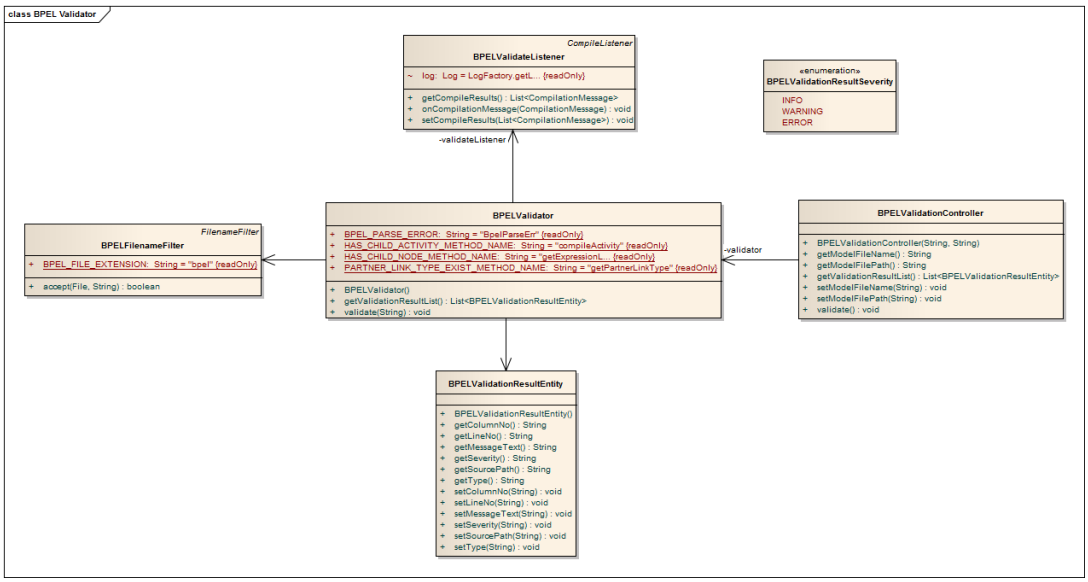

Figure 3.2: Class Diagram of SOSE Business Process Validation System

BPELValidationController

As its name says, it is the controller class of the corresponding graphical user interface class (`BPELValidationDialog`). Its main functionality is to make the business logic on the user interface. It validates and compiles all the business process in the SOSE project and then it lists the corresponding result with its details. This class delegates the validation logic to `BPELValidator` class.

BPELValidator

It is the main class that makes the validation and compilation of the business processes. It takes the business process files and then by using the Apache ODE API, it returns the val-

idation results. There are 95 static analysis fault in the system and these are shown in the validation result panel. Details of these static analysis faults are shown in Appendix A.

BPELValidationListener

This class is used for listening the Apache ODE API during the compilation process and store the validation results. If a compilation messega occurs, it adds the related compilation message to the validation result list.

BPELFilenameFilter

As its name states, it is used to filter the business process files which have .bpel file extension in the current SOSE project and it implements the `java.io.FilenameFilter` API class.

BPELValidationResultEntity

This class is an entity class which holds the validation result message details. After the compilation, the compilation result messages are converted to this class and the details of them are shown in the corresponding columns of the GUI table.

BPELValidationResultSeverity

It is an enumaration class and it holds compilation result message severity type. Currently there are three severity types: INFO, WARNING and ERROR.

3.3.2 Sequence Diagram

The Figure 3.3 indicates the basic sequence of the validation system.

All the details of the sequence is not shown in the diagram to make it simpler to understand. As it is seen, the validation activity is initially triggered by the user clicking the `Validate` button. This button click event is fired by the `BPELValidationDialog` class and it calls the `validate` method of the controller class which is `BPELValidationController`. The controller class extracts the business process file names in the current project and then passes these filenames to the `BPELValidator` class as an argument. `BPELValidator` class iterates over these filenames and validates each of the business process files. After the validation of all the business process files completes, the validation result messages are constructed and the

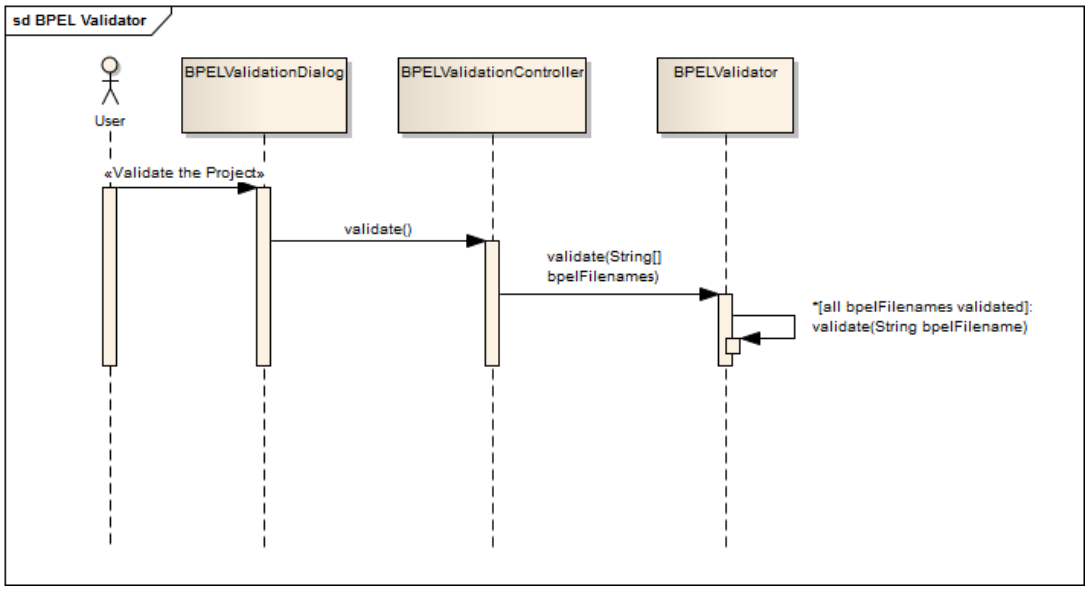

Figure 3.3: Sequence Diagram of SOSE Business Process Validation System

controller class uses these validation result messages to show them in the validation output table.

3.3.3 Studies

The static analysis approach in this thesis aims to decrease the development cost of a complex business system and rework in SOSE, as a result business process that does what it is wanted to be done. There are many studies in order to make the designed business processes more accurate and logically correct. Some of them suggest to write unit tests for business processes and the other ones suggest analysis approaches for business processes similar to the one in this thesis. Besides these studies, there are also BPEL tools that are capable of compiling and executing the BPEL processes. These tools can be classified into the commercial ones and the open-source alternatives.

Assertion language for BPEL process interactions (ALBERT) is a language that aims the composite services to be verified at design time by checking that it correctly obeys the certain relevant properties. These properties are assertions that specify composite services' functional and non-functional properties. Moreover, ALBERT extends the validation of composite services to run time. The developers of ALBERT's goals are to provide designers with a coherent

validation framework for composite services described in BPEL. They think that ALBERT provides such a way that it is capable of validating the composite business processes both at design time and run time [14].

In the study of [15], testing and validation of BPEL processes are not done with the corresponding WSDL files. Since the incorrect definition of WSDL elements which are part of BPEL (B-WSDL Element) will affect the interaction between BPEL process and WSDL file which will result exception in the related business process. In the study, initially several defect patterns relevant to B-WSDL is defined and then it constructs the related static analysis method. This method can find the B-WSDL Element related defects in the constructed static analysis phase. Hence, it decreases the possibility of exceptions and errors in the run time which will result fast development and improve the robustness of the process. They build a system for the static analysis phase called BPEL Defect Testing System (BPELDTS) which is capable of analyzing and reporting the errors related to B-WSDL Elements. This tool effectively finds the defects hidden in WSDL documents.

Execution Analysis Tool for BPEL (EA4B) is a tool that aims to address the quality assurance of BPEL business processes in two aspects: inadequacy of the tool and techniques that are used for determining the BPEL process specifications and the executing the BPEL processes to observe its correctness, and changing the execution of the service to identify erroneous service. This tool is also integrated with service analysis tool (WSAT) which is capable of finding the logic errors in BPEL process and generating error traces [16].

Testing of the BPEL business process can be thought a step forward from the validation of BPEL processes. But, if a BPEL business process passes the test then it means it will also pass the validation of process since testing a not valid BPEL business process is meaningless. The topic of the study [17] is the testability of BPEL business processes. In this study, two well known testability criteria, observability and controllabilty are evaluated. In order to evaluate them, they transform an Abstract BPEL (ABPEL) specification into corresponding Symbolic Transition System (STS). After the transformation, a graph is obtained and it can be analyzed with existing testability algorithms.

3.3.3.1 Commercial Tools

The most popular commercial tools that is capable of compiling and running the BPEL business processes are the Oracle BPEL Process Manager [18], the IBM WebSphere Process Server [19], and the Microsoft BizTalk Server [20] [21].

CHAPTER 4

ADAPTING THE APPROACH TO THE SOSECASE

4.1 SOSECASE Overview

Service Oriented Software Engineering Tool (SOSECASE) is a graphical modeling tool for SOA based system design and modeling and it supports the Service Oriented Software Engineering Modeling Language (SOSEML) notation described in detail in the previous chapter. The tool provides easy-to-use and completely graphical modeling interfaces to the users for constructing system decomposition trees and creating exact BPEL process models.

By using this tool, new SOSE models can be created, edited and saved. Most of the graphical modeling concepts offered by different commercial tools such as UML editors are included in the tool. Basic graphical modeling activities such as dragging and dropping graphical elements, editing features such as cut, copy, paste, delete and find operations are supported by SOSECASE.

SOSECASE uses the Eclipse's BPEL Designer plug-in for modeling BPEL processes graphically. This plug-in is completely an open source product and externally integrated to the main tool by using Eclipse's RCP (Rich Client Platform) architecture. The BPEL editor used in SOSECASE produces pure BPEL 2.0 codes (files with extensions .bpel and .wsdl which include the whole process model description). BPEL process models designed in SOSECASE can be used in anywhere else and by any other editor that supports BPEL 2.0 specifications.

A system decomposition tree can be constructed and each process in the tree can be modeled with BPEL by using SOSECASE graphical modeling tool for each SOSE model. The Figure 4.1 shows the general view of the main window of the tool.

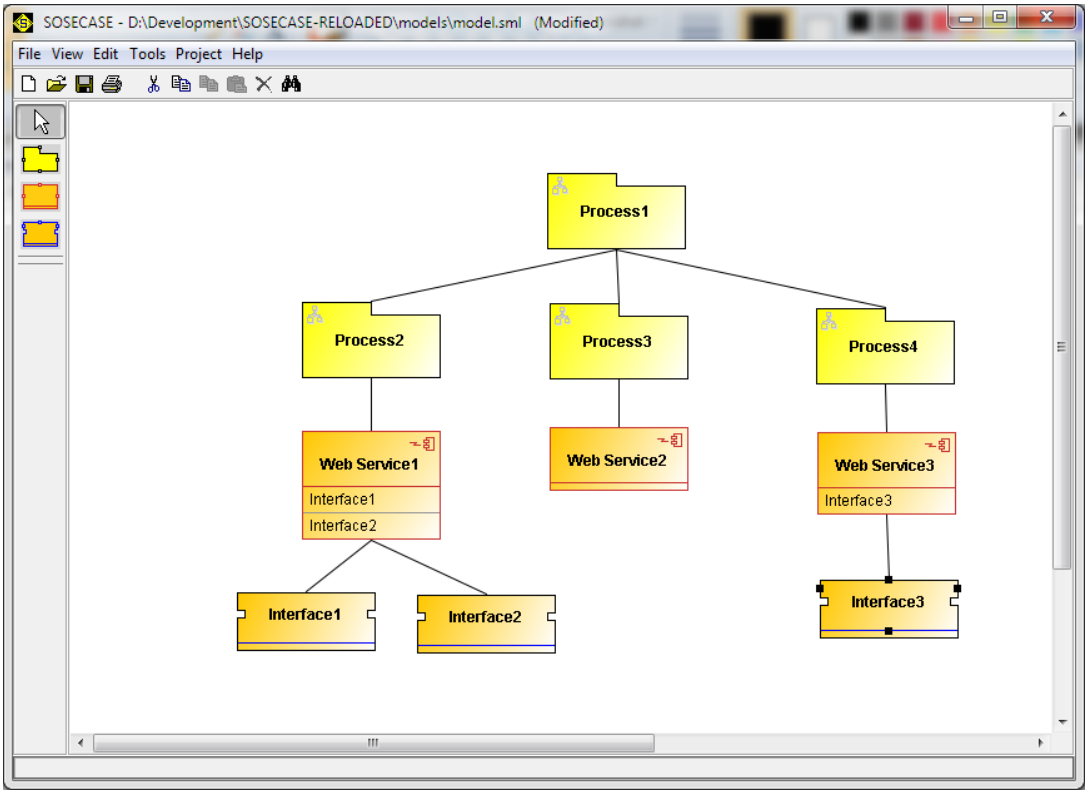

Figure 4.1: General View of the SOSECASE Main Window

There are four regions in the SOSECASE main window: main menu, top tool bar, SOSEML tool bar and main model panel. In the main menu and top tool bar, there are items and buttons for editing the model. SOSEML tool bar includes the graphical modeling elements that can be dragged and dropped over the main model panel. The final region, main model panel includes the hierarchical decomposition tree that is being modeled.

4.2 Extensions to the SOSECASE

As it is explained in the previous chapters, the aim of the BPEL business process validator is to make the design errors early available to the designer. Since the cost of testing of the design in the production environment is very high, the integration of this approach to the current SOSECASE tool is vital.

Before integrating the approach to the SOSECASE, the SOSECASE is investigated in detail in order not to ruin its simplicity and its user interface familirity to the users. Because of

that, SOSECASE design interface is not touched and remained as same. Since this kind of approach in the SOSECASE is going to be newly adapted and no functionality similar to this is implemented before in SOSECASE, a new menu item is required. A new menu item called `Project` is added to the top tool bar of the case tool. The name of the menu item is similar to the popular Integrated Development Environment (IDE) tools' such as Eclipse and the functionality and purpose of this menu item and its child menu items are also similar. In Eclipse, `Project` menu item includes the operations of project related issues such as compilation, running and debugging of the project. Since the purpose of the menu item in our approach is also similar that is to compile and validate the project, this name is the most convenient one. The Figure 4.2 shows the current location of the menu item in SOSECASE.

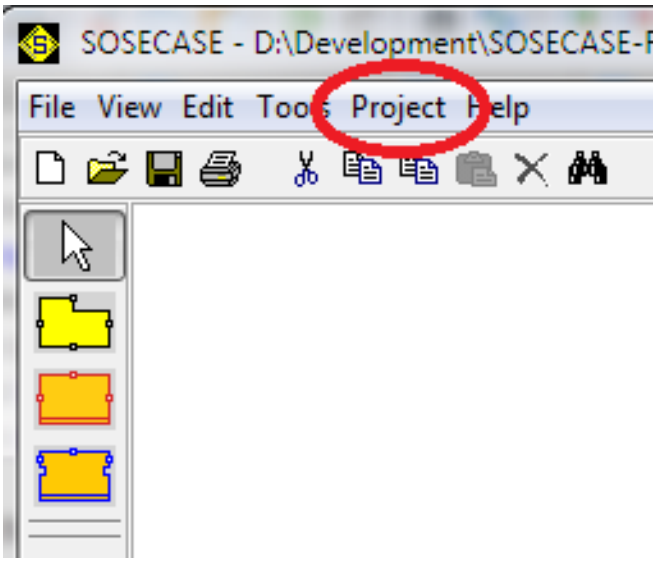

Figure 4.2: The Current Location of the Menu Item `Project` in SOSECASE

Since one approach that is related to the project is going to be integrated to the current SOSECASE tool, there is just one menu item element under the menu item `Project`. This menu item element is responsible for opening the validation dialog of the current project and its name is `Validate...`. Three dots at the end of the menu item name shows that if this element is clicked, a new dialog is going to be displayed. This convention is also very popular among the IDEs. The figure 4.3 shows the available menu item elements under the `Project` menu element in the top tool bar.

The selection of the `Validate...` operation is going to open a new dialog named as `Validation`

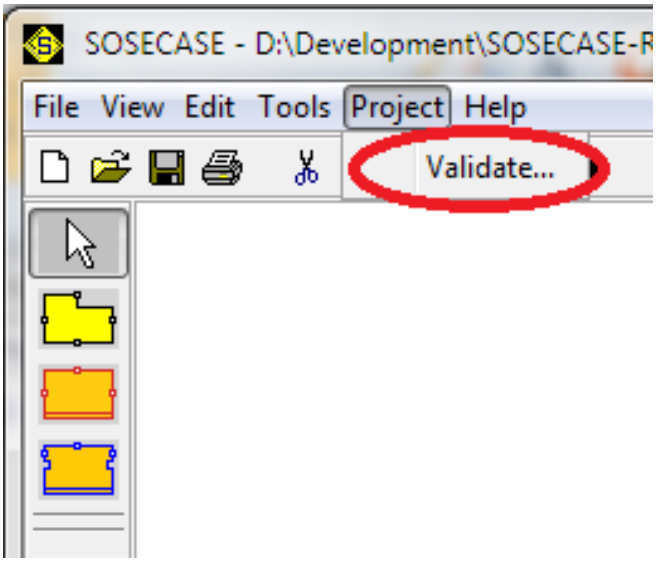

Figure 4.3: The Child Elements of the Menu Item `Project`

`Result Dialog`. The Figure 4.4 shows this dialog.

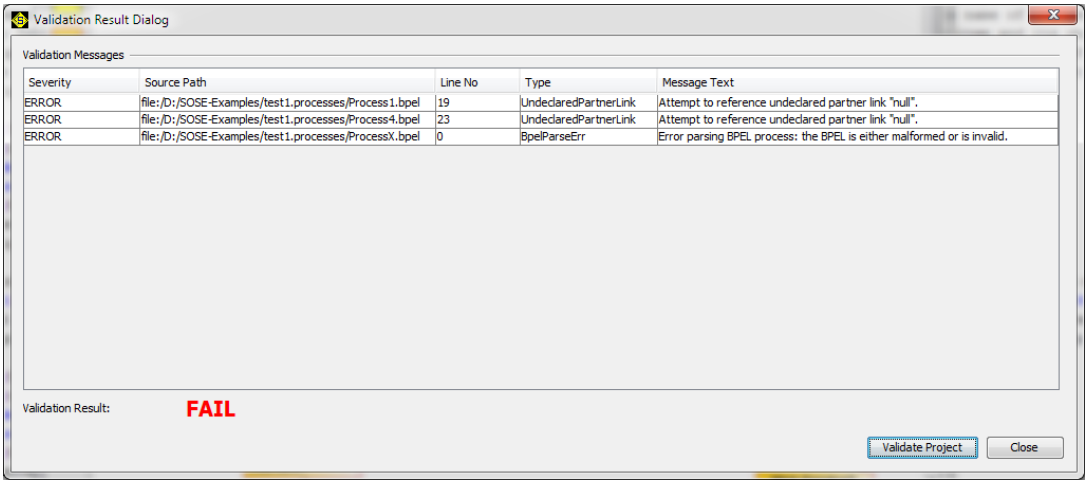

Figure 4.4: Validation Result Dialog

There are three regions in this dialog: Validation Messages Table, Validation Result Status Panel and the Operation Buttons Panel.

Validation Messages Table

In this table validation result messages are shown in detail. The Figure 4.5 shows Validation

Message Tables with sample validation messages.

Validation Messages

Severity	Source Path	Line No	Type	Message Text
ERROR	file:/D:/SOSE-Examples/test1.processes/Process1.bpel	19	UndeclaredPartnerLink	Attempt to reference undeclared partner link "null".
ERROR	file:/D:/SOSE-Examples/test1.processes/Process4.bpel	23	UndeclaredPartnerLink	Attempt to reference undeclared partner link "null".
ERROR	file:/D:/SOSE-Examples/test1.processes/ProcessX.bpel	0	BpelParseErr	Error parsing BPEL process: the BPEL is either malformed or is invalid.

Figure 4.5: Validation Messages Table

This table has five columns and each of this columns content are explained below:

- **Severity**: It expresses severity of the validation messages. The severity of a message can be one of INFO, WARNING and ERROR. The validation messages type of INFO and WARNING do not make the validation of the project fail, they just warn and gives any informative messages to the designer. On the other hand, the validation messages type of ERROR causes the validation of the project fail.
- **Source Path**: It shows absolute path of the validation message's responsible source document such as BPEL or WSDL file.
- **Line No**: It identifies the line number that causes the validation message. This information is very important to make a quick action.
- **Type**: It identifies the type of the validation messages. For instance, `BpelParseErr` is a type which occurs in case of BPEL process file is malformed or invalid. Almost all of these types corresponds to the static analysis faults explained in Appendix A.
- **Message Text**: It gives the details of the validation messages and expresses why this message has been fired.

Validation Result Status Panel

This panel gives the information of whether the validation result of the project is successful or not. If the validation of the project is successful, that is there is no validation messages type of ERROR, then a text named PASS appears that shows the validation is successful, if the validation of the project fails, then a text named FAIL appears on the panel. The figure 4.6 shows a sample text of the result of validation process.

Operation Buttons Panel

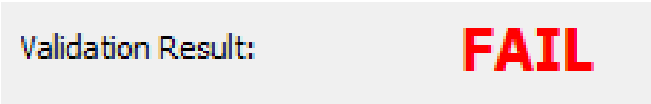

Figure 4.6: Validation Result Status Panel

In this panel, the corresponding buttons of the available operations appear. There are currently two buttons: `Validate Project` and `Close`.

- **Validate Project**: It initiates the validation process of the project. After the validation, the resulted validation messages are listed in Validation Message Tables and the result of the validation appears on the Validation Result Status Panel.
- **Close**: This button simply closes the dialog.

CHAPTER 5

A CASE STUDY: VALIDATION OF A MILITARY DEPLOYMENT PLANNING SYSTEM

In this chapter, a military deployment planning system modeled with SOSEML is validated to demonstrate the details of validation of a such a model. This sample military deployment planning system is already modeling in the thesis of Eren Kocak Akbiyik [4]. In that system, the decomposition tree and the business processes is used exactly same but the inner details of the business process models are discussed in order to show how a completely correct BPEL business processes can be modeled.

5.1 Description of the Military Deployment Planning Software

Military deployment planning software is supposed to be a SOA based decision support software that produces deployment plans for a given defense region and a military inventory including weapons and sensors [4]. The placement and task order information for the weapons, battlefield geometries and sensors are the components of a deployment plan. The main purpose of the software is to make decision about the placement of the units such as weapons, sensors, etc. and assign the necessary task orders. Finally, the battlefield geometries are added to the plan.

The Figure 5.1 adapted from [4] shows the inputs and output of the military deployment planning software.

Figure 5.1: Inputs and Output of Military Deployment Planning Software (Adapted from [4])

5.2 Modeling the System

In modeling phase, the suggested modeling technique of SOSE is used. First the hierarchical decomposition tree is constructed, and then by starting with the leaf level business processes are modeled. The details of how the hierarchical decomposition tree is constructed and business process are modeled are not given here since the main purpose is to demonstrate the validation process. The Figure 5.2 is the hierarchical decomposition tree of the entire military deployment planning software.

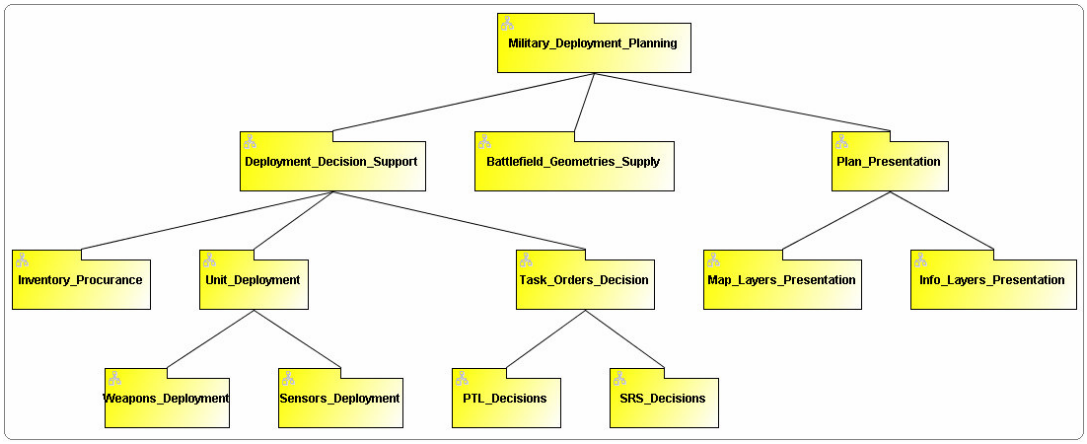

Figure 5.2: Hierarchical Decomposition Tree of the Entire Military Deployment Planning Software

5.3 Validation of the System

In this section the modeled BPEL business process for the military deployment planning software is validated. The validation of each modeled BPEL business process is evaluated separately. The modeled BPEL business processes in the original system is not modified, just the parts which cause the validation problems is modified to make it pass the validation process. Since the model file of the current system is not had, the BPEL business processes are modeled again.

5.3.1 Inventory Procurance Process Model

Inventory procurance process basically determines the existing weapons and sensors (with their working properties) in the inventory of a given army corps [4]. The Figure 5.3 depicts the designed BPEL business process for this process.

When the system is modeled like the Figure 5.3 and no further modeling is done with the activities, the validation of this process fails. The Figure 5.4 depicts the validation result of the process.

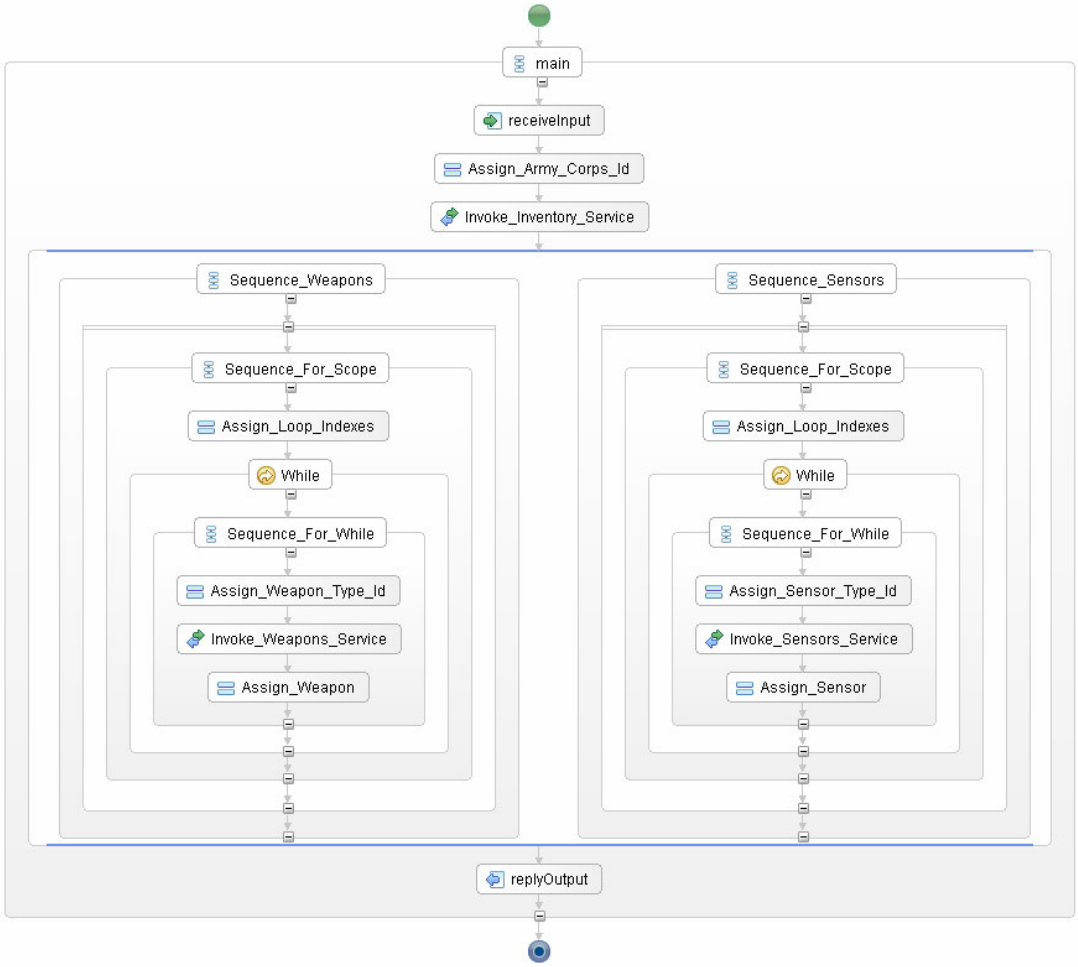

Figure 5.3: BPEL Model for the Inventory Procurance Process

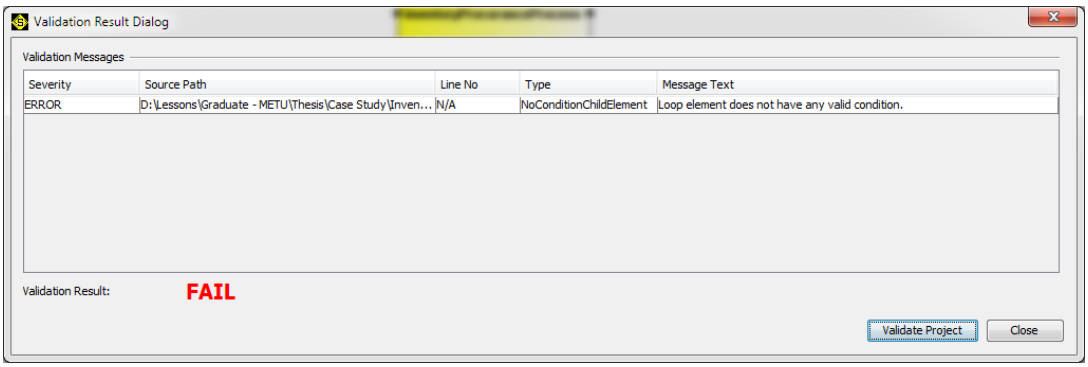

Figure 5.4: Validation Result of the Inventory Procurance Process

Since WS-BPEL 2.0 specification says that a condition must exist for a `while` activity, this process fails. In order to have a correct business process, a condition must have been added to `while` activities. The condition is added to the activities by using BPEL Designer as in the Figure 5.5.

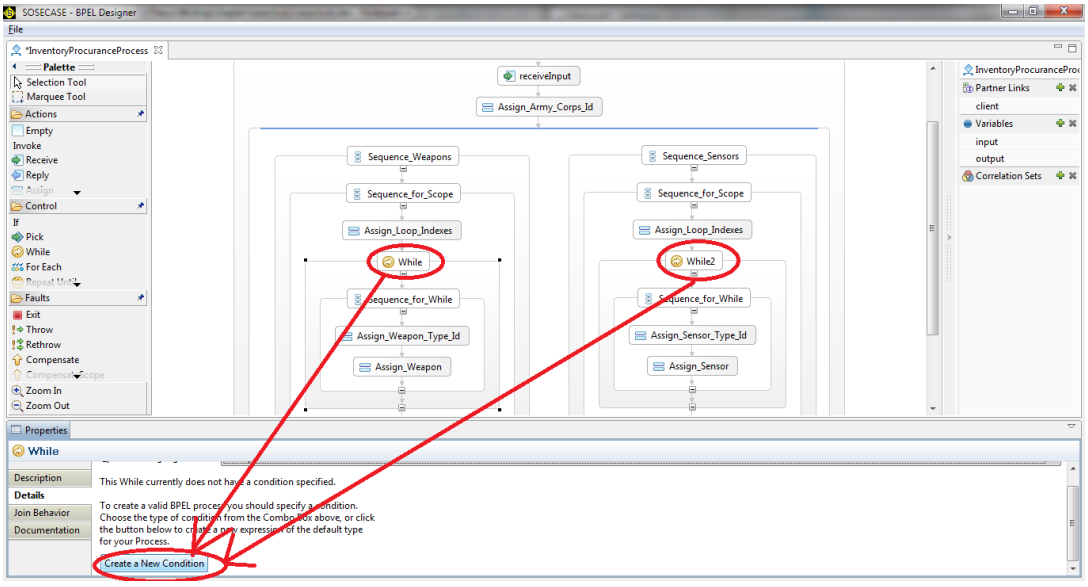

Figure 5.5: Conditions Added for While Activities

After adding the conditions for the `while` activities, there is no validation error in the process when it is validated again.

As a result, in this process the `while` activities are the erroneous ones.

5.3.2 Weapons Deployment Process Model

Weapons deployment process model produces the deployment information for the weapons. The Figure 5.6 depicts the designed BPEL business model for this process.

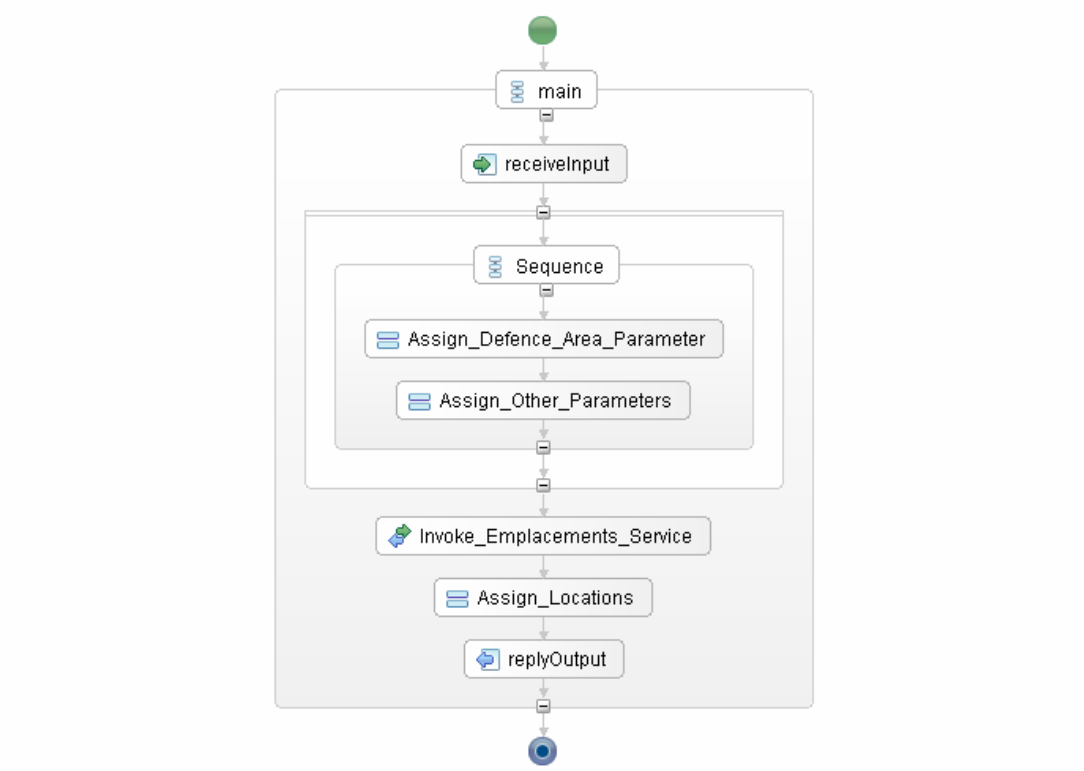

Figure 5.6: BPEL Model for the Weapons Deployment Process

The business process is modeled as in the Figure 5.6 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.7 depicts the validation result of the process.

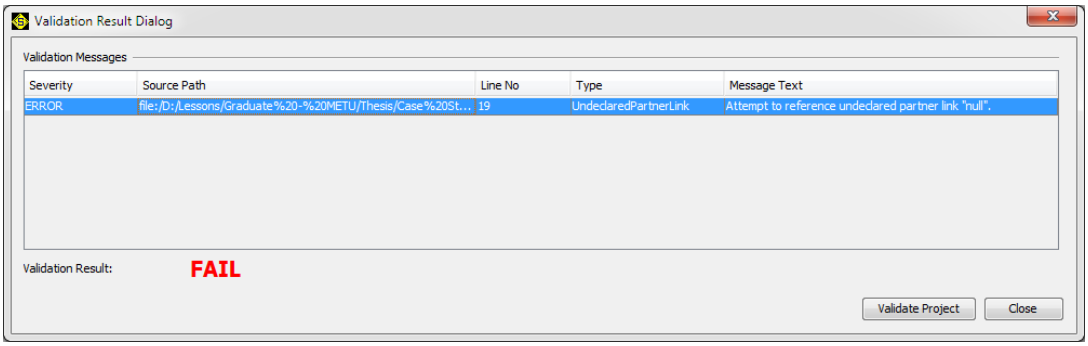

Figure 5.7: Validation Result of the Weapons Deployment Process

Since the validation result also identifies at which line the error occurs, the cause of this `UndeclaredLinkType` error is line number 19. When the line number 19 is investigated, it

is seen that no partner link is defined for the `receive` activity. After referencing the related partner link as in Figure 5.8, the validation of the process is done again.

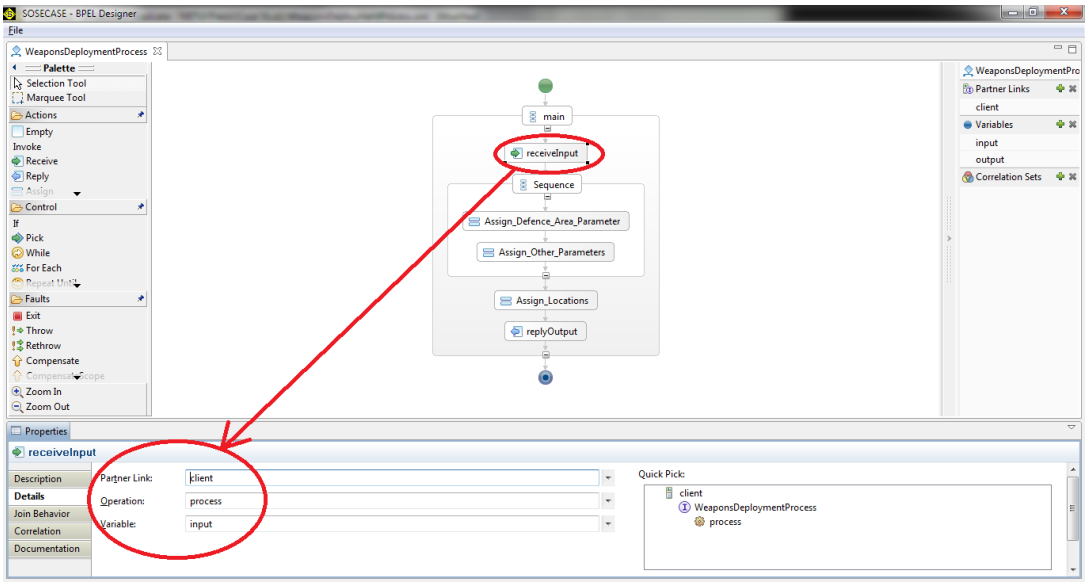

Figure 5.8: Modifying `Receive` Activity in BPEL Designer

But, after adding the partner link, the validation fails because of no operation is declared for the `receive` activity. The Figure 5.9 depicts the related validation message.

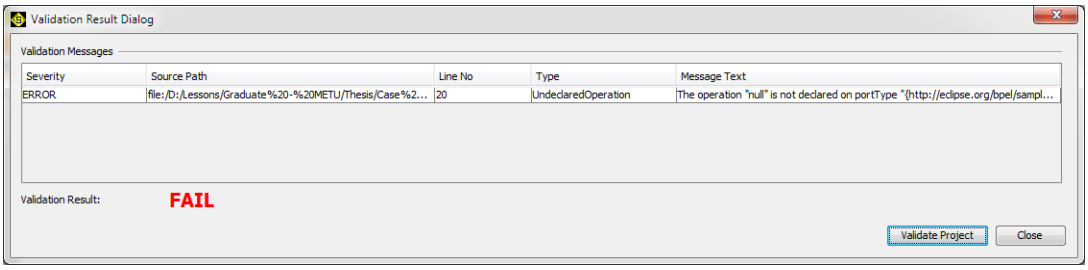

Figure 5.9: Second Validation Result of the Weapons Deployment Process

Declaring the operation for the `receive` as in Figure 5.8 is not enough since the validation fails again because of no variable is declared for the activity. The Figure 5.10 depicts the related validation message.

After adding the related variable as in Figure 5.8, the validation of the process passes and correct BPEL process is created.

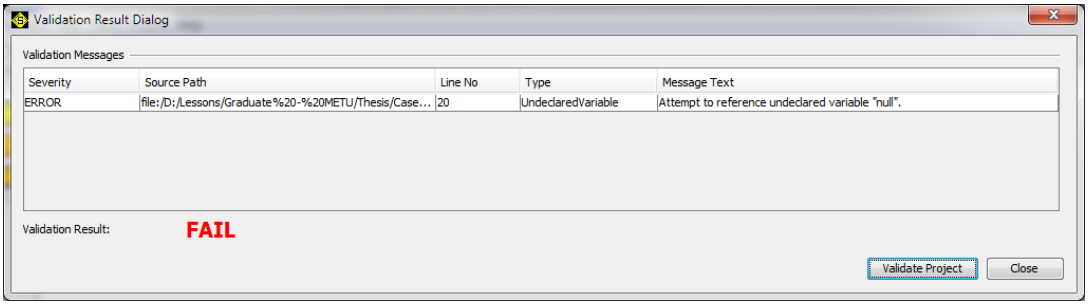

Figure 5.10: Third Validation Result of the Weapons Deployment Process

As a result, in this process the `receive` activity is the erroneous one. Moreover, since all the details of the implementation of this model is not done here just the errors related to the `receive` seen, but if the details are implemented, the potential of raising new validation errors is high.

5.3.3 Sensors Deployment Process Model

Sensors deployment process produces the deployment information for the sensors. There is one web service this process used named `SensorCoverageService`. The Figure 5.6 depicts the designed BPEL business model for this process.

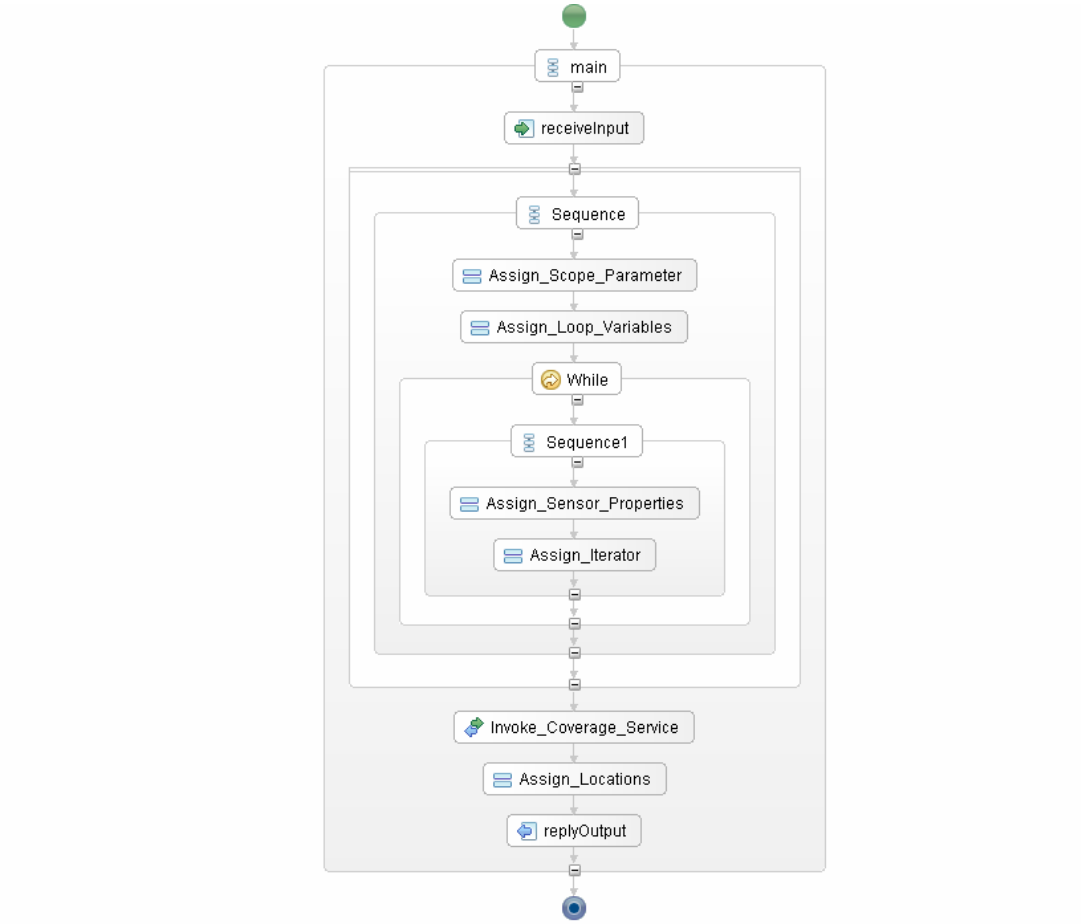

Figure 5.11: BPEL Model for the Sensor Deployment Process

The business process is modeled as in the Figure 5.11 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.12 depicts the validation result of the process.

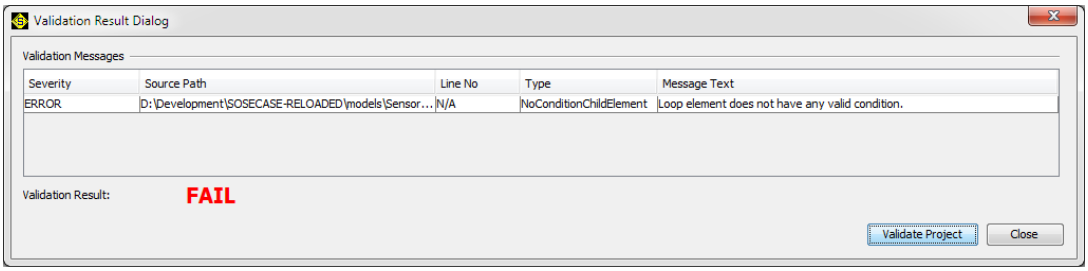

Figure 5.12: Validation Result of the Sensors Deployment Process

In this process, the validation fails again because of the `while` activity. It is the same reason

as in the Inventory Procurance Process Model: There is no conditional child element in the activity. After adding the related child condition, the validation passes.

As a result, in this process the `while` activity is the erroneous one.

5.3.4 Unit Deployment Process Model

Unit deployment process produces the deployment information for all units. Weapons deployment and sensors deployment sub processes are used as web services in this process. The Figure 5.13 depicts the designed BPEL business model for this process.

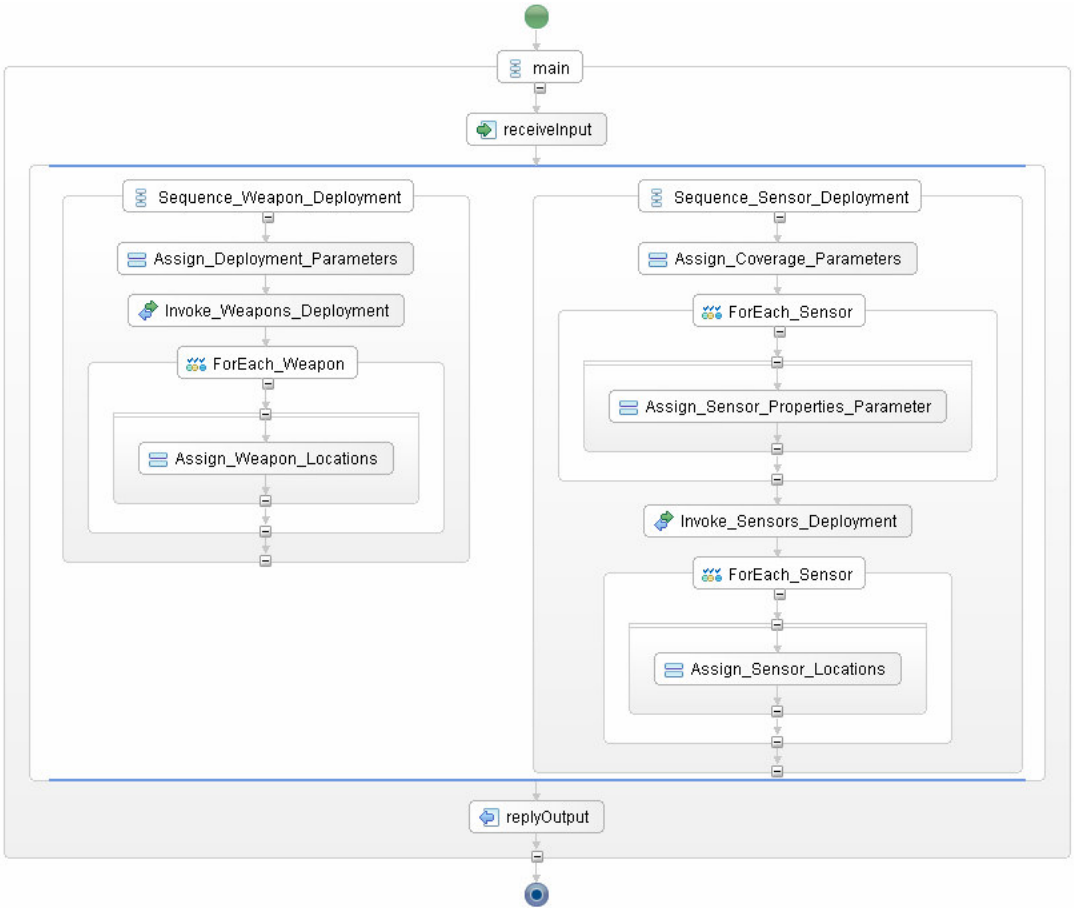

Figure 5.13: BPEL Model for the Unit Deployment Process

The business process is modeled as in the Figure 5.13 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.14 depicts the validation result of the process.

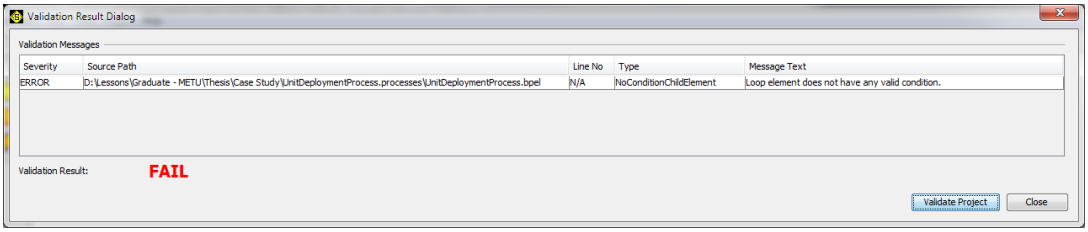

Figure 5.14: Validation Result of the Unit Deployment Process

The validation of BPEL process model is failed because of the `forEach` activities. As in the `while` activity, this activity must also have at least one conditional child element. Since three `forEach` activities in the model do not have any conditional child element. After adding related conditional elements to the activities, the validation passes.

As a result, in this process the `forEach` activity is the erroneous one.

5.3.5 PTL Decisions Process Model

PTL decisions process produces PTL task orders for the weapons. The only web service it uses is called PTLAnalyserService. The Figure 5.15 depicts the designed BPEL business model for this process.

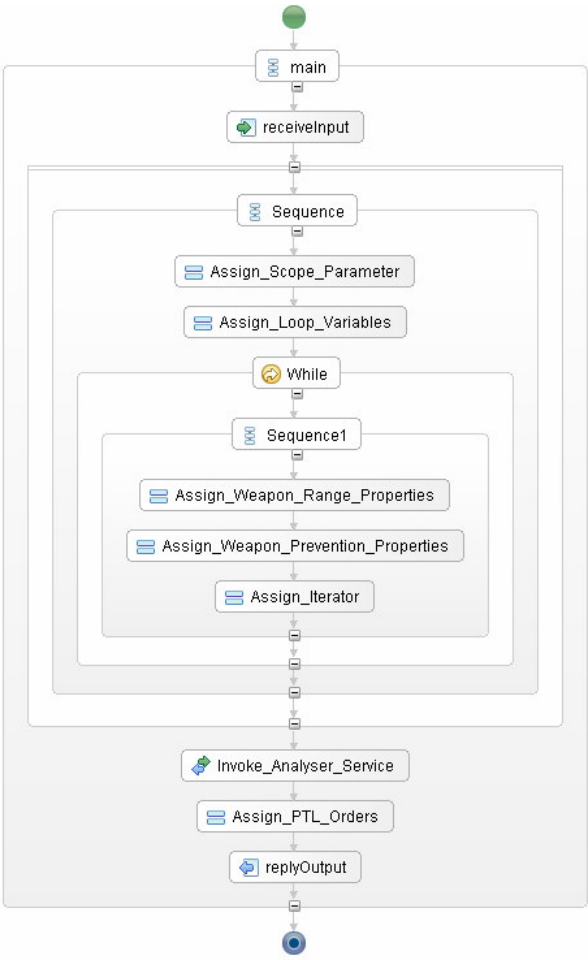

Figure 5.15: BPEL Model for the PTL Decisions Process

The business process is modeled as in the Figure 5.15 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.16 depicts the validation result of the process.

The validation of BPEL process model is failed because of the `while` activity as it is in the Inventory Procurance Process Model. After adding the related conditional child element, the validation passes.

As a result, in this process the `while` activity is the erroneous one.

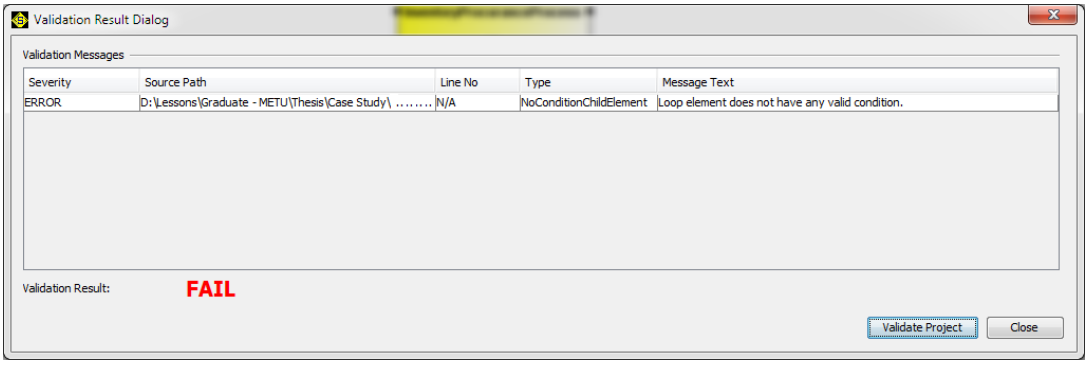

Figure 5.16: Validation Result of the PTL Decisions Process

5.3.6 SRS Decisions Process Model

SRS decisions process produces SRS task orders for the sensors. The only web service it uses is called SRSAnalyserService. The Figure 5.17 depicts the designed BPEL business model for this process.

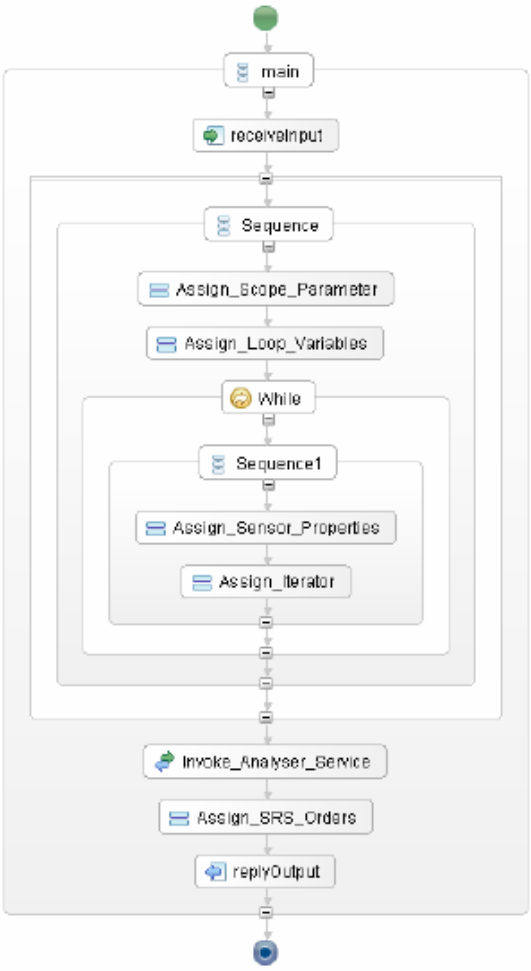

Figure 5.17: BPEL Model for the SRS Decisions Process

The business process is modeled as in the Figure 5.17 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.18 depicts the validation result of the process.

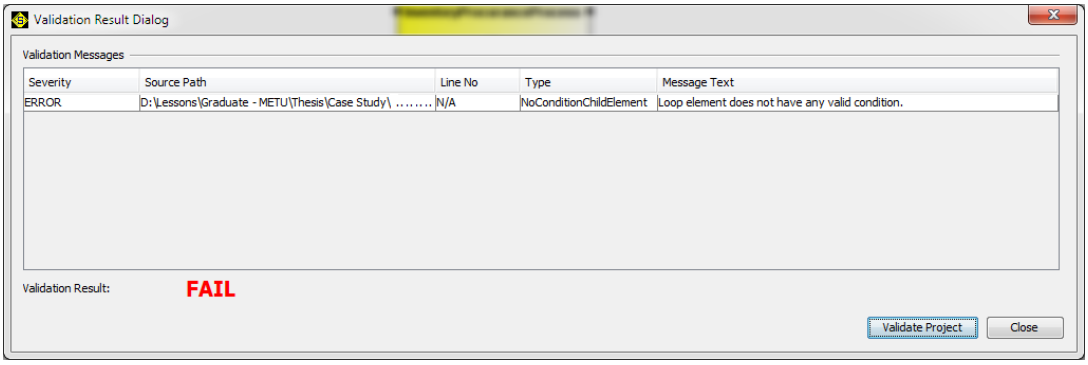

Figure 5.18: Validation Result of the SRS Decisions Process

The validation of BPEL process model is failed because of the `while` activity as it is in the Inventory Procurance Process Model. After adding the related conditional child element, the validation passes.

As a result, in this process the `while` activity is the erroneous one.

5.3.7 Task Orders Decision Process Model

Task orders decision process produces the PTL orders for the weapons and SRS orders for the sensors. As it can be predicted, PTL decisions and SRS decisions sub processes are used as web services by this process. The Figure 5.19 depicts the designed BPEL business model for this process.

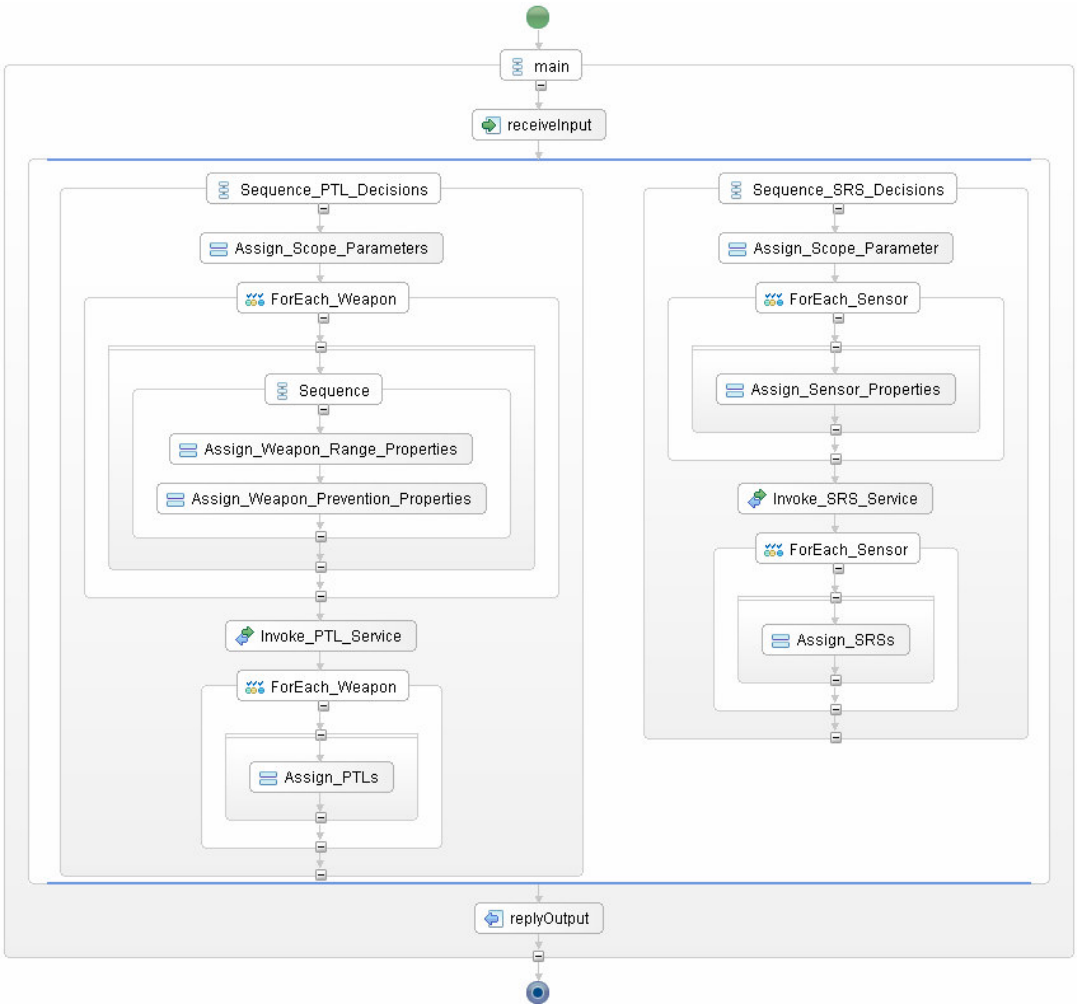

Figure 5.19: BPEL Model for the Task Orders Decisions Process

The business process is modeled as in the Figure 5.19 and no extra modification is done. As it is seen, this BPEL process model is similar to the Unit Deployment Process Model. When the designed process is validated, the validation fails. The Figure 5.20 depicts the validation result of the process.

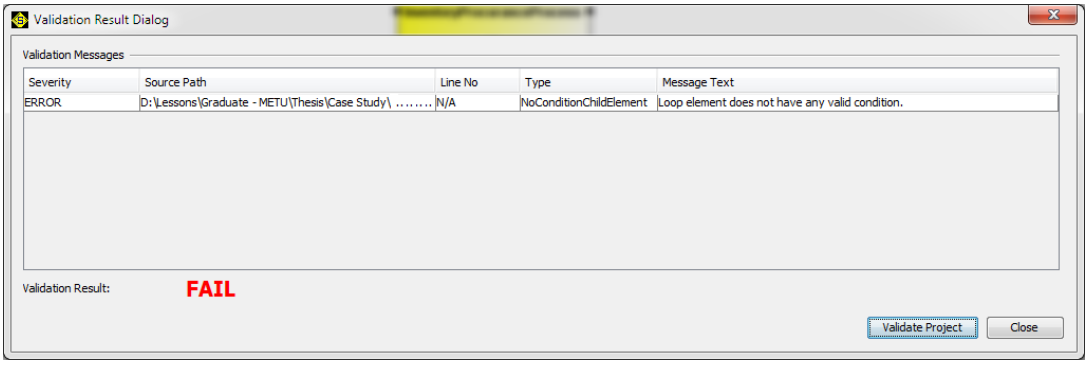

Figure 5.20: Validation Result of the Task Orders Decisions Process

The validation of BPEL process model is failed because of the `forEach` activities as it is in Unit Deployment Process Model. The four `forEach` activities do not have any conditional child element as it is specified in the WS-BPEL specification. When the related conditional child elements are added, the validation passes.

As a result, in this process the `forEach` activities are the erroneous ones.

5.3.8 Deployment Decision Support Process Model

Deployment decisions support process obtains the inventory and makes decisions for the placements and task orders of the units. Inventory procurance, unit deployment and task orders decisions sub processes are used as web services by this process. The Figure **??** depicts the designed BPEL business model for this process.

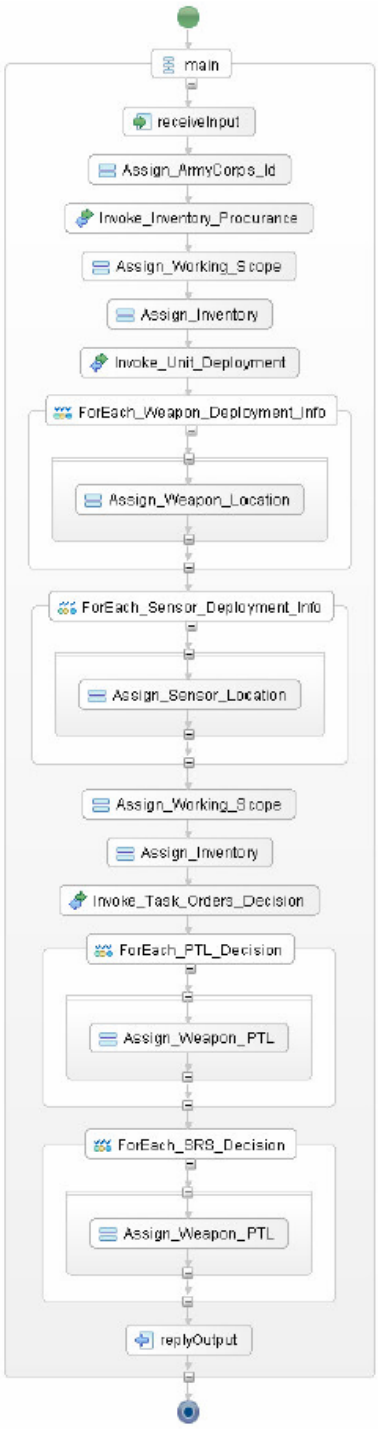

Figure 5.21: BPEL Model for the Deployment Decision Support Process

The business process is modeled as in the Figure 5.21 and no extra modification is done. When the designed process is validated, the validation fails. The Figure 5.22 depicts the validation

result of the process.

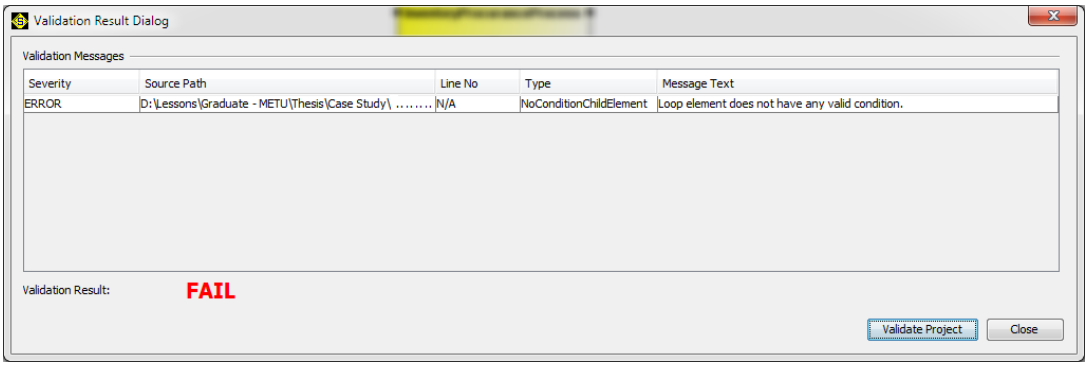

Figure 5.22: Validation Result of the Deployment Decision Support Process

The validation of BPEL process model is failed because of the `forEach` activities since they do not have any conditional child elements. When the related conditional child elements are added to the process model, the validation passes.

As a result, in this process the `forEach` activities are the erroneous ones.

5.4 Case Study Results

In the case study, a subset of the original military deployment planning system is used. In the original system, there were 13 BPEL business processes and eight of them are evaluated. Before going further to the results, it is important to point that not all the details of these BPEL business processes are implemented and that is the reason why the number of validation errors is low. Another point that is one of the reason that the number of validation errors is low is the used BPEL business processes were not so much complex, the processes just use `while`, `forEach`, `sequence` and other simple activities.

In validation process, it is seen that even if the validated BPEL business process is not so complex, it is error prone too. BPEL designer in SOSECASE tool provides modeling BPEL business processes that are syntactically correct according to the WS-BPEL namespaces not to the WS-BPEL specification. Some of the business processes are also correct according to the WS-BPEL specification but when the details of process are designed, some errors raise which is skipped by the BPEL designer of the SOSECASE. Statically analyzing the military

deployment planning system shows that this approach integrated to the SOSECASE is valuable because it shows the logical compilation or runtime errors before the whole system is moved to the production environment. It clearly detects the validation errors which are not correct according to the WS-BPEL specification. In this study, it finds the validation errors in all eight BPEL business processes.

CHAPTER 6

CONCLUSION

In this study, a static analysis approach is integrated to the service oriented software engineering modeling technique. This static analysis approach requires a designed business process to be correct as it is defined in WS-BPEL specification. In this approach, besides the namespace validation of the BPEL business processes, logical errors are detected. The main purpose of the approach is to identify the erroneous points in the business process early in the design phase of the business process. Moreover, the development cost and rework in the implementation of the target system will decrease by determining and fixing the undesired validation errors early in the design phase. Integrating this approach to the SOSE modeling technique makes a step forward to a runnable system designs developed in SOSECASE tool.

In the case study, a proposed system in the thesis [4] is modeled again in SOSECASE tool. In order to show how the static analysis approach finds the erroneous points during the modeling business processes, just the basic design errors in the BPEL processes are shown. Initially, the hierarchical decomposition tree of the system is constructed and then modeling of the system is done. There were 14 processes and sub processes in the system, and eight of them are used in the validation process. During the validation, it is seen that almost all of the processes are not so much complex, the validation errors raised. By this way, the designer of the system is warned before s/he faces the fact before moving it to the production environment. Furthermore, it is shown that the integration of the approach to the SOSECASE increased the efficieny of the tool and made it more attractive design tool. The combination of the success of SOSECASE in the development of a complex business system and the success of this validation approach completed each other.

The proposed static analysis approach clearly identifies compilation, runtime and logical er-

rors of complete business process of SOSE design. In other words, the subprocesses of the decomposed business process are validated so that all the hierarchical subprocesses and finally the main root business process are validated as a whole. Moreover, the integration of this approach to SOSECASE is easier than the other similar studies because its interfaces are well defined and easy to follow than others. Regarding the performance issue of the approach, it is seen that it can validate the whole business processes in the hierarchical decomposition tree in 60 seconds.

6.1 Future Work

As future work, adding a runtime engine to the SOSECASE will provide that the runtime behaviour of the system will be observed and the necessary corrections can be made easily in the tool. So, the tool will be an all-in one service-oriented architecture tool at the end.

Another future work is adding a capability of monitoring the data flow in the designed system. But, this can be done after adding a runtime engine to the SOSECASE to make it easier. By this ability, how the data between the business processes and corresponding web services flows and interacts with each other will be observed. Moreover, this capability will also show how the business process orchestrated visually.

REFERENCES

[1] J. Davis, *Open Source SOA*. Greenwich, CT, USA: Manning, 2009.

[2] "The next revolution in productivity." http://hbr.org/2008/06/the-next-revolution-in-productivity/ar/1, 10 August 2010.

[3] S. Blanvalet, J. Bolie, *BPEL Cookbook: Best Practices for SOA-based integration and composite applications development*. Birmingham, UK: Packt Publishing, 2006.

[4] E. K. Akbiyik, "Service oriented system design through process decomposition," August.

[5] "Web services business process execution language version 2.0 oasis standard." http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, 10 August 2010.

[6] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[7] T. Erl, *SOA Principles of Service Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008.

[8] B. Margolis, *SOA for the Business Developer-Concepts, BPEL, and SCA, First Edition*. Lewisville, TX, USA: MC Press, 2007.

[9] A. A. Mark Endrei, Jenny Ang, *Patterns: Service-Oriented Architecture and Web Services*. Researh Triangule Park, NC, USA: IBM Redbooks, 2004.

[10] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to web services architecture," *IBM Systems Journal*, vol. 41, no. 2, pp. 170 –177, 2002.

[11] "Web services architecture." http://www.w3.org/TR/ws-arch/, 30 July 2010.

[12] "Apache ode." http://ode.apache.org/index.html, 11 August 2010.

[13] "Ode - architectural overview." http://ode.apache.org/architectural-overview.html, 11 August 2010.

[14] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, "Validation of web service compositions," *Software, IET*, vol. 1, pp. 219 –232, december 2007.

[15] K. Ye, J. Huang, Y. Gong, and X. Yang, "A static analysis method of wsdl related defect pattern in bpel," in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 7, pp. V7–472 –V7–475, 16-18 2010.

[16] A. Gravel, X. Fu, and J. Su, "An analysis tool for execution of bpel services," in *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, pp. 429 –432, 23-26 2007.

[17] S. Salva and I. Rabhi, "A preliminary study on bpel process testability," in *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, pp. 62 –71, 6-10 2010.

[18] "Oracle bpel process manager." `http://www.oracle.com/technology/products/ias/bpel/`, 12 August 2010.

[19] "Ibm websphere process server." `http://www-01.ibm.com/software/integration/wps/`, 12 August 2010.

[20] "Microsoft biztalk server." `http://www.microsoft.com/biztalk/`, 12 August 2010.

[21] J. Fabra and P. A´ andlvarez, "Bpel2deneb: Translation of bpel processes to executable high-level petri nets," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pp. 496 –505, 9-15 2010.

APPENDIX A

STATIC ANALYSIS FAULTS

All static analysis faults are explained in the Table A.1.

Table A.1: Static Analysis Faults

Static Analysis Fault Code	**Static Analysis Description**
SA00001	A WS-BPEL processor MUST reject a WS-BPEL that refers to solicit-response or notification operations portTypes.
SA00002	A WS-BPEL processor MUST reject any WSDL portType definition that includes overloaded operation names.
SA00003	If the value of exitOnStandardFault of a `scope` or `process` is set to ”yes”, then a fault handler that explicitly targets the WS-BPEL standard faults MUST NOT be used in that scope.
SA00004	If any referenced queryLanguage or expressionLanguage is unsupported by the WS-BPEL processor then the processor MUST reject the submitted WS-BPEL process definition.
SA00005	If the portType attribute is included for readability, in a `receive`, `reply`, `invoke`, `onEvent` or `onMessage` element, the value of the portType attribute MUST match the portType value implied by the combination of the specified partnerLink and the role implicitly specified by the activity.
SA00006	The `rethrow` activity MUST only be used within a faultHandler (i.e. `catch` and `catchAll` elements).
SA00007	The `compensateScope` activity MUST only be used from within a faultHandler, another compensationHandler, or a terminationHandler.
SA00008	The `compensate` activity MUST only be used from within a faultHandler, another compensationHandler, or a terminationHandler.
SA00009	In the case of mandatory extensions declared in the `extensions` element not supported by a WS-BPEL implementation, the process definition MUST be rejected.

Table A.1 (continued)

Static Analysis Fault Code	**Static Analysis Description**
SA00010	A WS-BPEL process definition MUST import all XML Schema and WSDL definitions it uses. This includes all XML Schema type and element definitions, all WSDL port types and message types as well as property and property alias definitions used by the process.
SA00011	If a namespace attribute is specified on an `import` then the imported definitions MUST be in that namespace.
SA00012	If no namespace is specified then the imported definitions MUST NOT contain a targetNamespace specification.
SA00013	The value of the importType attribute of element `import` MUST be set to http://www.w3.org/2001/XMLSchema when importing XML Schema 1.0 documents, and to http://schemas.xmlsoap.org/wsdl/ when importing WSDL 1.1 documents.
SA00014	A WS-BPEL process definition MUST be rejected if the imported documents contain conflicting definitions of a component used by the importing process definition (as could be caused, for example, when the XSD redefinition mechanism is used).
SA00015	To be instantiated, an executable business process MUST contain at least one `receive` or `pick` activity annotated with a createInstance="yes" attribute.
SA00016	A partnerLink MUST specify the myRole or the partnerRole, or both.
SA00017	The initializePartnerRole attribute MUST NOT be used on a partnerLink that does not have a partner role.
SA00018	The name of a partnerLink MUST be unique among the names of all partnerLinks defined within the same immediately enclosing scope.
SA00019	Either the type or element attributes MUST be present in a `vprop:property` element but not both.
SA00020	A `vprop:propertyAlias` element MUST use one of the three following combinations of attributes: messageType and part, type or element
SA00021	Static analysis MUST detect property usages where propertyAliases for the associated variable's type are not found in any WSDL definitions directly imported by the WS-BPEL process.
SA00022	A WS-BPEL process definition MUST NOT be accepted for processing if it defines two or more propertyAliases for the same property name and WS-BPEL variable type.
SA00023	The name of a variable MUST be unique among the names of all variables defined within the same immediately enclosing scope.
SA00024	Variable names are BPELVariableNames, that is, NCNames (as defined in XML Schema specification) but in addition they MUST NOT contain the "." character.
SA00025	The messageType, type or element attributes are used to specify the type of a variable. Exactly one of these attributes MUST be used.
SA00026	Variable initialization logic contained in scopes that contain or whose children contain a start activity MUST only use idempotent functions in the from-spec.

Table A.1 (continued)

Static Analysis Fault Code	**Static Analysis Description**
SA00027	When XPath 1.0 is used as an expression language in WS-BPEL there is no context node available. Therefore the legal values of the XPath Expr (http://www.w3.org/TR/xpath#NT-Expr) production must be restricted in order to prevent access to the context node.
SA00028	WS-BPEL functions MUST NOT be used in joinConditions.
SA00029	WS-BPEL variables and WS-BPEL functions MUST NOT be used in query expressions of propertyAlias definitions.
SA00030	The arguments to bpel:getVariableProperty MUST be given as quoted strings. It is therefore illegal to pass into a WS-BPEL XPath function any XPath variables, the output of XPath functions, a XPath location path or any other value that is not a quoted string.
SA00031	The second argument of the XPath 1.0 extension function bpel:getVariableProperty(string, string) MUST be a string literal conforming to the definition of QName in [XML Namespaces] section 3.
SA00032	For `assign`, the `from` and `to` element MUST be one of the specified variants.
SA00033	The XPath expression in `to` MUST begin with an XPath VariableReference.
SA00034	When the variable used in `from` or `to` is defined using XML Schema types (simple or complex) or element, the part attribute MUST NOT be used.
SA00035	In the from-spec of the partnerLink variant of `assign` the value ”myRole” for attribute endpointReference is only permitted when the partnerLink specifies the attribute myRole.
SA00036	In the from-spec of the partnerLink variant of `assign` the value ”partnerRole” for attribute endpointReference is only permitted when the partnerLink specifies the attribute partnerRole.
SA00037	In the to-spec of the partnerLink variant of assign only partnerLinks are permitted which specify the attribute partnerRole.
SA00038	The literal from-spec variant returns values as if it were a from-spec that selects the children of the `literal` element in the WS-BPEL source code. The return value MUST be a single EII or Text Information Item (TII) only.
SA00039	The first parameter of the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) is an XPath string providing a URI naming the style sheet to be used by the WS-BPEL processor. This MUST take the form of a string literal.
SA00040	In the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) the optional parameters after the second parameter MUST appear in pairs. An odd number of parameters is not valid.
SA00041	For the third and subsequent parameters of the XPath 1.0 extension function bpel:doXslTransform(string, node-set, (string, object)*) the global parameter names MUST be string literals conforming to the definition of QName in section 3 of [Namespaces in XML].

Table A.1 (continued)

Static Analysis Fault Code	**Static Analysis Description**
SA00042	For `copy` the optional keepSrcElementName attribute is provided to further refine the behavior. It is only applicable when the results of both from-spec and to-spec are EIIs, and MUST NOT be explicitly set in other cases.
SA00043	For a copy operation to be valid, the data referred to by the from-spec and the to-spec MUST be of compatible types. The following situations are considered type incompatible: • the selection results of both the from-spec and the to-spec are variables of a WSDL message type, and the two variables are not of the same WSDL message type (two WSDL message types are the same if their QNames are equal). • the selection result of the from-spec is a variable of a WSDL message type and that of the to-spec is not, or vice versa (parts of variables, selections of variable parts, or endpoint references cannot be assigned to/from variables of WSDL message types directly).
SA00044	The name of a `correlationSet` MUST be unique among the names of all `correlationSet` defined within the same immediately enclosing scope.
SA00045	Properties used in a `correlationSet` MUST be defined using XML Schema simple types.
SA00046	The pattern attribute used in `correlation` within `invoke` is required for request-response operations, and disallowed when a one-way operation is invoked.
SA00047	One-way invocation requires only the inputVariable (or its equivalent `toPart` elements) since a response is not expected as part of the operation (see section 10.4. Providing Web Service Operations - Receive and Reply). Request-response invocation requires both an inputVariable (or its equivalent `toPart` elements) and an outputVariable (or its equivalent `fromPart` elements). If a WSDL message definition does not contain any parts, then the associated attributes variable, inputVariable or outputVariable, MAY be omitted,and the `fromParts` or `toParts` construct MUST be omitted.
SA00048	When the optional inputVariable and outputVariable attributes are being used in an `invoke` activity, the variables referenced by inputVariable and outputVariable MUST be messageType variables whose QName matches the QName of the input and output message type used in the operation, respectively, except as follows: if the WSDL operation used in an `invoke` activity uses a message containing exactly one part which itself is defined using an element, then a variable of the same element type as used to define the part MAY be referenced by the inputVariable and outputVariable attributes respectively.

Table A.1 (continued)

Static Analysis Fault Code	**Static Analysis Description**
SA00050	When `toParts` is, it is required to have a `toPart` for every part in the WSDL message definition; the order in which parts are specified is irrelevant. Parts not explicitly represented by `toPart` elements would result in uninitialized parts in the target anonymous WSDL variable used by the `invoke` or `reply` activity. Such processes with missing `toPart` elements MUST be rejected during static analysis.
SA00051	The inputVariable attribute MUST NOT be used on an Invoke activity that contains `toPart` elements.
SA00052	The outputVariable attribute MUST NOT be used on an `invoke` activity that contains a `fromPart` element.
SA00053	For all `fromPart` elements the part attribute MUST reference a valid message part in the WSDL message for the operation.
SA00054	For all `toPart` elements the part attribute MUST reference a valid message part in the WSDL message for the operation.
SA00055	For `receive`, if `fromPart` elements are used on a `receive` activity then the variable attribute MUST NOT be used on the same activity.
SA00056	A ”start activity” is a `receive` or `pick` activity that is annotated with a createInstance=”yes” attribute. Activities other than the following: start activities, `scope`, `flow` and `sequence` MUST NOT be performed prior to or simultaneously with start activities.
SA00057	If a process has multiple start activities with correlation sets then all such activities MUST share at least one common correlationSet and all common correlationSets defined on all the activities MUST have the value of the initiate attribute be set to ”join”.
SA00058	In a `receive` or `reply` activity, the variable referenced by the variable attribute MUST be a messageType variable whose QName matches the QName of the input (for `receive`) or output (for `reply`) message type used in the operation, except as follows: if the WSDL operation uses a message containing exactly one part which itself is defined using an element, then a WS-BPEL variable of the same element type as used to define the part MAY be referenced by the variable attribute of the `receive` or `reply` activity.
...	...
...	...
SA00095	For `onEvent`, the variable references are resolved to the associated scope only and MUST NOT be resolved to the ancestor scopes.