uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Amir Afrasiabi Rad

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (E-Business Technologies)

GRADE / DEGREE

E-Business Technologies

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Business Process Modeling in Web Services-based Healthcare Systems

TITRE DE LA THÈSE / TITLE OF THESIS

M. Benyoucef

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

C. Kuziemsky

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

D. Amyot

L. Peyton

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Business Process Modeling in Web Service-based Healthcare Systems

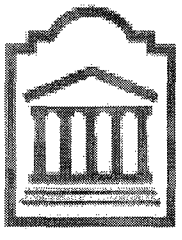
Amir Afrasiabi Rad

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the

MSc. degree in Electronic Business Technologies (e-business)



Faculty of Graduate and Postdoctoral Studies

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-61330-6
Our file Notre référence
ISBN: 978-0-494-61330-6

Abstract

Web services composition is an emerging paradigm for enabling inter and intra organizational integration, and a landscape of languages and techniques for modeling business processes in web service based environments has emerged and is continuously being enriched. With the advent of modeling standards, different business sectors are investigating the options for modeling their workflows. In terms of business process modeling, healthcare is a rather complex sector of activity. Indeed, modeling healthcare processes presents special requirements dictated by the complicated and dynamic nature of these processes as well as by the specificity and diversity of the actors involved in these processes. Little effort has been dedicated to evaluating the capabilities and limitations of modeling languages based on healthcare requirements. This thesis presents a set of healthcare requirements and proposes an evaluation framework for process modeling languages based on these requirements. The suitability of three major process-based service composition languages, namely BPEL, BPMN and WS-CDL, is evaluated.

To my beloved parents **Abbas** and **Marziyeh**, and my sisters **Parvaneh** and **Fatemeh**, who supported me with their love and encouragement

Acknowledgements

This is perhaps the easiest and hardest chapter that I have to write. Words fail to express my appreciation in a way that it should be.

First and foremost, I would like to offer my sincerest gratitude to my supervisors, Dr. Morad Benyoucef and Dr. Craig Kuziemsky, who shared with me a lot of their expertise and research insight. Without their support and thoughtful advice, this research would have never reached this point. Their deep and on-time insights gave me a clear of direction during my Master's studies. One simply could not wish for better or friendlier supervisors.

I also wish to express my appreciation to Dr. Bijan Raahemi who made many valuable suggestions and gave me constructive advice during tough times of my research.

I would like to present my special thanks to all friends who helped me during the completion of this thesis. Maryam, Vivian, Cate, and Ali who supported me with their ideas and shared their valuable knowledge with me.

It is difficult to overstate my appreciation to Dr. Ali Rahimi and his family, who supported me emotionally from the first day that I entered Canada. Also, I would like to express my special gratitude to my good friends Payam and Yasin for their true friendship and kindness.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Publications

Afrasiabi Rad, A., Benyoucef, M., Kuziemsky, C. E., 2009. On Modeling Web Service based Processes for Healthcare, Proceedings, BPSC-2009, Lecture Notes in Informatics 147, Leipzig, Germany, 23-24 March 2009, pp. 123-138

Afrasiabi Rad, A., Benyoucef, M. and Kuziemsky, C. E. (2009) Web Service based Process Modeling for Healthcare, Journal of Theoretical and Applied Electronic Commerce Research, to appear

Table of Contents

Abstract ii

Acknowledgements iv

Publications vi

Table of Contents vii

List of Tables xiii

List of Figures xiv

Table of Listings xvi

Nomenclature xvii

Chapter 1: Introduction 1

1.1 Thesis motivation and problem statement 1

1.2 Contribution 4

1.3 Methodology 5

1.4 Thesis Organization 6

Chapter 2: Healthcare Web Service Business Process Modeling Concepts 7

2.1 Business Processes 7

2.2 Service Oriented Architecture and Web Services 7

2.2.1 XML (eXtensible Markup Language) 10

2.2.2 WSDL (Web Services Description Language) 10

2.2.3 SOAP (Simple Object Access Protocol) 11

2.2.4 UDDI (Universal Description Discovery and Integration) 12

2.2.5 Web services benefits 13

2.2.6 Web services development challenges 14

2.3 Healthcare business process frameworks 14

2.4 Business Process Modeling 15

2.5 BPMN 17

2.6 Choreography and WS-CDL 18

2.7 Orchestration and BPEL 20

2.8 Workflow, Data flow, and Interaction patterns 22

2.8.1 Basic Control Patterns 22

2.8.2 Advanced branching and Synchronizing Pattern 24

2.8.3 Structural Patterns 25

2.8.4 Patterns Involving Multiple Instances (MI) 26

2.8.5 State-Based patterns 27

2.8.6 Cancellation Patterns 28

2.8.7 New Control Flow Patterns 29

2.9 Ontological Constructs 34

Chapter 3: Related Work **36**

Chapter 4: Challenges of Healthcare Processes Modeling **44**

4.1 Complexity of Processes 46

4.2 Security and Privacy 47

4.3 User Understandability 48

4.4 Optimized Models for Different Purposes 49

4.5 Evolution of Processes 49

4.6 Nested Processes and Integration 50

Chapter 5: Evaluation Framework for Business Process Modeling Languages in Healthcare **52**

5.1 Security and privacy 54

5.2 Pattern representation 55

5.3 Ontological Completeness 57

5.4 Extendibility 58

5.5 Notations 59

5.6 Modularity 60

5.7 Level of detail 61

5.8 Exception handling 62

Chapter 6: Assessment of Modeling Languages **64**

6.1 Choice of the modeling languages to be analyzed 64

6.2 Case study 65

6.3 Assessment of BPMN 69

6.3.1 Security 70

6.3.2 Pattern support 71

6.3.3 Ontological Completeness 75

6.3.4 Extendibility 77

6.3.5 Notations 77

6.3.6 Modularity 78

6.3.7 Level of Detail 78

6.3.8 Exception Handling 79

6.4 Assessment of BPEL 82

6.4.1 Security and Privacy 83

6.4.2 Pattern support 84

6.4.3 Ontological Completeness 83

6.4.4 Extendibility 85

6.4.5 Notations 85

6.4.6 Modularity 89

6.4.7 Level of Detail 90

6.4.8 Exception Handling 91

6.4.9 BPEL Implementation 89

6.5 Assessment of WS-CDL 91

6.5.1 Security and Privacy 93

6.5.2 Pattern support 95

6.5.3 Ontological Completeness 95

6.5.4 Extendibility 95

6.5.5 Notations 97

6.5.6 Modularity 97

6.5.7 Level of Detail 99

6.5.8 Exception Handling 99

6.5.9 WS-CDL implementation 100

Chapter 7: Implications for Healthcare Business Process Modeling 104

Chapter 8: Conclusions 108

References 112

List of Tables

Table 1: Comparison of modeling languages based on their ability to represent workflow and communication flow patterns, from Workflow Patterns Initiative [46] 38

Table 2: Ontology based expressiveness analysis of ebXML, BPML, BPEL4WS, and WSCI, from Green et al. [3, 13] 39

Table 3: Ontological overlap analysis of ebXML, BPML, BPEL4WS, and WSCI, from Green et al. [3, 13] 41

Table 4: BPMN Security data types, from Rodriguez et al. [20] 69

Table 5: BPMN Security elements 69

Table 6: BPMN workflow pattern support 71

Table 7: BPMN support for healthcare service based ontological constructs 76

Table 8: Pattern Analysis of BPEL 81

Table 9: Ontological constructs supported in BPEL, from Green et al. [3] 82

Table 10: Pattern Analysis of WS-CDL 94

Table 11: Ontological representation of WS-CDL, from Green et al. [3] 96

List of Figures

Figure 1: Service Oriented Architecture, from W3C [29] 8

Figure 2: Web Services Stack, from Bellwood [30] 9

Figure 3: Sequence Flow 22

Figure 4: Parallel Split and Synchronization 23

Figure 5: Exclusive Choice and Simple Merge 23

Figure 6: Multiple Choice and Multiple Merge 24

Figure 7: Arbitrary Cycle 26

Figure 8: Implicit Termination 26

Figure 9: Internet/Intranet environment for medical web services, representing data and patient exchange, from Anzbock and Dustdar [17] 42

Figure 10: Healthcare Complexity 45

Figure 11: Relation between framework criteria and healthcare's modeling challenges .. 53

Figure 12: Evaluation Framework 54

Figure 13: Scheduled Workflow Diagram, from IHE Int. [62] 65

Figure 14: SWF's Administrative and Procedure Performance process Flows 67

Figure 15: Part of SWF 68

Figure 16: Construct hierarchy in BPMN, from Rodriguez et al. [20] 75

Figure 17: Sub-Processes in BPMN 78

Figure 18: BPMN's Error End Event 79

Figure 19: Error Intermediate Event 79

Figure 20: BPMN's Exception Flow 80

Figure 21: BPMN implementation of SWF 80

Figure 22: Super Process Implementation in BPEL 86

Figure 23: Sub-Process Implementation in BPEL 86

Figure 24: ADT sub-Process and exception handling in BPEL 91

Figure 25: Roles and Relationships of SWF visualized by Pi4SOA 98

Figure 26: *RelationshipType* definition for SWF 99

Figure 27: Scenario description for part of SWF designed for WS-CDL modeling 101

Figure 28: Graphical Choreography Description for SWF generated in Pi4SOA 102

Figure 29: Combination of BPEL and WS-CDL 107

Table of Listings

Listing 1: Implementation of Super-process 87

Listing 2: Implementation of Sub-Process 88

Listing 3: Implementation of interfacing partnerlink 88

Listing 4: Message and port type declarations 89

Listing 5: Declarations of partners and partnerlinks 89

Listing 6: Central process 89

Listing 7: Selection of a WS-CDL description 103

Nomenclature

Acronym	Definition
ADT	Admit, Discharge & Transfer
B2B	Business-to-Business
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Modeling Notation
BPSS	Business Process Schema Specification
BWW	Bunge-Wand-Weber ontological framework
CRM	Customer Relationship Management
DICOM	Digital Imaging and Communications in Medicine
ebXML	Electronic Business using eXtensible Markup Language
ERP	Enterprise Resource Planning
HL7	Health Level Seven
HTTP	Hypertext Transfer Protocol
IHE	Integrating the Healthcare Enterprise

Infoway	Canada Health Infoway
IT	Information Technology
OASIS	Organization for the Advancement of Structured Information Standards
OS	Operating System
PDL	Process Definition Language
Pi4SOA	Pi Calculus for SOA
SWF	Scheduled Work Flow
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WS-CDL	Web Services Choreography Description Language
WS-Security	Web Services Security
WSCI	Web Services Choreography Interface
WSDL	Web Services Definition Language
WSFL	Web Service Flow Language
XLANG	Web Services for Business Process Design

XML	eXtensible Markup Language
XPath	XML Path Language
XPDL	XML Process Definition Language
XSD	XML Schema Document

Chapter 1: Introduction

1.1 Thesis motivation and problem statement

The complexity of business processes has increased the need for new approaches for understanding and representing them, and thus business process modeling has become one of the research areas which has gained the attention of researchers during the last decade. Moreover, the emergence of web services and service oriented concepts, as well as suggestions about using them on complex and collaborative systems [18], [42] has increased the researchers' attention towards modeling business processes within web service-based environments. Web services foster the integration of disparate systems and applications despite being developed at different times by different people. Efforts to standardize the composition and modeling of web services have resulted in the release of two major process-based service composition streams, orchestration and choreography, as well as various standards and languages, namely Process Definition Language (PDL), XML Process Definition Language (XPDL), Business Process Schema Specification (BPSS), Business Process Modeling Language (BPML), Web Services Choreography Interface (WSCI), Electronic Business using eXtensible Markup Language (ebXML), Business Process Modeling Notation (BPMN), Web Services Choreography Description Language (WS-CDL), and Business Process Execution Language for Web Services (WS-BPEL).

Modeling languages have differences with regard to their flexibility, expressiveness, adaptability, dynamism, and complexity. Moreover, due to web services' wide range of

applications, most of the aforementioned modeling languages are individually focused on a few areas of web service modeling. Being focused on a specific area makes the language more limited, and by comparing the languages several overlapping features can be found.

On the other hand, various business sectors are different in their adoption of technology. Advantages of web services, such as easy integration, more privacy, standardization, platform and language independency, simplicity, reusability, cost efficiency, and versatility, add value to businesses. At the present time, due to the wide variety of advantages that web services offer, many business sectors implement some aspects of service-oriented architecture, expose their functionalities as web services, or use existing web services provided by service suppliers; however, because of various applications of web services in different business sectors, web service modeling and implementation face a broad range of challenges and requirements. Studies on ontological constructs [22], workflow systems, document flows, communication flows, and data flow patterns [58] - [60], [62] - [64], [69], [70] showed that some web service modeling standards have overlapping elements and constructs in terms of expressive and/or functional elements. Moreover, the same studies claim that some modeling standards are unable to represent a number of patterns and/or ontological concepts, or they represent some patterns complexly. Consequently, the increase in the number of modeling standards, with specific features, and the diversity of business sectors adopting service oriented architecture, with their special requirements and challenges, has caused modeling to become a very complex task. Therefore, the question of identifying suitable modeling languages for specific business sectors has arisen. Many frameworks [21] - [23], [35], [64], [66] have been proposed for evaluating the suitability of a process modeling language for a given business sector; however, most of the frameworks only focus on one or a few aspects of modeling languages, and evaluation results usually present overlaps, limitations, inconsistencies, and ambiguities. These limitations are particularly

observable when modeling complex systems such as healthcare. Healthcare consists of a group of the most complicated processes because of the integration of workflows, data transactions and collaboration between different units. Also, the existence of different medical data transaction and process execution standards (e.g., Health level Seven (HL7) and Digital Imaging and Communications in Medicine (DICOM)) further adds to the complexity. Moreover, the increasing number of medical disciplines and the dynamic nature of healthcare delivery calls for dynamic process models [1], [2], [5].

Considering the aforementioned challenges, existing works on business process modeling do not completely satisfy the requirements of the healthcare sector. Healthcare's dynamicity and its need for security modeling have been neglected in most researches. Although there are researches that focused on security [25], [51] and dynamicity [32], the other important common factors do not appear in these researches. Moreover, investigations on modeling language simplicity and understandability [37], [52], [54] focused only on a certain group of actors while healthcare actors are different in terms of IT and modeling knowledge. Therefore, there is a lack of comprehensive analysis in the healthcare modeling area considering all its modeling requirements.

A process model is not only a graphical representation, but should also serve as a base for communicating domain details between stakeholders, and for communicating domain details to system designers [21]. Moreover, according to [33], models should be representative, easily understood, easy to use, optimized in the level of details, and support abstraction. The ability to use a process model for communication becomes difficult if modeling specialists are the only people who understand the models. Therefore, models should be representative enough to be understood by all model users such as healthcare administrative staff or clinical stakeholders. A further issue is integration. If each unit of a healthcare organization uses a modeling language that best fits its requirements, the entire organization's model will be a combination of several standards making the integration and comprehensiveness of these models difficult.

The aforementioned reasons have made modeling healthcare processes one of the researchers' interests over the last decade [18], [33], [42]. Modeling healthcare business processes in a web service-based environment using various modeling standards is a challenging research area question, and the lack of focus on web service-based environments in healthcare research, as well as the focus on conventional business processes add to the challenge. Furthermore, the selection of web service-based business process modeling standards focusing on healthcare was not based on pre-identified and established criteria for selecting a modeling standard. As mentioned earlier, the criteria for selecting a modeling standard highly depend on the existing standards' ability to overcome challenges and satisfy requirements of business processes. Moreover, the underlying models should be easily understood by designers and developers, a requirement that can be met through the ability of the modeling standard to accommodate and integrate documentation in the model. Another important aspect is the range and type of expressive elements and constructs existing in the language. The modeling standard should be easy to use and contain all mentioned features.

1.2 Contribution

The ultimate goal of this research is to identify the suitability of modeling languages to meet the modeling requirements of healthcare systems, which are representative of complex systems. In order to achieve that objective it is necessary to:

1. Outline the challenges and requirements of modeling complex healthcare processes in a service-based environment.

2. Identify language features contributing to overcoming the identified challenges or satisfying the identified requirements.

3. Investigate existing modeling languages which meet the identified requirements.

4. Suggest solutions to address the shortcomings of modeling languages for modeling complex healthcare systems

This research tries to answer the following questions:

- What are the challenges associated with modeling healthcare's business processes?
- What features should the modeling languages for modeling healthcare business processes have?
- Which combination of modeling standards can represent modeling requirements for web service-based healthcare business processes, identified in the first question, more completely, and without ambiguity, as defined before in this thesis?

1.3 Methodology

In order to answer the questions above, candidate business processes will be extracted from standard healthcare business process frameworks, a modeling process will be applied on them, and their conventional processes, as documented in frameworks, will be exposed as services. Furthermore, since the validation process of this research would be the composition of a small healthcare web service-based information system in order to measure the success of models, some implementation requirements, such as security, transaction, coordination, infrastructure, standards of healthcare systems (HL7 and DICOM), etc. will be briefly discussed.

We used an incremental and spiral method based on experiences on different languages in each phase. The data on healthcare processes is gathered from previous studies and healthcare process documentations on the subject of business process modeling, healthcare system modeling, and studies on comparison of web services modeling

standards. Furthermore, modeling some scenarios using different standards (as the experimentation part of the research) incrementally adds to our gathered data. Scenarios are selected based on complexity and are representative of healthcare modeling challenges. However, the main selection criterion is still the ability of the business process to be implemented in a web service-based environment.

For the evaluation of business process modeling languages, two methods are widely used. For those metrics that can be evaluated by studies on the specifications of modeling languages, we used theoretical approaches without implementation. In this study, the specification, notations and grammar of modeling languages are investigated based on defined metrics, and then a comprehensive conceptual comparison of modeling languages is feasible.

The second method is an empirical study. This method is utilized for those metrics that cannot be measured using theoretical approaches, for instance the cost of modeling using modeling languages can be evaluated by modeling a series of identical business processes using target languages by the same modeling experts, so the cost of modeling in terms of time, effort, model size, and other metrics can be feasibly evaluated.

1.4 Thesis Organization

This thesis is organized as follows: after defining our concepts in Chapter 2, this thesis continues with a brief discussion of similar work in Chapter 3. Chapter 4 discusses the requirements for modeling healthcare processes. In Chapter 5, based on the identified requirements, we introduce a framework for evaluating business process modeling languages in healthcare. In Chapter 6, we use the framework to conduct a comprehensive evaluation of three leading modeling languages, namely BPEL, BPMN and WS-CDL. Chapter 7 investigates using a combination of modeling languages to the address deficiencies of a single language.Concluding remarks are presented in Chapter 8.

Chapter 2: Healthcare Web Service Business Process Modeling Concepts

2.1 Business Processes

Capgemini [14] defines a process as *"A specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and output ".* Although this definition refers to some functionality of business sectors, the common understanding about processes includes almost all business activities; however, the only important processes are those that add value to offered products and services, so, most commonly, these processes are subject to modeling due to their competitive advantages.

Based on what is mentioned about processes, business process can be defined as *"the structure by which the organization physically does what is necessary to produce value for its customers and are broadly defined across functions/departments"* [14]. In general, a business process is a timely order of activities that produces a specific output based on the given inputs. A business process follows a special goal and usually is conducted by integration of several activities and interaction of departments, even though it can be done by only one department and through one activity.

2.2 Service Oriented Architecture and Web Services

Service Oriented Architecture (SOA) is a layered software application integration architecture that packages systems' business process functionalities as a collection of

services, and is officially defined by OASIS [45] as *"a collection of best practices, principles and patterns related to service-aware, enterprise-level, distributed computing"*. SOA describes how two computing entities interact in a way that enables one entity to perform a task on behalf of another entity. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. The standardization on SOA affects translation coordination, orchestration, collaboration, loose coupling, business process modeling, and other concepts that support agile computing [9]. The SOA architecture and its corresponding integration protocols and methods are represented in Figure 1.

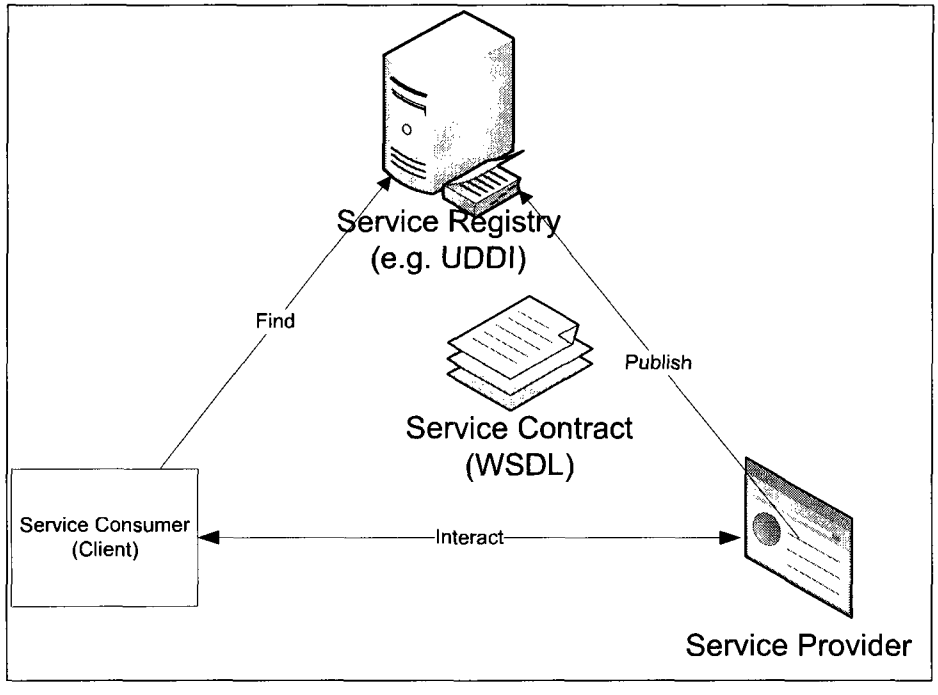


Figure 1: Service Oriented Architecture, from W3C [72]

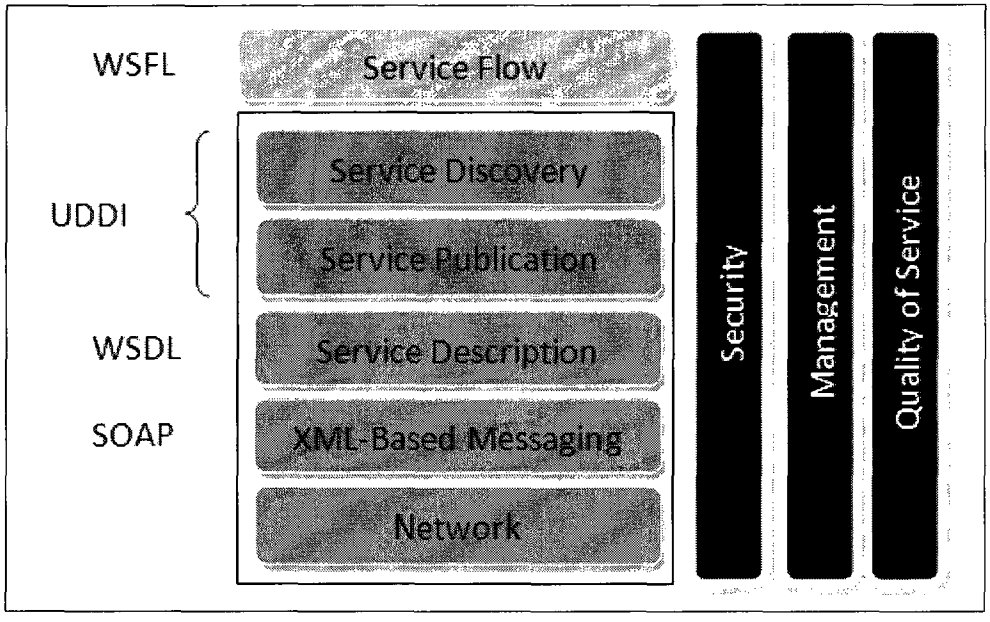


Figure 2: Web Services Stack, from Bellwood [10]

A service *"is a function that is well-defined, self-contained, and does not depend on the context or state of other services"* [9]. Web services are platform neutral interfaced software components or applications that interact using pervasive standard-based web technologies including open XML and Internet technologies [40]. Web services make the communication between applications easier and faster and have a wide range of use from a simple bank account transaction to customer relationship management (CRM) and enterprise resource planning (ERP) systems. Open standards for transactions and communication (SOAP, WSDL and XML) created hardware, operating systems (OS), and programming language independent environments for communication, and web services registries make the service available and easy to access on the web.

Web services are powered by XML and three other core technologies, WSDL, SOAP, and UDDI, which are best known as the main elements of the web service stack (Figure 2).

WSDL documents describe the service's location on the Web and the functionality the service provides. Information about the service is entered in a UDDI registry, which allows Web service consumers to search for and locate the services they need. Based on information available in the UDDI registry, the Web services client uses instructions in the WSDL to construct SOAP messages for exchanging data with the service over HTTP. The Web service stack is further detailed below.

2.2.1 XML (eXtensible Markup Language)

XML is a W3C (World Wide Web Consortium) [72] specification that defines a meta-language for data description. XML describes data with customizable, text-based tags that give information about the data itself as well as its hierarchical structure. Hence, XML is human readable and also platform independent. The advantages of XML made it a widely accepted standard for interaction between heterogeneous applications and platforms. XML is the basis of web services technology and its enabler technologies, i.e. UDDI, WS-CDL, and SOAP.

2.2.2 WSDL (Web Services Description Language)

WSDL is an XML-based format for describing web services. WSDL defines services as collections of network endpoints, or ports, using an XML format for documents. Clients wishing to access a Web service can read and interpret its WSDL file to learn about the location of the service and its available operations, which are defined as port types. A WSDL document acts as an interface to the actual implemented web service. There are six main elements existing in a WSDL file:

- **Port type:** an abstract collection of operation descriptions performed by a service through its defined interface.

- **Port:** specifies an address for a binding port defined by associating a network address with a reusable binding. Since ports are separated from their concrete instances, web services are highly reusable both in definition and utilization.

- **Messages:** abstract descriptions of the data being exchanged, and describe the names and format of the messages supported by the service.

- **Types:** data types definitions for variables used in the messages sent and received by the web service.

- **Binding:** definition of the communication protocols that web service operations support.

- **Service:** specifies the URL address where the service is available to access.

The concrete protocol and data format specifications for a particular port type constitute a reusable binding, where the operations and messages are then bound to a concrete network protocol and message format. In this way, WSDL describes the public interface of the web service.

2.2.3 SOAP (Simple Object Access Protocol)

SOAP is an XML-based enabler technology for web services developed by W3C that provides a standard data exchange format over HTTP between service provider and consumer. Due to its XML base and its transference over HTTP, SOAP messages can be sent between applications regardless of their platform or programming language. Most of interoperability features of web services exist because of interoperable messaging format supported by SOAP, so, as an important part of web service stack, SOAP has to provide security, management, and quality of service in its descriptions. To be able to provide these features, SOAP messages contain the following elements:

- **Envelope:** Encloses XML document as a SOAP message.
- **Header (optional):** Provides information about the message. SOAP headers become mandatory when the security, management, and quality of services are aimed to be provided for or through SOAP messages.
- **Body:** Includes the message payload.
- **Fault (optional):** carries information about a client or server error within a SOAP message.

The format of specific SOAP messages relevant to each service is defined in a WSDL file. The clients use the format specified in the WSDL file to create request and response SOAP messages and exchange data between the client(s) and the web services. Because both clients and server adhere to the WSDL contract while creating and interpreting SOAP messages, the messages are guaranteed to be compatible.

2.2.4 UDDI (Universal Description Discovery and Integration)

Being sponsored by OASIS (Organization for the Advancement of Structured Information Standards) [45], UDDI is a specification for creating an XML-based registry, and is often described as a 'Yellow Pages' for web services. It provides a standards-based set of specifications for service description and discovery, as well as a set of Internet-based implementations.

UDDI addresses a number of business problems. First, it broadens and simplifies business-to-business (B2B) interactions. UDDI helps businesses discover service providers and create relationships with them, regardless of their utilized technology and platform, using interfaces provided in UDDI.

UDDI also eases the dynamic discovery of services, and integration of relevant Web services into an aggregate business process. For instance UDDI provides instant connections to industry service providers for the businesses that do not have partnership and dynamically selects their service providers based on different conditions and prices.

2.2.5 Web services benefits

Web services provide several technological and business benefits, a few of which include application and data integration, versatility, code re-use, and cost savings. Web services provide inherent interoperability, which comes with the use of vendor, platform, and language independent XML technologies and the ubiquitous HTTP as a transport mean, so any application can communicate with any other application using Web services. In web service client and server interaction, the client only requires the WSDL definition to identify the right SOAP message formats, and neither part needs to know how the other is implemented or in what format its underlying data is stored.

Meanwhile, web services are designed to be accessible by humans in addition to client applications. Data gathered through web services can be combined to perform complicated applications. Hence, web services can be reused without considerations regarding the language that they are programmed in, and even complicated Enterprise Resource Planning (ERP) systems can be developed using web services.

All mentioned benefits add up to significant cost savings. Web services' ease of interoperability, their basis on open standards, and also their lack of requirement for additional technologies and infrastructure reduce development costs of highly customized applications for data integration, which can be expensive. Moreover, existing investments in systems development and infrastructure can be reused easily and combined to add value.

2.2.6 Web services development challenges

With the numerous advantages of Web services come a few challenges. Although web services are designed to simplify the application interaction and development, their development and implementation are costly. WSDL syntax becomes complicated quickly, especially when building a service with multiple operations in a text-based editor. Moreover, readability and understandability of complicated web service documents are very low, so without detailed model development, understanding, and maintenance of web services are difficult.

2.3 Healthcare business process frameworks

Several frameworks have been defined to extend the enterprise application integration to a level of scenario-based interaction. Business process frameworks document all business processes including workflows, document flows, and data transmissions in each of the scenarios happening in the specific business sector. Frameworks define usage scenarios aiming to integrate products, which conform to the framework, seamlessly.

Frameworks model the business process problems and possible solutions to the problems. They also define detailed transactions to support these IHE profiles using current established standards to solve the business problems [28].

Several frameworks have been defined for healthcare business processes including IHE [28] and Infoway [13]. These frameworks documented and modeled business processes in a conventional environment. Conforming to frameworks brings some benefits to systems. Because business processes are defined and documented in the frameworks, and they conform to healthcare communication standards (HL7 [26] and DICOM [43]), and information systems have been developed based on these frameworks, the communication and interaction of systems is more seamless and the number of medical errors, which result from lack of interaction, will decrease. Moreover, implementation

costs will decrease and workflows will perform more efficiently. Better informed medical decisions and faster results bring satisfaction to both patients and physicians.

The development of frameworks and the introduction of standards to the healthcare environments lead to a better representation of business processes, which improves the way they can be implemented. Newly developed healthcare business processes form on top of these frameworks and use existing standards for communication, so healthcare needs models developed based on these standards and frameworks. All the models developed in this thesis are built using these standards and are designed to represent business processes documented in these standards.

2.4 Business Process Modeling

New requirements force companies to develop new applications. A business process model is *"[a] mechanism for describing and communicating the current or intended future state of a business process* [33]." As it is mentioned before, the requirements of the company are the main reason that companies develop business processes, so business process modeling starts with identifying requirements. Requirement analyses can define business processes' goals. The business process modeling starts after identifying goals and requirements.

There are two types of models: descriptive and normative. Descriptive models do not propose any limitations on the mapping of concepts to the domain of investigation. Descriptive models attempt to unearth, and perhaps explain, the actual state of the object at the time of its inspection [55]. Consequently, these models are not trying to change the state of a system, so they are not easily improvable and changeable. The findings of a descriptive model intend to enlarge our knowledge about the object, so they shall not be used in practice, at least immediately. On the other hand, the normative models, which are extensively used by designers, intend to discover ways to improve the object or

similar latter objects, by pointing out possible improvements for the object of study [55]. They follow a specific set of rules and use some specific constructs that make them more incremental than the former ones, so they are more often used than descriptive models for systems needing evaluation and improvement [33]. It is useful to mention that these models can be converted to each other interchangeably.

The reasons that make companies develop process models before developing any software are listed as follows. First, modeling can create a representation of a business process in an understandable way. Second, modeling makes the system more abstract and the analysts can focus on the large picture of the tasks instead of details about activities, and present the steps of a business process. Furthermore, the models could be used in order to train staff about the current business processes of the company. Models can also be the basis for designing and developing new software and applications or improving existing ones. Most business processes can become obsolete or very complicated over time, but, if the models are available, process improvement, and consequently improvement of organizational structure of the company, will be more feasible. Moreover, the models can be the basis for implementing innovative ideas, in that the model can show the feasibility of implementing new business processes. The models also define interactions outside of the boundaries of the company either for receiving input or sending output. Finally, a business process model can be the basis for other models like interaction and software models [27], [33].

The result of modeling can be summarized in improvement of the business, easier change management, faster development, and easier identification of errors and process deficiencies. After all this, we will have an effective and efficient process which is the main goal of modeling.

2.5 BPMN

Business Process Modeling Notation (BPMN) is a graphical workflow modeling language that does not focus on any technology; it is able to model processes focusing on service oriented architectures. BPMN was first introduced by the Business Process Management Initiative (BPMI) [12] and accepted as a standard in May 2004. The primary goal for the development of BPMN was to provide an easily understandable notation that can readily be useful to all business actors [68]. BPMN also provides features to transform BPMN graphical models into executable BPEL processes using its internal model.

BPMN is composed of a set of basic elements: flow objects, connecting objects, swim-lanes, and artifacts. Within each basic category of elements, additional variation and information can be added to extend the language and the model itself, in order to support the requirements for complexity without imposing any dramatic change to the basic look of the diagram.

BPMN is designed to cover many types of modeling and allows the creation of end-to-end business processes, and it is used to communicate a wide variety of information to a wide range of audiences. BPMN has a set of structural elements that will help the viewer to easily differentiate between separate sections of a BPMN diagram.

Although BPMN is a strong language in terms of representing private, abstract, and collaborative business processes, it still has drawbacks in some areas. BPMN is mainly intended to reduce the complexity and increase the understandability of models, but still there is some ambiguity in BPMN models. BPMN is also incapable of representing routine work and knowledge work. BPMN only captures very simple routine work for example data entry and structured decision making. When it comes to human knowledge work, or data transmission, a BPMN diagram simply does not contain the right information. BPMN is partially considered as a software design notation. Hence, even if

business people can be persuaded to try and understand BPMN notation, which is full of engineering symbols, it is the wrong tool to capture collaborative, adaptive, innovative human activity. However, modelers can develop syntactically and semantically correct models using BPMN that could never be implemented using the existing programming languages. Effectively, BPMN is a high-level software design notation.

2.6 Choreography and WS-CDL

Choreography defines collaboration rules and interactions between two or more business entities or processes. It describes all participants' observable collaborative behavior from the perspective of all involved parties, and it enables the use of web services in a collaborative manner by providing the means for describing the behavioral interface of a single web service, and for specifying interaction protocols for web service collaborations [6] - [8].

WS-CDL is an XML-based language for describing service interactions in a peer-to-peer manner in which each involved entity remains autonomous and where there is no coordinating web service or any coordinated one. It is used to describe the collaborative observable behavior of multiple interacting service-based entities that pursue a common goal. Interactions between entities are documented and described as a set of message exchanges which form a contract between entities and set a basis for their further interactions. Since WS-CDL is not explicitly bound to WSDL, it is possible to use WS-CDL for describing collaboration protocols for both services that have no WSDL descriptions and those that have WSDL descriptions [38], [49].

According to W3C [72], primary goals for developing WS-CDL are to permit:

- **Reusability**. The same Choreography definition is usable by different parties operating in different contexts with different software, programming languages, and/or platforms

- **Cooperation**. Choreographies define how different independent parties should cooperate by specifying the sequence of message exchanges between participating parties.

- **Multi-Party Collaboration**. Several parties and processes can participate in a choreography, and the number of participating parties are not a point of concern.

- **Semantics**. Choreographies can include human-readable documentation and they can also embed semantics into the description and documentation for all the components in the choreography.

- **Composability**. New choreographies can be created by reusing and combining existing choreographies.

- **Modularity**. Probably one of the most important advantages of choreography is its modularity. Choreographies can be compiled from parts involved in several different choreographies.

- **Information Driven Collaboration**. Choreographies describe how parties make progress within a collaboration through recording of exchanged information.

- **Information Alignment**. Observable information of involved parties in choreographies can be communicated and synchronized.

- **Exception Handling**. Exceptional situations in the performance of choreography message exchanges can be defined and documented, and also choreographies can define how those exceptions are handled.

- **Transactionality**. Although the behavior of each involved party in choreography is non-observable to other parties, they can coordinate long-lived collaborations.

- **Specification Composability**. Choreographies can include functionalities of different standards into their definitions. Thus, standards such WS-Security can be applied to add security features to choreographies.

2.7 Orchestration and BPEL

Orchestration, one of two main service composition methodologies, is seen through the Business Process Execution Language (WS-BPEL or BPEL). In the context of orchestration, a central process takes control of the involved web services providing a workflow style composition of services and coordinating the execution of operations on the involved web services. Orchestration imposes a level of abstraction on all involved web services, except the process coordinator as they do not need to know that they are involved in a composition process, and they only focus on incoming and outgoing messages [36], [37]. BPEL is a language for specifying and executing service orchestration processes against a domain of web services. It is built on Web Services Flow Language (WSFL) and Web Services for Business Process Design (XLANG), and inherits its ancestors' graphical and text based notations. Genuine BPEL is notably used for creating executable processes, while it is able to model abstract processes forgoing execution [15].

There are eight different goals for developing BPEL:

- Defining business processes that are able to interact with external entities using Web Services operations.

- Defining business processes using an XML-based language that does not impose any particular technology or methodology

- Defining a set of Web service orchestration concepts that are meant to be used by both the external and internal views of a business process.

- Providing both hierarchical and graph-like controls.
- Providing data manipulation functions for the simple manipulation of data needed to define process data and control flow.
- Supporting the basic and advanced lifecycle mechanism for processes, including implicit creation and termination of processes in addition to suspend and resume.
- Defining a long-running transaction model with the support for compensation, scoping, and failure recovery actions.
- Decomposing and assembling processes using Web Services based models.

Although there are some similarities among goals for creation of BPEL and WS-CDL, these two modeling languages have some critical differences:

- WS-CDL provides a definition of the information formats being exchanged by all participants, while BPEL observes the system from a single process viewpoint, and provides the information formats exchanged by one participant.
- In WS-CDL the processes are defined by the functionality of each participant in the choreography, so each party defines the next steps in the process by selecting the messages that should be exchanged, whereas in BPEL the process is predefined in the central orchestrator that provides process rules for the execution, so the next steps are inferred.
- WS-CDL specifies a common understanding of a message exchange through a model, while BPEL has no such concept.

2.8 Workflow, Data flow, and Interaction patterns

Research done by Van Der Aalst et al. [71] resulted in the identification of workflow, data flow, and interaction patterns, to describe the potential capabilities that a workflow server may have during the performance of business processes. Workflow patterns cover the behaviors that can be captured within most business process models.

2.8.1 Basic Control Patterns

Basic control patterns define the basic functionalities and modeling patterns of business processes. The models represented here are developed by Yet Another Workflow Language (YAWL) [61] proposed by Van Der Aalst and his team to represent all patterns in a workflow language.

Sequence

The sequence pattern defines the basic sequence flow of activities in an ordered manner, with the result that activities start after completion of the previous activities. (Figure 3)

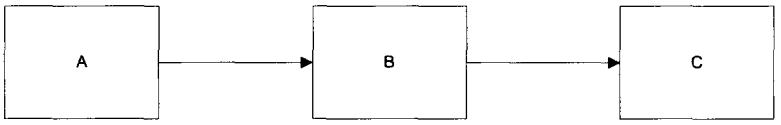


Figure 3: Sequence Flow

Parallel Split

The parallel Split pattern defines the mechanism for simultaneous performance of activities. Thus, the activities will start concurrently, and the process will not continue unless both activities are completed. (Figure 4)

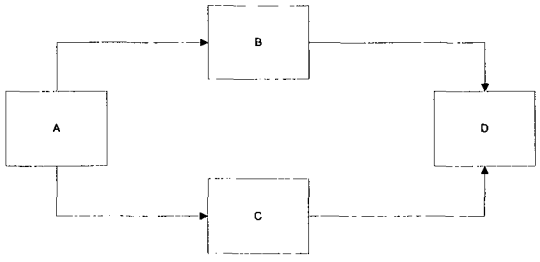


Figure 4: Parallel Split and Synchronization

Synchronization

After completing the activities in the split, the path should continue. The Synchronization pattern makes sure that the next activity will not start before the completion of both split activities. (Figure 4)

Exclusive Choice

Exclusive choice pattern is similar to the split pattern, but only one activity will be executed at a time, and other activities will be ignored in the continuation of the process. In other words, only one of the alternative paths may be chosen for the process to continue. (Figure 5)

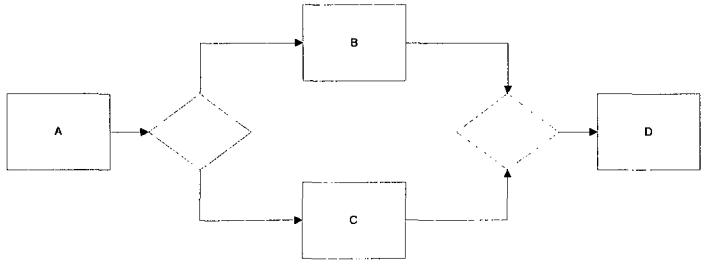


Figure 5: Exclusive Choice and Simple Merge

Simple Merge

The simple merge pattern is defined to explain the join mechanism of the activity paths in processes. The simple merge will follow the exclusive choice pattern. (Figure 5)

2.8.2 Advanced branching and Synchronizing Pattern

Multiple Choice

The exclusive choice pattern only allows for one activity to be performed while in the multiple choice pattern, more than an activity can be chosen to be performed during the process. Based on its definition, the multiple choice pattern should even allow zero activities to be chosen. (Figure 6)

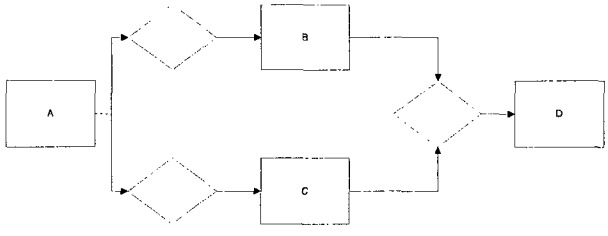


Figure 6: Multiple Choice and Multiple Merge

Multiple Merge

The multiple merge pattern represents the location that the flow of process merges after passing the multiple choice. Unlike the simple merge pattern, the activities in each selected path continue independently. (Figure 6)

Discriminator

The discriminator pattern is another way of combining the paths that were generated from a Parallel Split pattern. It differs from the synchronization pattern in that the first completed activity will pass the discrimination point and other remaining activities will be blocked and ignored even if they complete their functionalities.

N out of M Join

The N out of M join pattern is a combination of features that are provided by the synchronization and discriminator patterns. N out of M Join defines N of the M processes that should continue in a synchronized manner while others (M - N) will be blocked like in the functionality of discriminator pattern.

Synchronizing Merge

This pattern is similar to the synchronization pattern with unknown number of entering complete activities. The synchronizing merge pattern should somehow identify how many complete activities will enter the merge point.

2.8.3 Structural Patterns

Arbitrary Cycles

The arbitrary cycle pattern is a loop allowing multiple entry and exit points. This pattern is important for the visualization of valid, but complex, looping situations in a single diagram. (Figure 7)

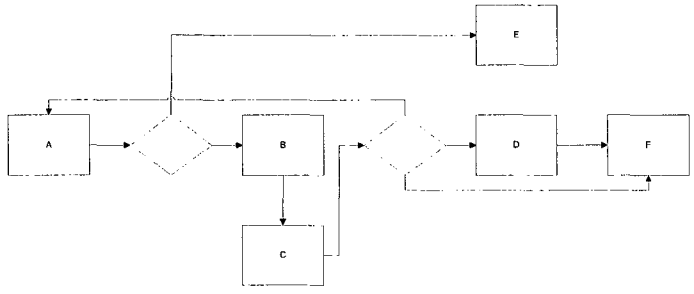


Figure 7: Arbitrary Cycle

Implicit Termination

The implicit termination pattern allows a specific path in a process to terminate whilst other paths are still processing (Figure 8).

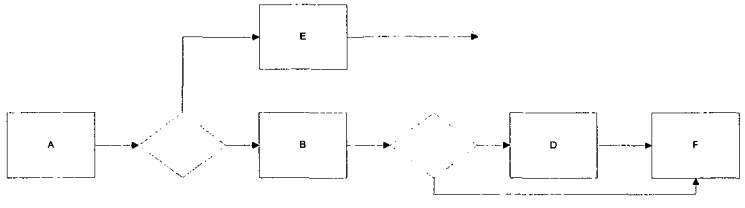


Figure 8: Implicit Termination

2.8.4 Patterns Involving Multiple Instances (MI)

MI with a Priory Design Time Knowledge

This pattern describes how an activity can be instantiated a known number of times, and in parallel. That is, the modeler knows how many times the activity should be performed and defines that number in the process model.

MI with a Priori Runtime Knowledge

This pattern is similar to the MI with a priori design time knowledge pattern, except that the number of copies is not known until the process is being performed and it cannot be set ahead of time. In addition, this pattern allows that the copies be performed sequentially as well as in parallel.

MI with No Priori Knowledge

This pattern differs from the previous two patterns in that the number of copies of an activity cannot be determined before the copies are created. The exact number is actually determined during the performance of those copies. This means that a standard looping or multi-instance pattern will not be sufficient to create this behavior.

MI Requiring Synchronization

This pattern is similar to the MI with a Priori Runtime Knowledge pattern except that it requires that all the copies of the repeated activity (performed in parallel) be completed before the process continues.

2.8.5 State-Based patterns

Deferred Choice

This pattern represents a type of exclusive decision, similar to the Exclusive Choice pattern. However, the basis for determining the path that will be taken is different. The Exclusive Choice pattern is based on the evaluation of process data while the Deferred Choice pattern is based on an event that occurs during the process.

Interleaved Parallel Pattern

The use of the word "parallel" in the name of this pattern can be a bit deceptive since the activities in the pattern must be performed sequentially, but in no specific order. The sequential performance of the activities is often due to the requirement that the activities share or update the same resources. The performers of the activities will decide the order of the activities.

Milestone

There are points within a process where it is important to know whether a specific event occurred or a condition has been met. These events or conditions can be referred to as milestones.

2.8.6 Cancellation Patterns

Cancel Activity

This pattern describes how it is possible for there to be two competing activities. When one of the activities completes, it signals that the other activity should stop processing.

Cancel Case

This pattern is an extension of the Cancel Activity pattern. In this pattern, however, an entire process is cancelled.

2.8.7 New Control Flow Patterns

Structured Loop

The ability to execute an activity or sub-process repeatedly, using pre-test or post-test condition associated with it. The looping structure has a single entry and exit point.

Recursion

The ability of an activity to invoke itself during its execution or an ancestor in terms of the overall decomposition structure with which it is associated.

Transit Trigger

The ability for an activity to be triggered by a signal from another part of the process or from the external environment.

Persistent Trigger

Similar to Transit trigger, but triggers, here, are persistent in form and are retained by the workflow until they can be acted on by the receiving activity.

Cancel Region

The ability to disable a set of activities in a process instance even though there may be activities that are already executing, so then they will be withdrawn.

Cancel Multiple Instance Activity

If the number of created activities within a given process instance is known at the design time, at any time, the multiple instance activity can be cancelled and any instances which have not completed are withdrawn.

Complete Multiple Instant Activity

If the number of created activities within a given process instance is known at the design time, these instants can be completed before any other activities created.

Blocking Discriminator

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The discriminator construct resets when all active incoming branches have been enabled.

Cancelling Discriminator

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled, and the execution of the rest of other branches will be cancelled.

Structured Partial Join

The convergence of M branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Subsequent enablement of incoming branches does not result in the thread of control being passed on.

Blocking Partial Join

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. The join construct resets when all active incoming branches have been enabled once for the same process instance.

Cancelling Partial Join

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Triggering the join also cancels the execution of all of the other incoming branches and resets the construct.

Generalized AND-Join

The convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled.

Static Partial Join for Multiple Instant

Whenever the number of instants within a given process instance is known at the design time, multiple concurrent instances of an activity can be created. Once N of the activity instances have completed, the next activity in the process is triggered.

Cancelling Partial Join For Multiple instants

Whenever the number of instants within a given process instance is known when the first activity instance commences, multiple concurrent instances of an activity can be created. Once N of the activity instances have completed, the next activity in the process is triggered and the remaining M-N instances are cancelled.

Cancelling Partial Join For Multiple instants

Whenever the number of instants within a given process instance is known at runtime, multiple concurrent instances of an activity can be created. At any time, whilst instances are running, it is possible for additional instances to be initiated.

Acyclic Synchronizing Merge

The convergence of two or more branches which diverged earlier in the process into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled. Determination of how many branches require synchronization is made on the basis of information locally available to the merge construct.

General Synchronizing Merge

The convergence of two or more branches which diverged earlier in the process into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled or it is not possible that the branch will be enabled at any future time.

Critical Section

Two or more connected sub-graphs of a process model are identified as "critical sections". At runtime for a given process instance, only activities in one of these "critical sections" can be active at any given time.

Interleaved Routing

Each member of a set of activities must be executed once. They can be executed in any order but no two activities can be executed at the same time. Once

all of the activities have completed, the next activity in the process can be initiated.

Thread Merge

At a given point in a process, a nominated number of execution threads in a single branch of the same process instance should be merged together into a single thread of execution.

Thread Spilt

At a given point in a process, a nominated number of execution threads can be initiated in a single branch of the same process instance.

Explicit Termination

A given process (or sub-process) instance should terminate when it reaches a nominated state. Typically this is denoted by a specific end node. When this end node is reached, any remaining work in the process instance is cancelled and the overall process instance is recorded as having completed successfully.

2.9 Ontological Constructs

An evaluation method for language concepts is the use of ontological constructs. Ontology defines a set of conceptual constructs for a system or a technology. One of the most complete and useful ontological frameworks for evaluating modeling grammars is Bunge-Wand-Weber (BWW) ontological framework which is proposed and explained in detail in [65]. The authors argue that modeling notations which are not able to represent all of the ontological constructs are incomplete. Meanwhile, there are the same issues

with ambiguity that are described in the previous section. Thus, the authors categorized the possible issues with modeling languages in five groups, namely completeness, clarity, overload, redundancy, and excess.

Language incompleteness reduces understandability because incomplete language features and structures force the modeler to ignore some features of models as there is no construct supporting those specific needs. Nevertheless, while there may be other ways to represent the needed constructs in order to compensate the languages' incompleteness, these ways are diverse and may cause ambiguity.

The redundancy refers to a situation where two or more elements represent the same construct. Excess is the complete opposite, and it refers to the existence of a construct with no corresponding ontology; thus both redundancy and excess can be source of ambiguity.

Chapter 3: Related Work

Modeling languages have fundamental differences in regard to expressiveness, flexibility, adaptability, dynamism, and complexity aspects [34]. A consequence of this is that different classes of modeling languages suit different business sectors. Explorations on the common factors of modeling languages have concluded that the understandability and complexity of models have a positive relationship [24]. However, to date, focus has been on common features of all modeling languages while unique features have been neglected.

As Van Der Aalst et al. identified work flow and communication flow patterns, they developed a framework for analyzing the languages based on those patterns [71]. They evaluated different languages, standards, and commercial products for their compatibility with the pattern-based framework. The result of their research illustrated a lack of completeness in all evaluated languages and products. In addition to the inability of languages to model all patterns, almost all languages were unable to model some patterns in an understandable way, which caused ambiguity in the model development and use. Table 1 presents their research results for the different languages they examined. In the table the plus represents a pattern that can be presented by a language construct dedicated to it. Minus representative of a pattern that cannot be modeled n the language by any means. Plus/minus refers to a pattern that can be represented in the language by combining language constructs, but there is no constructs aimed for representing the pattern. In this table the difference between BPEL with XLANG and WSFL is shown.

BPEL is created and developed from a combination of these two languages. The table is important since it represents the improvements made to BPEL.

The research improves the base for identifying a suitable language for different business sectors, but it only evaluates the languages from one point of view. Hence, this research cannot solely provide a complete justification for deciding on the use of a language use.

The understandability of modeling languages is a fundamental requirement for increasing a modeling language's usability. Sedera et al. [56] proposed that the understandability of business process models has a great effect on modeling success. In terms of understandability, the outcome of a modeling language, (a business process model) is closely related to the business modeler's capabilities in applying modeling language features. Human factors are inevitable and quite dependant on the expertise level of the modeler, his creativity, and his familiarity with the business. On the other hand, there are some factors that affect the understandability of modeling languages. Understandability factors have a tight connection with other metrics used for the evaluation of modeling languages and models. Some understandability factors for business process models are mentioned in [36], [37], two of which can be used for modeling languages as well. Understandability factors are:

- **Separability:** degree of articulation points in models
- **Structuredness:** how far a model is built by nesting blocks of matching join and split routing elements
- **Graphical notations:** notation standards and graphics representing connectivity, and flow patterns

Van Der Aalst et al.'s paper did not suggest any comparison of modeling languages for their understandability.

Table 1: Comparison of modeling languages based on their ability to represent workflow and communication flow patterns, from Workflow Patterns Initiative [71]

Pattern	Products		
	BPEL	**XLANG**	**WSFL**
Sequence	+	+	+
Parallel Split	+	+	+
Synchronization	+	+	+
Exclusive Choice	+	+	+
Simple Merge	+	+	+
Multiple Choice	+	-	+
Synchronizing Merge	+	-	+
Multiple Merge	-	-	-
Structured Discriminator	-	-	-
Arbitrary Cycles	-	-	-
Implicit Termination	+	-	+
Multiple Instances without Synchronization	+	+	+
Multiple Instances with a Priori Design-Time Knowledge	+	+	+
Multiple Instances with a Priori Run-Time Knowledge	-	-	-
Multiple Instances without a Priori Run-Time Knowledge	-	-	-
Deferred Choice	+	+	-
Interleaved Parallel Routing	+/-	-	-
Milestone	-	-	-
Cancel Activity	+	+	+
Cancel Case	+	+	+

From another aspect, Green et al. [22], [23] provided a framework for the evaluation of languages based on their identified ontological constructs. Based on their analysis, modeling languages can have one of the following conditions against any specific construct:

Table 2: Ontology based expressiveness analysis of ebXML, BPML, BPEL4WS, and WSCI, from Green et al. [22], [23]

Ontological construct	ebXML BPSS construct	BPML construct	BPEL4WS construct	WSCI construct
Thing				
Property	✓	✓	✓	✓
Class	✓	✓	✓	✓
Kind				
State	✓	✓	✓	✓
Conceivable state space	✓			
State law	✓			
Event	✓	✓	✓	✓
Conceivable event space				
Lawful event space	✓			
Transformation	✓	✓	✓	✓
Lawful transformation	✓	✓	✓	✓
Lawful state space	✓			
History	✓			
Acts-on			✓	✓
Coupling	✓		✓	✓
System	✓		✓	✓
System composition			✓	✓
System environment				
System structure			✓	✓
Subsystem	✓			
System decomposition				
Level structure				
External event	✓	✓	✓	✓
Stable state	✓			
Unstable state	✓			
Internal event	✓	✓	✓	✓
Well-defined event	✓	✓	✓	✓
Poorly defined event	✓	✓	✓	✓

- Ontologically incomplete (or construct deficit): there is not at least one grammatical construct for each ontological construct.

- Ontologically complete: there is at least one grammatical construct for each ontological construct. However, the ontologically complete language should be analyzed for ontological clarity in case one of the following conditions happens:

    - Construct overload: there is a grammatical construct that represents more than one ontological construct.

    - Construct redundancy: there is more than one grammatical construct representing the same ontological construct.

- Construct excess: there is a grammatical construct that does not map into any ontological construct.

Table 2 represents the analysis made by Green et al. about different modeling languages.

Green et al. [22], [23] recommended a combination of standards (Table 3) to overcome that limitation. That recommendation, however, was only based on a discussion on ontological constructs, while we believe that other measures such as workflows, document flows, data flows, understandability, portability, notations, and domain support should be considered as criteria for evaluating standards. They conducted an overlap analysis on support for ontological constructs in different modeling languages, and their combination. In each combination, they analyzed the total number of constructs and the number of overlapped constructs in addition to specifying the number of constructs that can be only implemented by one of the languages involved in the combination. Meanwhile, the combinations suggested in [22], [23] are not practically tested, yet they represent a strong theoretical background for further research. In addition to the limitations mentioned above, BPEL and some other composition languages lack support for certain web service composition requirements, for example BPEL cannot support attachments directly [1] - [5].

On the other hand, in healthcare process modeling research topics, most research initiatives have chosen BPEL as their modeling standard while neglecting other available modeling languages. However healthcare process modeling research does not cover all aspects of healthcare business processes, and, since it is limited to some complexity features (coordination, transaction, security) neglecting user understandability, language completeness and flexibility, it is not easy to claim whether or not BPEL can model and satisfy all requirements of web service enabled healthcare business processes. Moreover, there is no absolute solution offered for healthcare's modeling complexity. Moreover, as research by Wohed et al. [69], [70] and other similar studies identified that BPEL and all

other web service modeling standards are not completely able to model all the flows and constructs of a system's business processes, adding to the uncertainty.

Anzbock and Dustdar [1] - [5] have explored IHE workflows in order to model them in a web service based environment. As Van Der Aalst et al. [58] - [60], [62] - [64] discussed, an internet-based network infrastructure restricts inter-organizational workflow integration, since the integration should be considered in both intranet and internet environments. The diagram in Figure 9 illustrates the current healthcare environments where workflow items such as patient and data are being exchanged between the departments (e.g. Admit, Discharge and Transfer (ADT) and acquisition modality) and over networks (intra and inter organizational network gateways) respectively.

Table 3: Ontological overlap analysis of ebXML, BPML, BPEL4WS, and WSCI, from Green et al. [22], [23]

	ebXML BPSS (20 constructs)	BPML (10 constructs)	BPEL4WS (15 constructs)	WSCI (15 constructs)
ebXML BPSS (20 constructs)		10 overlap 20 constructs 10 ebXML BPSS 0 BPML	12 overlap 23 constructs 8 ebXML BPSS 3 BPEL4WS	12 overlap 23 constructs 8 ebXML BPSS 3 WSCI
BPML (10 constructs)	10 overlap 20 constructs 10 ebXML BPSS 0 BPML		10 overlap 15 constructs 0 BPML 5 BPEL4WS	10 overlap 15 constructs 0 BPML 5 WSCI
BPEL4WS (15 constructs)	12 overlap 23 constructs 8 ebXML BPSS 3 BPEL4WS	10 overlap 15 constructs 0 BPML 5 BPEL4WS		15 overlap 15 constructs 0 BPEL4WS 0 WSCI
WSCI (15 constructs)	12 overlap 23 constructs 8 ebXML BPSS 3 WSCI	10 overlap 15 constructs 0 BPML 5 WSCI	15 overlap 15 constructs 0 BPEL4WS 0 WSCI	

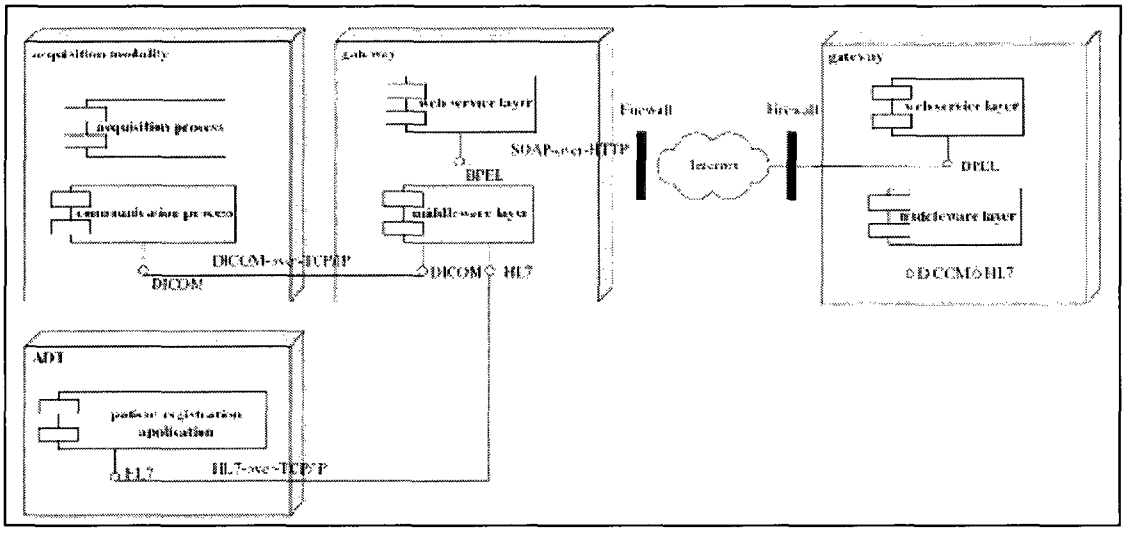


Figure 9: Internet/Intranet environment for medical web services, representing data and patient exchange, from Anzbock and Dustdar [2]

Anzbock and Dustdar also argued that in these systems, healthcare standards (HL7 and DICOM) are not communicated over the internet and they just convey data within the organizational intranet. The gateways mentioned in the diagram have two responsibilities including implementation of IHE conformant web service-based workflow model for medical web services and enabling internal nodes to participate in IHE conformant workflows. The web services in the latter model are responsible for communication between intranet and internet. They identified the essence of web service implementation and modeling in addition to considerations that should be done for modeling workflows as web services. They defined the following requirements in an implementation environment and provided solutions for them:

- Security and reliability
- Compatibility with healthcare data transmission standards
- Web service bindings

- Infrastructure requirements
- Protocols defined in healthcare business process frameworks

However, the explored requirements have been mainly developed for the implementation of environments while the requirements for the first step of system development (i.e. modeling) have been neglected.

Nystevold and Krogstie [44] developed a generic quality framework for assessing modeling languages for a specific case study on an insurance company. They evaluated the framework from an organizational behavior viewpoint. The focus of the research is not on a specific technology and, as well, they do not consider all aspects of modeling. Their method is more concerned with models in contrast with our method with a focus on modeling languages.

Other researchers [37], [54] have taken a similar approach on assessing model understandability, and developed frameworks to analyze models; however, Rossi and Turrini [54], in addition to research on the understandability of models, developed a method as a guideline for the development of future modeling languages. Their work is similar to our work in many aspects, but they do not provide a complete measurement base. Moreover, their work is a general analysis that does not consider the challenges and requirements caused by a specific business sector. Finally, they only considered common language features, although unique features will improve the modeling capability in many cases, especially in complex systems.

Chapter 4: Challenges of Healthcare Processes Modeling

Modeling complex processes has always been a challenge for modelers, and, as systems and models get more complex, the use of complicated models becomes challenging for model users. As an important factor in modeling, models should be easily understood by their target users in addition to containing an appropriate level of detail to fulfill their development purposes. Furthermore, depending on system requirements, completeness, extensibility, and the ability to represent security and exception handling features are important criteria. These requirements are more coherent in the context of modeling complex systems. Specific features of complex systems impose the existence of many previously mentioned features in their models. Not unlike other complex systems, healthcare systems have specific modeling requirements.

Referring to previous works on modeling web service-based healthcare processes, Anzbock and Dustdar [3] identified four major challenges of modeling medical web services:

- a high degree of vertical standardization through DICOM, HL7 and IHE
- currently implemented systems based on conventional middleware
- lack of inter-organizational workflow support as a common problem
- no current Web services-based approach which tries to fill this gap

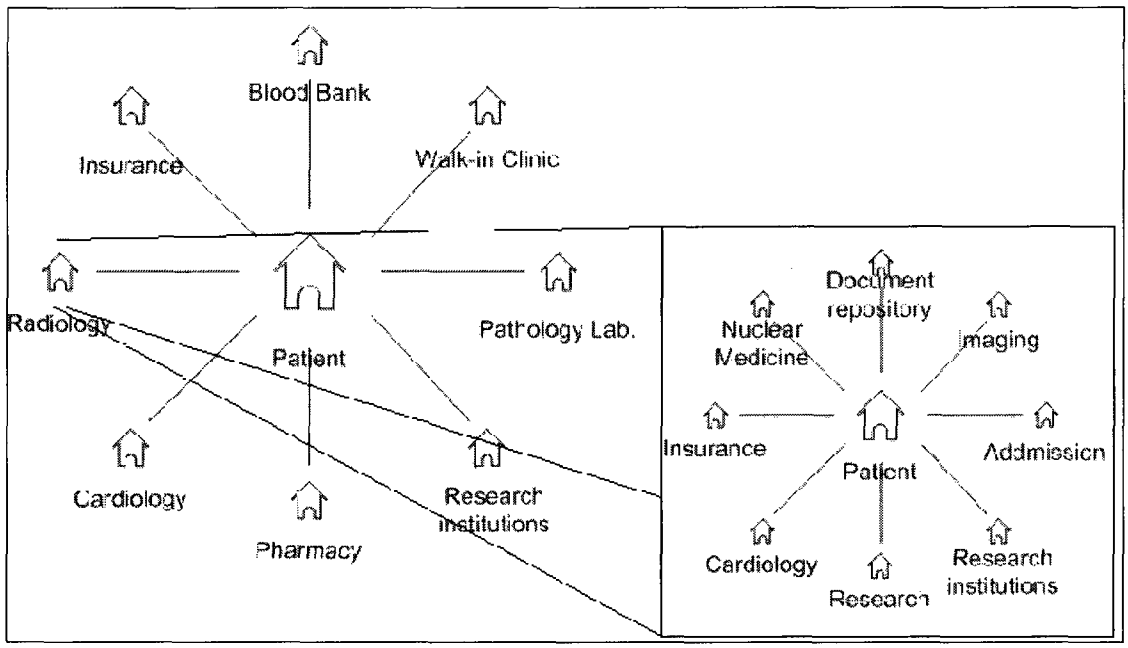


Figure 10: Healthcare Complexity

The authors argued that some challenges are caused by the immaturity of the web service stack standards and the existence of competing standards in that context. Anzbock and Dustdar argued that some challenges like what comes in the following paragraph have not been solved yet.

In the transfer and encryption of large binary data it is not clear, whether a Web service or an IHE based infrastructure should be preferred. Next, some of the standard specifications of the Web service stack are not yet widely implemented or, especially for coordination services, competing standards exist. Finally, there are problems normalizing activity diagrams resulting of ambiguities in the medical industry standards. These identified shortcomings are mainly discussed in the implementation phase, but the lack of integration, workflow support, and suitable system architecture imply design and modeling level deficiencies.

Plesk et al. [47], [48] and Fraser et al. [19] investigated healthcare processes not from a modeling perspective, yet their work is useful to identify challenges of healthcare processes and environments. Combing these challenges with the modeling deficiencies explained in [33] - [35], [39], [44], [46], [51], [52] lead us to identify challenges of healthcare process modeling more accurately. In this section we elaborate on these deficiencies as well as healthcare's modeling requirements and complexity, with special attention paid to features that should be included in modeling languages to satisfy those requirements.

4.1 Complexity of Processes

Modern health services involve medical care delivered by members of multiple groups with different knowledge levels. A simple healthcare process often involves a large number of caregivers, most likely residing in different physical locations, providing various sets of services. Containing complicated processes with high number of transactions, healthcare is considered a complex system.

Studies on process complexity discovered that two of the measures of process complexity are the number of units involved in the process and the number of transactions between the collaborating parts [24]. As we mentioned earlier, even simple healthcare processes are collaborations of several units with a large number of messages forwarded and received. For instance, the radiology subsystem in the IHE framework contains 34 units, more than 65 transactions, and each transaction contains several message types.

Meanwhile, in addition to inter-unit transactions, each healthcare unit consists of several separate departments that contribute to the accomplishment of each unit's tasks. The increased number of transactions and involved parties in a process make it more and more complex. Figure 10 represents the diversity and multiplicity of healthcare units and

their internal departments, in addition to complete transactional complexity for both inter-unit and intra-unit transactions.

Modeling such a complex system is not an easy task. Sparx Systems [57] defined seven specifications of a business process, four of them directly connected to organizational collaborations. Business process modeling languages have focused on representing collaborations as their main goal [16]. However, the results presented in [24] show that generated models for complex systems are still far from meeting expectations. The models are large, and unreadable in many cases. In addition, the models can be interpreted in many different ways since they become vaguer as model size increases. Moreover, some complex interactions cannot be modeled using current modeling languages as will be discussed in Chapters 5 and 6.

4.2 Security and Privacy

Increases in the process complexity, rises in the number of involved parties, and the involvement of several actors in a process increases the probability of security infringement and privacy-related issues during system development and use. This issue has a major importance in healthcare systems due to its interaction with other organizations such as insurance, financial and billing companies that need detailed information about patients. Most importantly, patient history and healthcare transactions should be private in nature. In order to be able to develop secure systems and work environments with enabled privacy on the transactions, the security requirements of the system should be represented in the models. Therefore, representing security features to determine the type of communication channel for interaction among a large number of contributing partners and privacy features for representing and classifying transmitted data for each partner are both important aspects of model development for healthcare systems. Security and privacy are particularly important when communication is

conducted over the Internet [48], such as within a service-oriented environment which gives access to the systems from anywhere on the network.

In addition to system development, models are also used for staff training and management decision making. Each of these mentioned uses of models has a need for in-detail representation of the system in an understandable way. Hence, security and privacy representation is an important required feature for models and modeling languages.

4.3 User Understandability

Modelers and system managers pursue a number of goals in process modeling, especially in complex processes like healthcare. Although models are being developed for documenting and describing systems, they should also enhance the stakeholders' general understanding of processes. Complex systems are usually far from being mature and optimal, thus process redesign and reengineering take place to reform process inefficiencies, reduce cycle time, and remove non-value-added tasks [19]. The important question is that if models do not provide a vivid picture of processes to the systems' stakeholders, how can process inefficiencies be identified?

Models are also being used for training purposes and decision making procedures in addition to system documentation and development. For both training and decision making purposes, models should provide a graphical view of the system [33]. Training in healthcare systems with a huge number of staff, departments, and relations is an important issue and models can play an important role as training tools. This becomes even more important when target users are not information technology (IT) professionals. In fact, efficient and effective model design mostly depends on the modeler's creativity, but some modeling language features such as uniqueness of the language's notational constructs, and following the in-place standards in representation of each construct improve the quality of the target users' grasp of the model.

4.4 Optimized Models for Different Purposes

To cope with the complexity of healthcare processes through the creation of different detailed level models for use in the practice environment, process modeling languages should allow modelers to create optimized models with different numbers of specifications to enable model production specifically for some model-using purposes such as decision making, training, development, etc. For models that are intended to be used by healthcare staff with minimal IT knowledge, for instance, more detailed models can be confusing; hence, for this group of staff, models should only contain high level documentation of processes in mainly graphical format. However, models that are intended to be used by IT department staff require more detailed documentation, text based documents, and complete graphical representations. To comply with the need for different views, flexibility is required in the level of details and representation.

4.5 Evolution of Processes

The new definition of success for health services describes healthcare's goal as not merely change, improvement, and response, but changeability, improvability, and responsiveness [48], which can be seen as the 'evolving' dimension of healthcare process complexity. As a result, constant evolution and the dynamic nature of healthcare systems are the two main contributing factors to process complexity. Innovation, development of new technologies, and discovering medical techniques lead to the emergence of new expertise and sometimes to new units and departments, so a change in processes is very probable in order to adopt newly offered services [47]. Evolution and change in processes and systems introduce new modeling and implementation requirements. In the optimal situation, currently developed system models should be able to accommodate extended solutions to the new requirements. Since this is not always the case, a modeling language, at least, should be able to adapt to a wide range of changes in models without imposing

extensive remodeling [44]. The mentioned requirements can be satisfied by the support of modularity and abstract model design.

Modularity and support for abstraction are key concepts for change adoption in process models, and they are also pre-requisites of loose-coupled design. Modular models increase readability while decreasing resistance to change. Modularity also increases reusability in model development levels. Furthermore, due to the dynamic nature of healthcare, an agent's actions do not always follow normal routines since many emergency situations occur. In emergency situations, speed has the most important role; hence when agents react, they usually do not follow the routines, which may lead to conflicts. Modeling languages should be able to represent the dynamic nature of healthcare processes through exception handling.

4.6 Nested Processes and Integration

As defined earlier, an obvious attribute of a complex system is the number of its separate units and amount of interactions. However, complexity increases when some departments have overlapping tasks and transactions. In fact, overlapping features are representatives of combined functionality or departmental tasks, and overlaps can be eliminated by creating new standalone units, and assigning the overlapping tasks to them. Hence, overlaps in tasks and transactions are an additional factor in recognizing complex systems. In some cases, like healthcare, overlaps are so high that it is very costly to separate the overlaps in standalone units; as a result, overlaps are indispensable. The aforementioned factors create a nested structure in the systems' processes. Moreover, in complex systems, different agents may simultaneously be members of more than one unit, and their memberships may change frequently due to unforeseen events.

Nested healthcare processes create inherently non-linear relationships and remove boundaries between sub-processes [47], [48]. This reality complicates the modeling task

and increases the difficulty of changing a model as internal and external rule sets are not very strict. Therefore, process modeling languages should be able to represent healthcare's fuzzy departmental boundaries [47], [48].

The nested processes defined in the models are more critical in case of a need for change since the influence of change in complex systems is greater if inter-departmental interaction is tight. In this case, a change in an agent's task can affect other agents, so changes can be transmitted to other units in an incremental manner due to complex systems' non-linear behavior.

Integration of agents and departments across different subsystems is a common element of complex systems. Different subsystems may have dissimilar features and therefore divergent modeling requirements. In order to accommodate all requirements in models, it is sometimes essential to use different modeling languages for different departments. These models should be able to represent communication between departments and this would not be possible if modeling languages did not have matching features and the ability of a language to be mapped to other languages. We need to be able to map different modeling languages or integrate them together as needed to satisfy all requirements.

Chapter 5: Evaluation Framework for Business Process Modeling Languages in Healthcare

In Chapter 4, we identified general challenges for modeling healthcare processes, and requirements that healthcare processes impose on modeling languages. In this chapter we shall investigate the enabling features that help modeling languages to satisfy identified requirements, and propose a framework with eight components to address such an issue. As it will be detailed below, one requirement is usually met through one or several components/criteria of the framework (Figure 11).

Process modeling languages are innovatively designed or developed from existing modeling approaches by unifying several methods or adding features to existing ones [46], so most languages have several similar features in addition to a few unique components that distinguish them among other languages. We developed a framework (Figure 12) for evaluating modeling languages for service-based healthcare environments. The major difference of our framework from existing frameworks, besides its specificity, is that we consider the unique features of languages, in addition to modeling languages' common features, given that unique features are key metrics for specific uses. General modeling qualities, which are required by all systems, in addition to addressing healthcare specific requirements are also considered in our framework. As a result, the proposed framework is composed of important features that allow us to measure the quality of modeling languages. This chapter is dedicated to explaining our proposed evaluation framework and describing the metrics that are used for measuring each criterion. However, it is notable that this framework is the result of an exploratory

study, and the framework is the first representation of the results. Further developments to the framework will add more analyses concerning ranking the framework elements based on their importance in the development of specific models for different modeling goals. Examples of different goals include modeling for education, implementation, or system design.

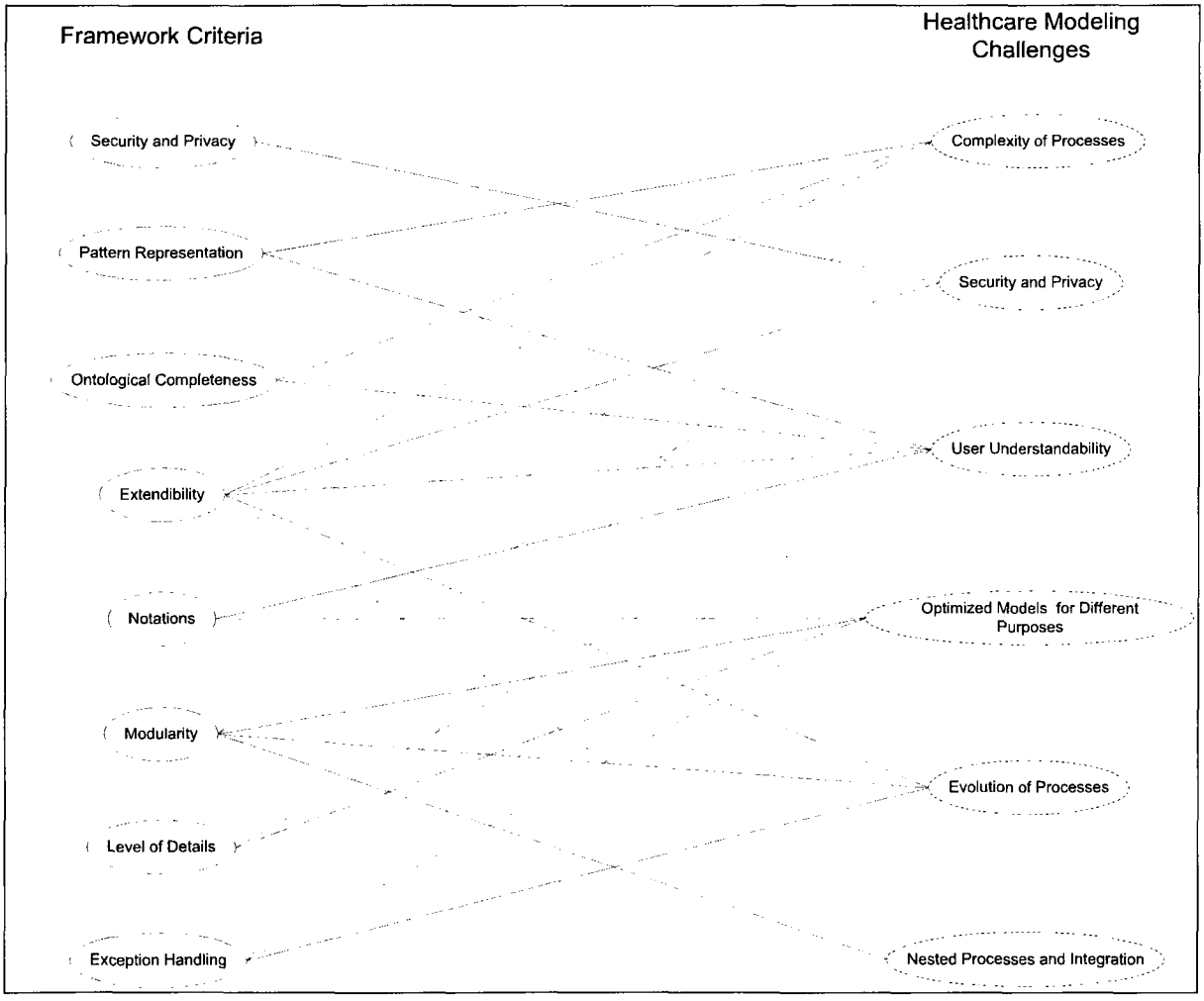


Figure 11: Relation between framework criteria and healthcare's modeling challenges

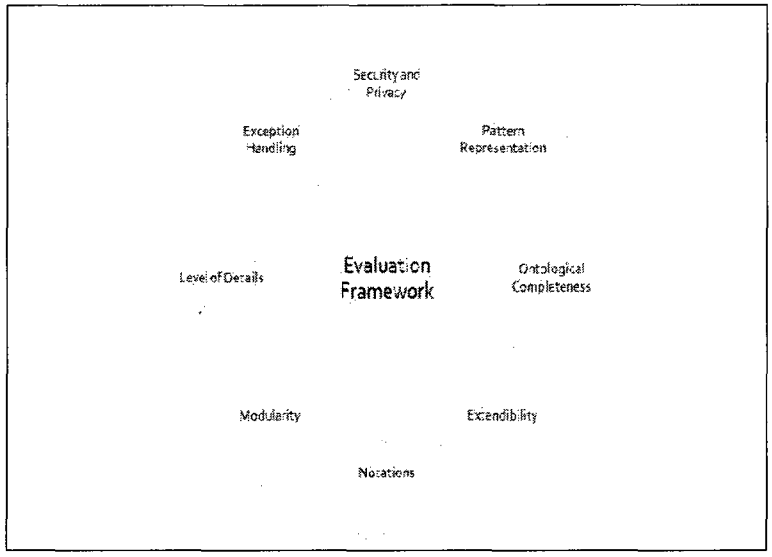


Figure 12: Evaluation Framework

5.1 Security and privacy

In all processes and systems, security and privacy are two important aspects to consider. They gain more attention and importance while dealing with service oriented architectures, and in systems that convey their communications over the Internet. In addition to systems, the processes themselves become privacy issues for process owners. In some cases, the process must not be exposed to partners, so the process will be hidden behind an interface, and all the communication is conveyed by means that are provided by the interface to keep the process private.

As mentioned earlier in Chapter 2, models are created and utilized for several purposes. As the models build the fundamental structure of the systems, security and privacy gain importance in the modeling as well as implementation. Security and privacy are of such importance that they are usually considered for most modeling purposes including training, development, documentation, etc.

As the need for security and privacy representation is critical for models, modeling languages should provide constructs to support these features. Support for security and privacy has a major effect on healthcare's requirements for security and privacy in addition to the need for optimized models. Different modeling goals need to represent these issues to some extent. Optimized models for implementation purposes should have more strict and detailed security factors while the sufficiently optimized level of detail for training or management should be restricted to graphical representation.

To identify language support for privacy and security features, the language should contain the security representational constructs. This fact is more related to representational languages, however, in case the modeling language supports execution and testing (e.g. BPEL) the security features should be embedded in the language, in a way that the processes could be executed with securely communicated messages. Hence, two factors are important: security constructs for representation, and implementation related features to implement secure processes.

5.2 Pattern representation

The understandability of a modeling language is a fundamental requirement for increasing the language's usability, and is tightly related to the process modeler's capabilities in addition to the modeling language's features. A modeling language's capability to represent data flow, communication flow, and control flow patterns in a complete and unambiguous way is an important criterion for the understandability of the language. The language will be more easily understood and used by the modelers when there are a defined set of elements corresponding to the existing identified patterns. Studies in [62] - [64] showed that some patterns cannot be modeled by any of studied modeling language, and some patterns can only be modeled using a limited number of languages. If there is a construct in the language that directly represents a pattern, the understandability of the language will increase. However, combining several constructs

in order to represent a pattern decreases the understandability and makes the language ambiguous, since all model developers may not be creative enough to do this, and also model users may have trouble understanding the complicated combination of language elements. In addition, there may be several combinations for representing a special pattern that makes the language and eventually the models ambiguous.

Rossi and Turini [54] stated many language advantages coming from full support for unambiguous pattern representation. They developed a modeling language (YAMN), which was later used by Van Der Aalst and his team, based on support for all patterns. They stated that more simply and unambiguously represented patterns help the language to be less complex, and more expressive, composeable, intractable. Moreover, models generated by the language are more maintainable. Furthermore, Recker and Mendling [50] argue that more pattern support increases the communicability and mapping between languages.

Pattern representation is important in the context of healthcare modeling since more unambiguously supported patterns mean a decrease in model complexity. Model complexity decreases due to the fact that each healthcare process is a combination of patterns, and as modeling languages provide elements corresponding to patterns, the model size and complexity decrease. Furthermore, familiarity with patterns and language elements increases model understandability; therefore processes can be better understood by different healthcare clinical and managerial actors.

Although we emphasize the number of represented patterns in this research, the quality and usefulness of patterns are also important. Some patterns (e.g. choice) are more useful for modeling and the lack of these patterns in a modeling language, makes it more incomplete, and sometimes useless. As a result, the patterns should be ranked based on their quality and importance. The importance of patterns will be developed in further extensions to this framework. To measure a language's compliance with this criterion,

three factors should be considered as stated before: Support for key patterns, number of supported patterns, and quality of support (lack of ambiguity).

5.3 Ontological Completeness

Ontological completeness refers to the existence of language elements to represent all ontological constructs in a way that there is no excess or redundancy, as explained in Section 2.9. Like pattern representation, ontological completeness is discussed in the context of healthcare to reduce model size and complexity, as well as to increase the understandability of models and processes.

Recker and Mendling [50] reach the same conclusion analyzing ontological completeness as they did for pattern support. More constructs in the language that correspond to ontological constructs makes the language more communicable and ease mapping between languages. Hence, the fact that languages can communicate and be mapped to each other provides some basis for language combination. Based on this fact, Green et al. [22] analyzed representation of ontological constructs for combined languages, and results show some improvements even though some overlaps are visible.

Pattern support and ontological completeness are very similarly effective in the context of healthcare. Both criteria are concerned with modeling languages' representational capability (see the discussion in Sections 4.1 and 4.3).

The same measurement criteria that are discussed for patterns can be applied to ontological constructs as well. Basically, there are some ontological constructs that are more critical for modeling languages, so a modeling language should include them all, in addition to other constructs. Again, the quality of support for constructs gains attention in order to decrease ambiguity.

5.4 Extendibility

As discussed earlier in this and the two previous chapters, modeling languages may not support all the requirements of modeling healthcare. To fulfill the requirements, there is either the option of combining the language with other possible solutions, or extending the language to support the requirements. Extensions help to support different technologies and different business sectors, and also to overcome modeling language deficiencies; however, extending modeling languages have some disadvantages that mostly affect the understandability of the languages and models, which represent one of the important aspects of healthcare modeling. Although many researcher such as Nysetvold and Krogstie [44] and Wang et al. [66] emphasized the advantages of extendibility, no many researchers talked about its disadvantages, so we will talk about some disadvantages in the next paragraph.

Model developers will also have to learn additional features of the modeling language, to be able to maintain the model in further developments. Moreover, model users are only familiar with basic model notations and language concepts, so extensions in a language can create confusion for those who are not familiar with the extended features. This problem might be an issue in healthcare where actors do not have extensive modeling and IT knowledge. Consequently, extendibility is a compromise between increased capability and general understandability.

Extensions and language extendibility are almost inevitable to eliminate some challenges in the healthcare modeling process (e.g. security representation, as it will be discussed in the next chapter), while they create some additional challenges. The extendibility factor affects all the identified criteria except the diversity of actors. As discussed earlier, since the level of IT knowledge among healthcare staff is different, extensions may require further training for the management and IT support staff.

5.5 Notations

Many researchers in the area of modeling and modeling languages address notations. Gruhn and Laue [24] analyzed the role of notations in model size and complexity. But the most important comment is perhaps from Rossi and Turini [54] who states that "A simple graphical notation is a fundamental requirement to improve the usability of a modeling language". This was verified by Mendling et al. [37] in a survey of different modeling notations, among expert and non-expert users which suggested that similar notations (or what we call following standards in notations) make the models more understandable. Wang et al [66] follow a similar approach in comparing BPMN and UML 2.0, and reach similar results.

Notation-wise, modeling languages are categorized into graph-based languages, rule-based languages, and a combination thereof [34]. Each group of languages is intended for a special group of audiences with a certain amount of knowledge and a special task.

Language notations affect the language's understandability for different knowledge levels, so it decreases the challenge regarding the diversity of actors. Meanwhile, it increases the ability to define the optimized level of models with various detail levels especially in the languages that support execution. Since actors involved in healthcare environments have different IT and modeling knowledge levels, the coherent and standard language notation improves model use for all healthcare actors.

Languages, for their usability in the healthcare modeling, can be notation-wisely measured by the way they follow the standards. In this thesis, the word "notation standard" refers to the set of accepted graphical elements with a special size, color, and shape. With regard to textual notation, the measurement is based on the same concept. Hence, the keywords representing concepts and descriptive words should follow their existing standards. For languages that provide execution features, separation of computation and representation of processes differentiate two diverse aspects of

modeling, so each actor can access a part of the model which addresses his/her needs. Meanwhile, uniformity (i.e. use of the same set of notation with unique and unified meanings) and formality (i.e. choosing commonly accepted graphical notations for language concepts) of notation improve the language's ease of use [34].

As a reference, Unified Modeling Language (UML) is accepted as a graphical standard for developing modeling languages. Most successive language developers adopted UML's graphical notation as a reference for language's notational elements [30]. On the other hand, for text-based languages, XML is becoming a standard.

5.6 Modularity

Modularity is a design method that refers to support for abstraction and use of pre-designed components in the final model or program. Modularity is one of the important features of languages. Paige et al. [46] discuss modularity as one of the modeling language design criteria with a major effect on simplicity, reusability and documentation. Other researchers [24], [54] reached similar conclusions regarding modularity. The modularity of modeling languages can be measured by considering the support for abstract processes and sub-processes. Abstraction prevents the revelation of the underlying layers of a process, hence improving the understandability of process models for non-developer actors. Also, the developers and designers' comprehension level of the models will improve, and the modeling task will be easier when the unimportant and/or more detailed data is hidden.

On the other hand, reusability is an important factor and it is tightly related to modularity. The ability to use previously modeled processes increases the speed and accuracy of modeling. It also reduces the time required for understanding the models. Maintenance and model modification is easier with reused parts. Reusability has a positive connection with understandability, in that more reused sections in models help decrease the required

effort for learning the process. However, it is not necessarily true that languages that support sub-processes and abstraction also support reusability.

Modularity is effective in almost all healthcare's modeling challenges. Modular design reduces the complexity which is one of the main issues in healthcare models. It also increases the ability to hide unnecessary information for different model users and various model uses. This factor is important in healthcare because of the existence of different actors with different grasp levels of models. Meanwhile, modular models can tolerate the evolving nature of healthcare processes better.

5.7 Level of detail

The level of detail provided by modeling languages should be optimized for different modeling proposes. This metric has a close positive relationship with other aspects of a modeling language such as notation, abstraction and ability of the language to execute processes. Detailed documentation of processes improves the level of understanding for the designer and developer actors; however, the end user, who uses the models for training, does not need detailed information, and more detailed documentation may impair his understanding. Consequently, modeling languages should be flexible in the level of details, and provide facilities to support both detailed documentation and high level representation of processes. This criterion can be measured by support for abstract processes. Also, languages should not force the modelers to create detailed models. Languages that support execution should be able to separate execution and representation in order to hide unnecessary details.

Generally, languages that separate graphical notation and text-based notations and support both are more flexible in the level of details. Roser and Bauer [53] emphasized this factor and considered it as a requirement for developing different levels of complexity and detail in models for different purposes. Bobkowska [11], on the other

hand, investigated different methods of graphical model development in order to handle different levels of detail inclusion in graphical only models. The author stated that ties in the graphical representation and text-based models make model creation cumbersome.

Flexibility in the level of detail affects the challenges of modeling healthcare because of the diversity of actors and their expectations from models. High level and less detailed models enhance the general view of the model for healthcare's high level management actors who are not concerned with implementation details. However, other actors who are involved in the details of processes or are members of an IT team need a fully detailed view of the system. For the first group, graphical models are better, but the latter group needs a detailed description of processes that can be achieved using text-based models.

5.8 Exception handling

Exception handling is an important area of evaluation. Exceptions have been indispensable parts of business process modeling for more than two decades [17]. Exception handling features increase modeling difficulty, but at the same time, they increase the adaptability of models to exceptional circumstances. Improved adaptability helps model users to predict all model behavior before and at the time that exceptions occur.

Healthcare environment and its processes are full of exceptions in various forms. There are many emergency situations occurring everyday that do not follow the routine procedures and workflows. However, most emergency situations have their own defined procedures that are closely followed by caregivers and healthcare administrations. These exceptional situations and the way that they should be processed must be foreseen in model development. Use of exception handling features improves readability of decreases process model complexity.

Investigations in modeling languages show that, for instance, not all modeling languages contain cancellation functionalities. In the case of those languages, the processes end naturally; the modeler does not have the ability to terminate a process at a special time. Being able to define the termination conditions and terminate processes at a certain moment improves the understanding of process timelines.

The evaluation framework we proposed provides a starting point to evaluate different modeling languages in the healthcare domain. The strength of the framework is its extensiveness with regard to covering unique features of languages, which makes it an unbiased test-bed for comparing different language families. Implications for general modeling are also considered.

Chapter 6: Assessment of Modeling Languages

This chapter is intended for a thorough evaluation of three modeling languages (i.e. BPMN, BPEL, and WS-CDL) for their usability in the healthcare modeling based on our proposed framework. Based on what we have described in Chapters 3, 4, and 5, identifying a suitable web service based modeling language for the healthcare sector is a difficult task. This assessment focuses on language features and also leads to better understanding of each language's backbone methodology.

6.1 Choice of the modeling languages to be analyzed

The strategy for choosing a modeling language lies in the features of the existing modeling languages. The choice of languages was made based on healthcare modeling requirements. Since three of six requirements are related to healthcare actors, their understanding of the models and languages, and the process complexity, one of the criteria for choosing a modeling language was decided to be its ease of representation (i.e. graphical notations to be easily understood). As a result, one of the chosen languages is BPMN which is a fully graphical language without restrictions with regards to technology. On the other hand, for a more detailed analysis and adding non-graphical languages to the review, we considered BPEL and WS-CDL. Since choreography and orchestration are two main service oriented modeling methodologies, we chose two leading languages, are from each side, in order to conduct our assessment. Each language's features will be assessed based on the evaluation framework to investigate their compliance with identified requirements.

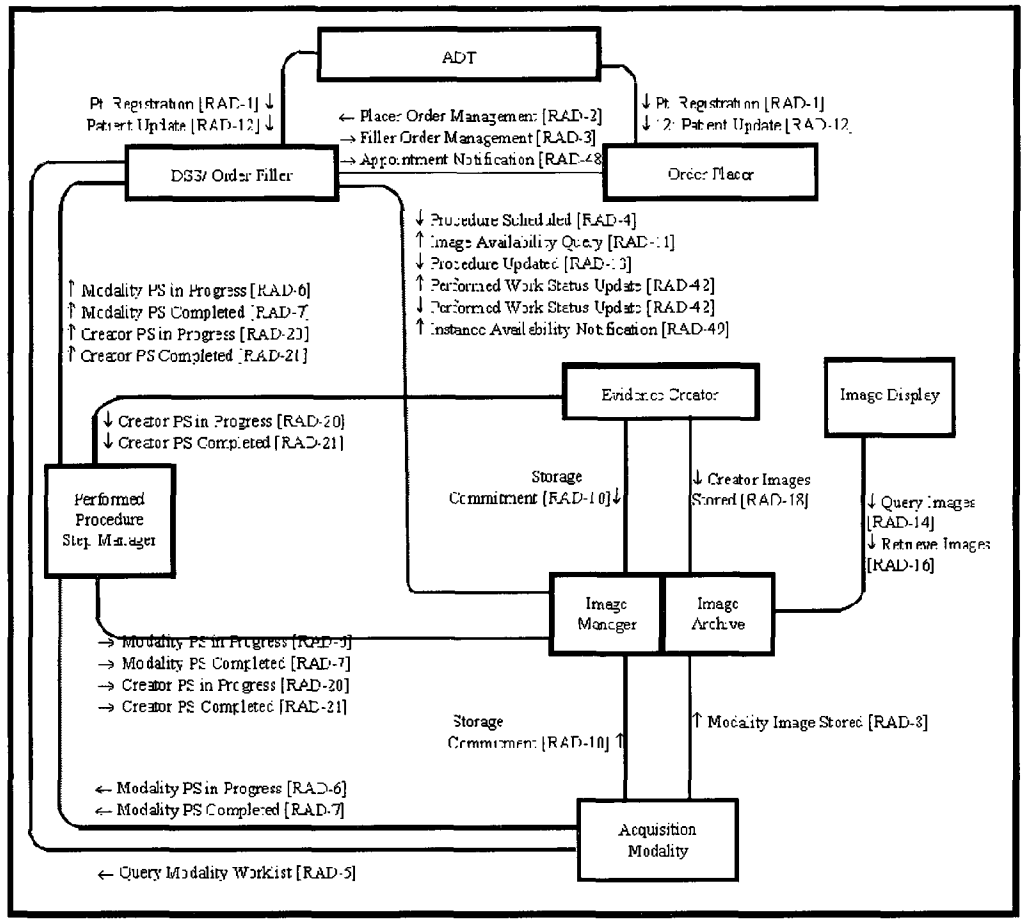


Figure 13: Scheduled Workflow Diagram, from IHE Int. [29]

6.2 Case study

In this subchapter, a complex healthcare service-based process is chosen as a case study, so all the languages will be tested for their ability to model process in a practice. Scheduled workflow (SWF) is an IHE process that represents the ordering, scheduling, imaging acquisition, storage and viewing activities associated with radiology exams [1], [29]. Consisting of twenty transactions and eight different IHE Actors, SWF is representative of healthcare processes. SWF provides a complete test base for the

evaluation of our framework because of its complexity, large amount of domain communication, and service like behavior. Since security, privacy, permanent change, and exceptional situations are indispensable parts of healthcare processes, SWF represents all features of healthcare processes and all requirements of healthcare process models.

SWF establishes the continuity and integrity of basic departmental imaging data. The IHE integration profiles discuss the integration needs of healthcare sites and the integration capabilities of healthcare IT products [28]. The IHE radiology integration profile specifies a number of transactions throughout the imaging acquisition procedure in order to maintain the consistency of patient and ordering information. It also provides the scheduling and imaging acquisition procedure steps [29]. This profile also provides a central coordination scheme for process completion and scheduling notifications. Figure 13 represents the complexity of transacted messages as well as the large number of involved units in SWF.

The first step in modeling is to understand the process and its requirements (here, participating services). The sequence diagram in Figure 14 represents the interaction flow and the timing between services on patient admission when a request is made to perform an imaging procedure on a patient. UML sequence diagrams are widely used as a representation language for workflows, especially when there are many transactions involved in the process. In addition, knowing the transmitted messages and their flow will help to create robust communication protocols in choreography which will be discussed later in this chapter.

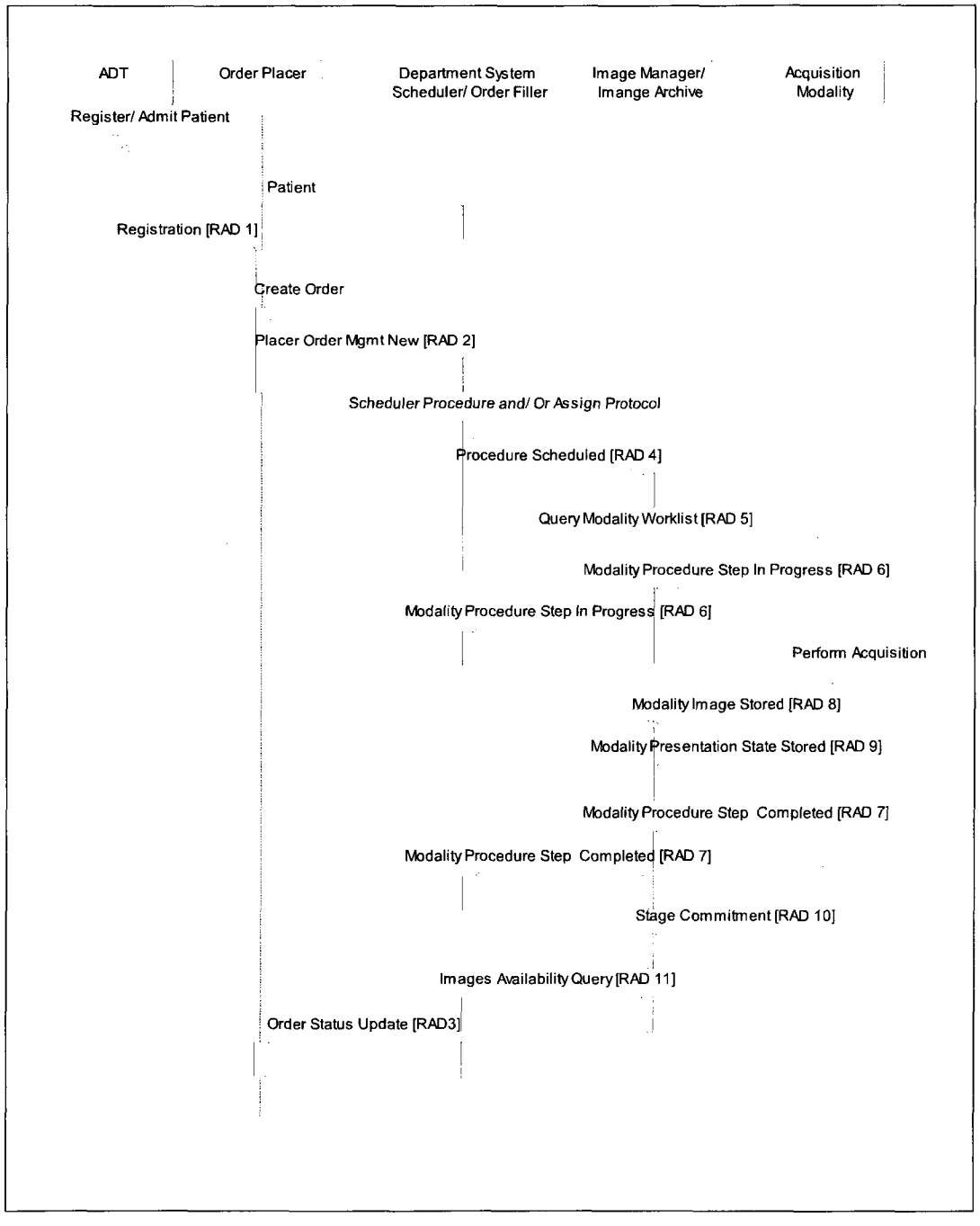


Figure 14: SWF's Administrative and Procedure Performance process Flows

As a test-base we selected to represent a procedural section of SWF because it has a more service-based process like behavior. The admission procedure, which includes several transactions and inter-departmental interactions between Order Pacer, Order Filler, and ADT (Admit, Discharge & Transfer), is the most suited fraction of SWF for our purpose. It is complex enough to show the capabilities of modeling languages, and it is the most used process section in SWF, so its reliability and understandability are important. Furthermore, based on the day-to-day evolution of healthcare systems, it is more likely to change than other processes.

The sequence diagram (Figure 15) is derived from the one presented in IHE documentation [29]. The compensation messages are presented for the readability of processes when they are modeled in BPMN, BPEL and/or WS-CDL. The services involved can be easily identified in the diagram.

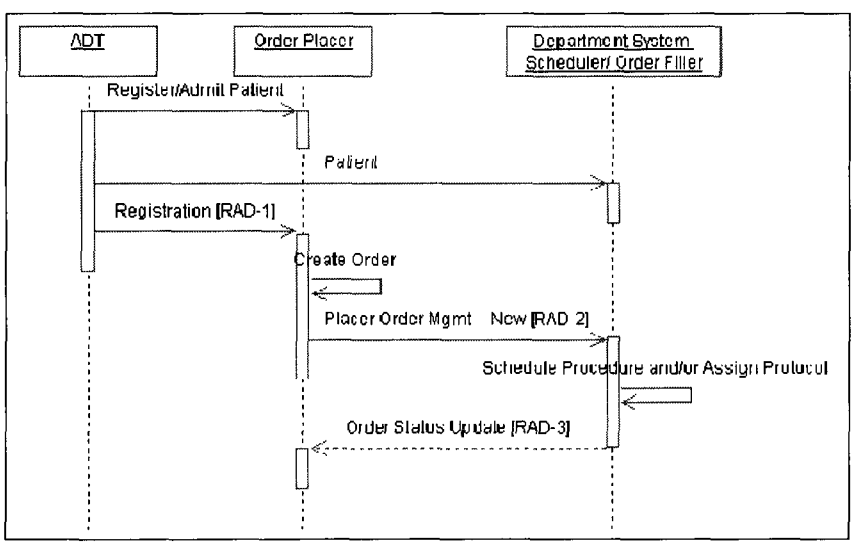


Figure 15: Part of SWF

Table 4: BPMN Security data types, from Rodriguez et al. [51]

Name	Description
SecReqType	Represents a type of security requirement that must be specified for non-repudiation (NR), attack/harm detection (AD), integrity (I), privacy (P), or access control (AC)
PerOperations	An enumeration for possible operations related to permissions granted over an object.
protectDegree	An abstract level representing criticality as low (L), medium (M), or high (H)
PrivacyType	Anonymity (A) or confidentiality (C)
AuditingValues	Representing different security events related to processes' security requirements

6.3 Assessment of BPMN

BPMN is a graphical workflow modeling language that does not focus on a specific technology; thus, it is able to model processes based on service oriented architectures. By definition, BPMN was created to simplify model development, increase model readability and understandability, and decrease the complexity of models [68]. Moreover, BPMN provides a way to connect the modeling directly to the implementation by its ability to be converted into executable BPEL processes. Based on this, it is expected that BPMN can provide solutions for challenges in healthcare business process modeling in most aspects, since the main characteristic of healthcare processes is their complexity, and the actors' understanding of processes.

In this section we assess BPMN against our proposed framework to identify its suitability for healthcare process modeling in service-based environments, and the language will be tested in the test-bed defined in the previous section.

Table 5: BPMN Security elements

Name	Description	Notation
SecurityRequirement	Abstract class that contains security requirements specifications	
NonRepudiation	Establishes the need for denial of interaction	
AttackHarmDetection	Indicated the degree of attack detection success	
Integrity	Establishes the degree of non authorized corruption	
Privacy	Indication of the degree that non authorized parts are avoided to receive classified information	
AccessControl	Indicates the need to define access control mechanism	

6.3.1 Security

BPMN, in its standard description, has no construct to represent security and privacy in models, but there are various security representations provided in extensions to the language. Therefore, BPMN, without the use of extensions, lacks the fundamental security features. Rodrigues et al. [51] developed a security-enabled version of BPMN, but it is not yet accepted as an inclusion to the standard. Extensions, in some cases, make a significant alteration in languages. In case of BPMN's security enabled version, security enabled elements, and features (Table 4 and Table 5) are added to the language in addition to some modifications to the language's basic elements to embed the security features.

Although BPMN's security extension is readable and easily understandable, it is not a basic language characteristic. Therefore, inclusion of these features in this research will

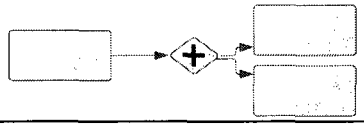
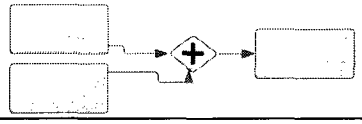
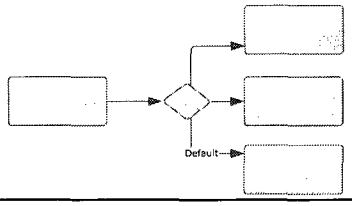
be out of scope. As it is discussed in section 5.4, extensions always reduce understandability, so BPMN is identified as a language lacking security features.

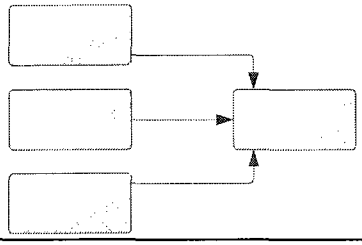
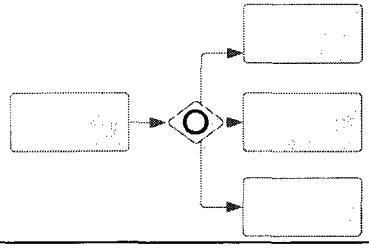
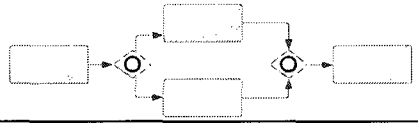
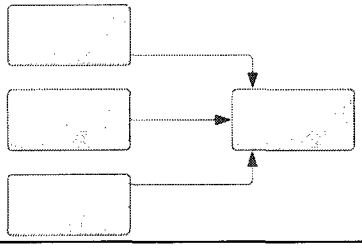
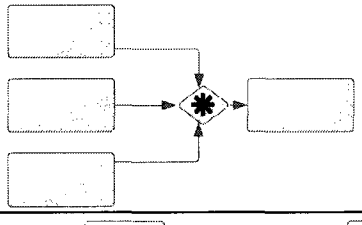
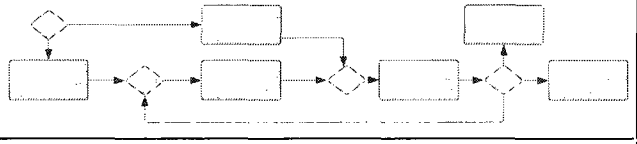
6.3.2 Pattern support

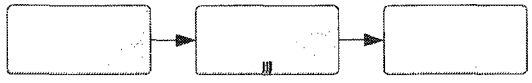
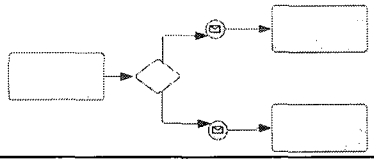
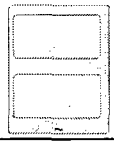
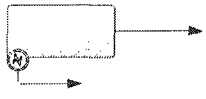
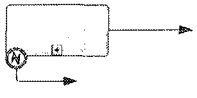
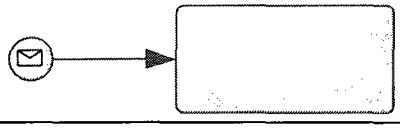
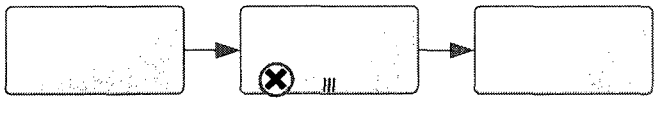
Based on our evaluation criteria, one of the measurements is the level of support for the workflow and data flow patterns involved in healthcare processes. Patterns are described in detail in section 2.8 and also documented in [71].

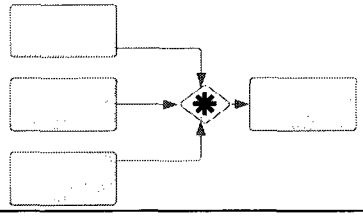
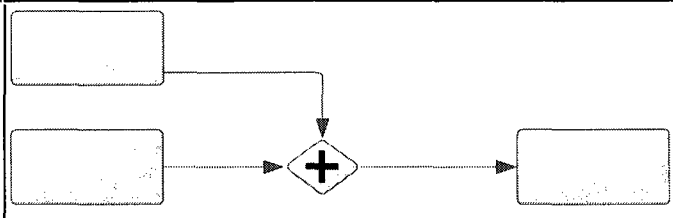
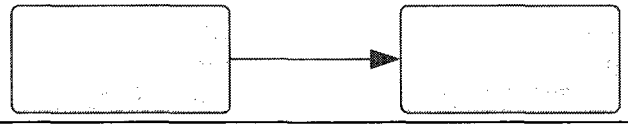
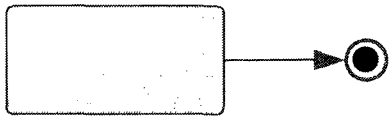
BPMN is a powerful language in terms of support for patterns, as it supports sixteen patterns in a fully understandable manner (Table 6); however, all the ways used by BPMN to support patterns are not easily understandable. The understandability level of the patterns can be measured by the criteria that are discussed earlier in Chapter 5.

Table 6: BPMN workflow pattern support

Pattern	BPMN Support	Understand-ability
Sequence		+
Parallel Split		+
Synchronization		+
Exclusive Choice		+

Simple Merge		+
Multiple Choice		+
Synchronizing Merge		+
Multiple Merge		+
Structured Discriminator		+
Arbitrary Cycles		+
Implicit Termination		+
Multiple Instances without Synchronization		+

Multiple Instances with a Priori Design-Time Knowledge		+
Multiple Instances with a Priori Run-Time Knowledge		+
Multiple Instances without a Priori Run-Time Knowledge	N/A	-
Deferred Choice		+
Interleaved Parallel Routing		+
Milestone	N/A	-
Cancel Activity		+
Cancel Case		+
Structured Loop		+
Recursion	N/A	-
Transit Trigger	N/A	-
Persistent Trigger		+
Cancel Region	N/A	-
Cancel MI Activity		+
Complete MI Activity	N/A	-

Blocking Discriminator	N/A	-
Cancelling Discriminator		+
Structured Partial Join	N/A	-
Blocking Partial Join	N/A	-
Cancelling Partial Join	N/A	-
Generalized AND-Join		+
Static Partial Join for MI	N/A	-
Cancelling Partial Join for MI	N/A	-
Dynamic Partial Join for MI	N/A	-
Acylic Synchronizing Merge	N/A	-
General Synchronizing Merge	N/A	-
Critical Section	N/A	-
Interleaved Routing	N/A	-
Thread Merge		+
Thread Split		+
Explicit Termination		+

6.3.3 Ontological Completeness

As explained earlier in Chapters 2, 4, and 5, findings in [22] on ontological mapping of modeling languages and Bunge-Wand-Weber (BWW) ontological base model show that modeling languages are not able to represent all the ontological constructs documented in [65], [67]. We studied healthcare processes for their constructs and mapped them to BWW models. Our findings show that not all of the ontological constructs are used in healthcare processes.

Table 7 illustrates the constructs used in healthcare and the findings and methods used in [22] are used in order to evaluate the ability of BPMN in support for the ontological constructs (BPMN construct hierarchy is represented in Figure 16).

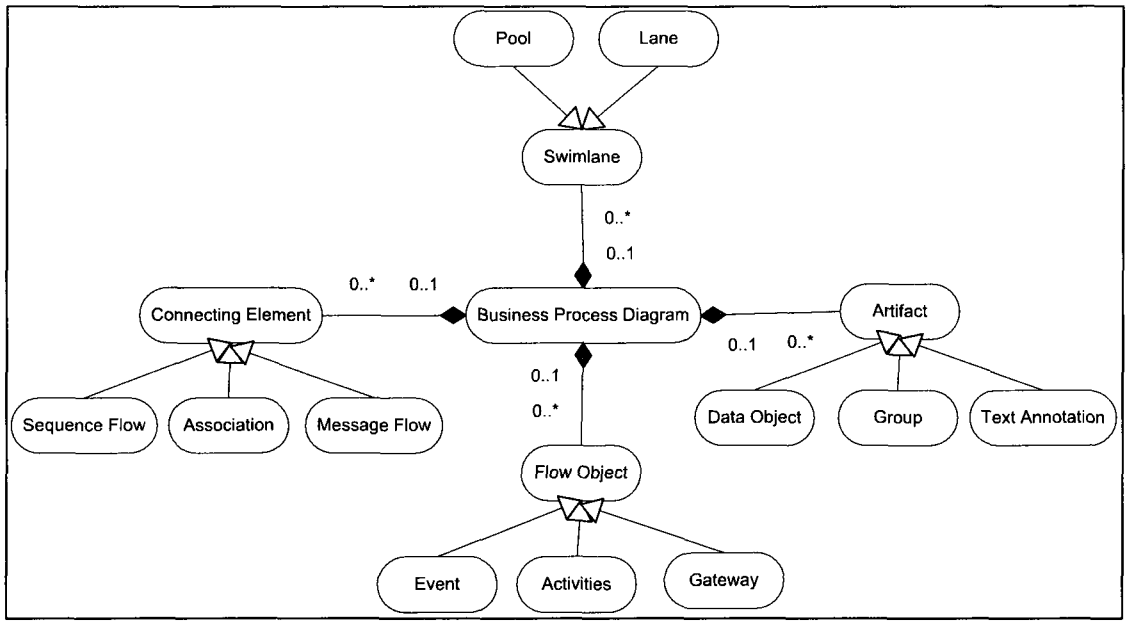


Figure 16: Construct hierarchy in BPMN, from Rodriguez et al. [51]

Table 7: BPMN support for healthcare service based ontological constructs

Constructs	BPMN Support
Property	+
Class	-
State	+
Conceivable state space	-
State Law	-
Lawful State Space	-
Event	+
Conceivable Event Space	+
Transformation	+
Lawful Transformation	+
Lawful Event Space	+
Coupling	+
System	+
System Composition	+
System Environment	-
System Structure	+
Subsystem	+
System Decomposition	+
Level Structure	+
External Event	+
Stable State	-
Unstable State	-
Internal Event	+

6.3.4 Extendibility

BPMN is intended to be extensible by modelers and modeling tools. BPMN's extensibility enables modelers to add non-standard elements in order to satisfy specific modeling requirements. One of the fundamental rules in extension and extendibility is that the basic elements and also the look of the diagrams made by the extended modeling language should not change to keep the understandability and readability in a high level. Furthermore, mappings to execution languages may be affected if new flow elements are added to BPMN. BPMN satisfies the need for additional modeling elements by providing the concept of artifacts that can be linked to the existing flow objects through associations, yet do not affect the basic elements of BPMN and do not cause difficulties on converting the BPMN models to the executable BPEL processes. Moreover, graphical notations of BPMN are defined in an open way and allow additional information or marks to be included in graphical notations.

6.3.5 Notations

BPMN follows an accepted standard for its notations. Its representational components are unique in both size and shape. BPMN's elements enable the easy development of simple flowchart-like diagrams that look familiar to most business analysts. Elements are chosen in a way that they are distinguishable from each other and they utilize shapes that are familiar to most modelers. For instance, activities are represented in rectangles and decisions are diamond-shaped. As it is mentioned, BPMN qualifies for all requirements for graphical notation. However, its lack of textual and more detailed notation may bring some difficulties for implementers even though BPMN has a mechanism that enables its models to be converted to executable BPEL processes.

6.3.6 Modularity

The overload of transmitting information, the high amount of communication and the need for reusability make sub-processes an indispensable feature in healthcare process modeling. BPMN supports sup-processes in the modeling procedure as a basic internal feature of the language. The ability to describe the processes in a modular way decreases the BPMN model size. It also increases the readability of models while decreasing complexity.

BPMN has an element named "sub-process" to directly support modularity. The sub-process element has two active ways of representation: collapsed and extended. Figure 17 represents the basic elements that support sub-processes in BPMN.

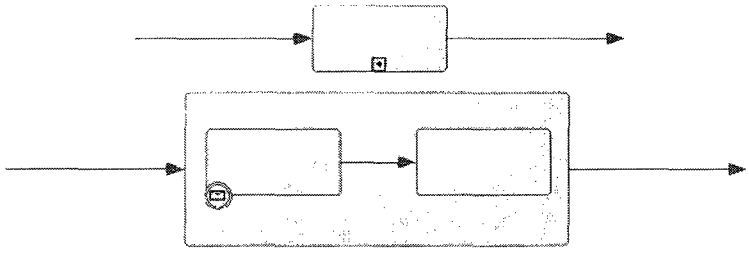


Figure 17: Sub-Processes in BPMN

6.3.7 Level of Detail

Different healthcare model users require different views of the processes. Different detail levels specify the intended use of models. BPMN is very flexible in the level of detail. Although it does not allow the model designer to add coding to the model, additional detail for special model users can be added in the form of comments. BPMN does not force the user to a pre-specified level of comments as opposed to some other modeling languages in which the model designer is forced to create fully detailed models.

6.3.8 Exception Handling

BPMN provides a complete set of exception handling and compensation features. Healthcare processes demand adequate support for exception handling features to increase the understandability of processes. The following paragraphs introduce and explain exception handling objects in BPMN.

Error End Event: This type of event indicates that a named error should be generated (triggered) as a consequence of the process ending.

Figure 18: BPMN's Error End Event

Error Intermediate Event: This type of Event can only be attached to the boundary of an activity. It reacts to (catches) a named error, or to any error if a name is not specified.

Figure 19: Error Intermediate Event

Exception flow: This occurs outside the Normal Flow of process and is based on a BPMN's intermediate event that occurs during the performance of process. Exception handlers catch error triggers, interrupt the activity and create exception flow when error intermediate events are attached to the boundary of activities. The triggers could come from an error end event, technical fault or system fault.

Figure 21 illustrates the complexity of processes, BPMN's modularity, and exceptions handling in BPMN. It also represents the modeling of selected healthcare processes (SWF).

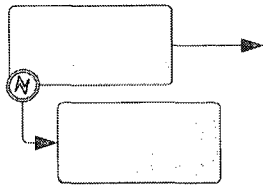


Figure 20: BPMN's Exception Flow

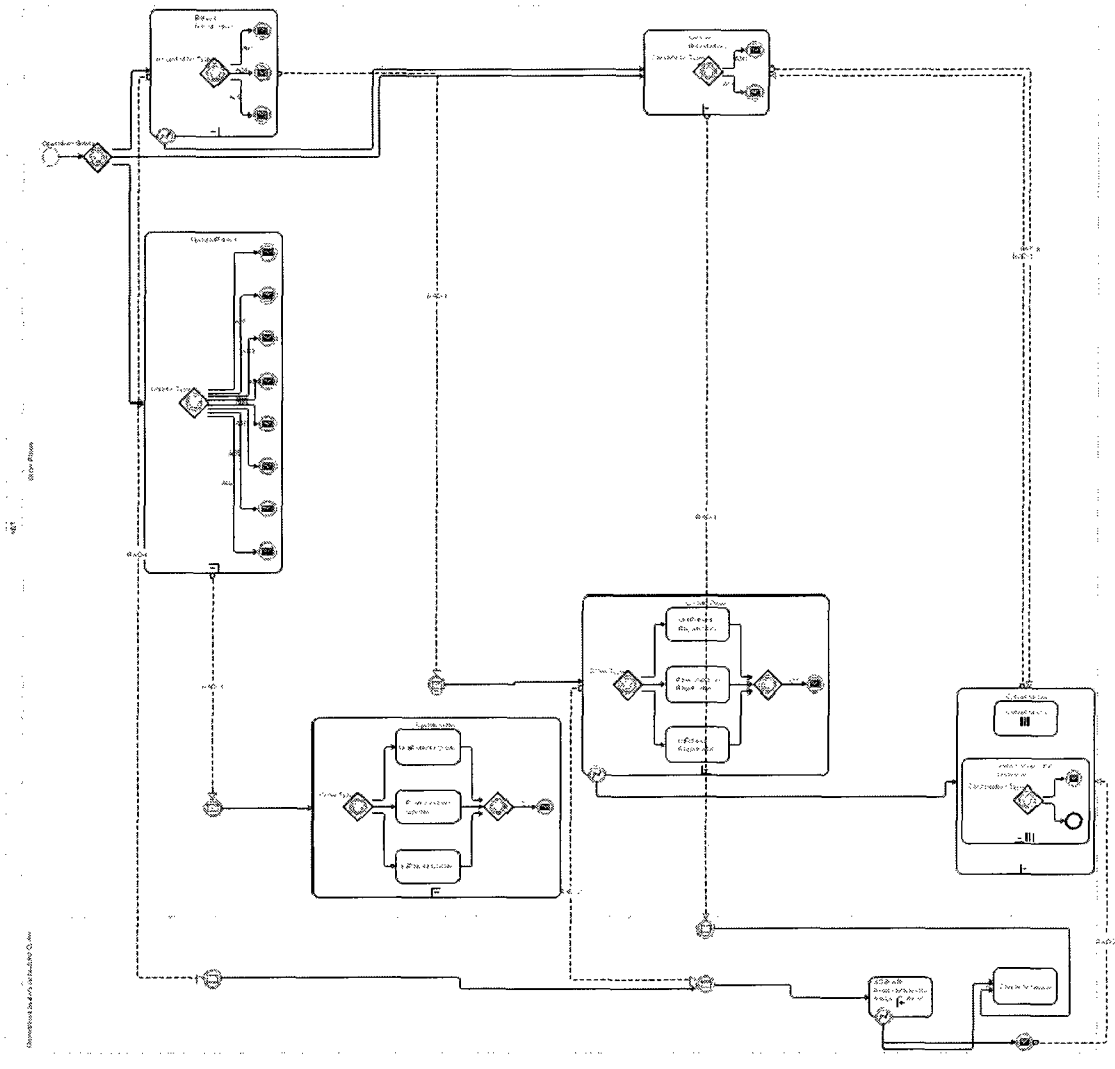


Figure 21: BPMN implementation of SWF

Table 8: Pattern Analysis of BPEL

Pattern	BPEL Support	Ambiguity
Sequence	<sequence>	+
Parallel Split	<flow>	+/-
Synchronization	Placing an activity after Parallel Split pattern	+/-
Exclusive Choice	<switch> OR using Transition Conditions and Links	+
Simple Merge	Placing an activity after Exclusive Choice pattern	+
Multiple Choice	using Transition Conditions and Links	+
Synchronizing Merge	Use of Join Condition in Multiple Choice Pattern	+
Multiple Merge	N/A	-
Structured Discriminator	N/A	-
Arbitrary Cycles	N/A	-
Implicit Termination	<flow>	+
Multiple Instances without Synchronization	<while> AND <invoke> then using other activity to instantiate the target activity	+/-
Multiple Instances with a Priori Design-Time Knowledge	Replicate activity as many times as needed	+
Multiple Instances with a Priori Run-Time Knowledge	<while> AND <pick> AND Message events to create and finish Activities (Not a complete support)	-
Multiple Instances without a Priori Run-Time Knowledge	<while> AND <pick> AND Message events to create and finish Activities, and also stop Activity creation (Not a complete support)	-
Deferred Choice	<pick>	+
Interleaved Parallel Routing	<scope> Combined with <flow>	+
Milestone	<while> (Not a complete support)	-
Cancel Activity	<terminate>	+
Cancel Case	Using fault and compensation handlers	+
Structured Loop	<while>	+
Recursion	N/A	-

Transit Trigger	N/A	-
Persistent Trigger	<pick>	+
Cancel Region	activities within the same scope can be cancelled	+/-
Cancel MI Activity	N/A	-
Complete MI Activity	N/A	-
Blocking Discriminator	N/A	-
Cancelling Discriminator	N/A	-
Structured Partial Join	N/A	-
Blocking Partial Join	N/A	-
Cancelling Partial Join	N/A	-
Generalized AND-Join	N/A	-
Static Partial Join for MI	N/A	-
Cancelling Partial Join for MI	N/A	-
Dynamic Partial Join for MI	N/A	-
Acylic Synchronizing Merge	links within a <flow> construct	+
General Synchronizing Merge	N/A	-
Critical Section	<scope>	+
Interleaved Routing	<scope>	+
Thread Merge	Needs extensions	-
Thread Split	<invoke> construct in conjunction with the correlation facility	+/-
Explicit Termination	N/A	+

6.4 Assessment of BPEL

BPEL is a textual modeling standard, which enables modelers to create detailed models for different purposes. Moreover, BPEL is an execution language that transfers the visualized understanding of the processes to an executable prototype or even final

implementation of processes. The following section evaluates BPEL against our framework.

Table 9: Ontological constructs supported in BPEL, from Green et al. [22]

Construct	BPEL's Capability
Property	+
Class	+
State	+
Conceivable state space	-
State Law	-
Lawful State Space	-
Event	+
Conceivable Event Space	-
Transformation	+
Lawful Transformation	+
Lawful Event Space	-
Coupling	+
System	+
System Composition	+
System Environment	-
System Structure	+
Subsystem	-
System Decomposition	-
Level Structure	-
External Event	+
Stable State	-
Unstable State	-
Internal Event	+

6.4.1 Security and Privacy

Like BPMN, BPEL contains no constructs to represent task level or process level security in the models and neither provides a complete secure collaborative basis in the execution.

Although WS-Security [41] can be embedded in transmitted messages between services in BPEL, it does not provide any representational aspect, and cannot be easily identified by non-modelers. The expressiveness of process languages becomes important in complex business sectors such as healthcare, where non-modeling experts interact with models for many purposes, including training, decision making and process improvement.

As a language meant for execution of services, BPEL takes advantage of more developed extensions that contain added security features. However, these extensions, which contain representational features, need extra learning effort, plus modeling tools do not support all extensions and probably need some plug-ins installed. Moreover, to the best of our knowledge, none of the proposed extensions are complete enough to be mentioned as an acceptable solution by any of the organizations supporting BPEL.

6.4.2 Pattern support

As is mentioned in previous chapters, investigation on BPMN was also based on work done by Van Der Aalst et al., [71] and some similar research [70] has proven that none of the studied process languages are capable of representing all known documented patterns. However, the level of support differs significantly in different languages.

The result of evaluating each pattern in conformance with BPEL's capabilities is presented in Table 8. The elements and methods that make BPEL capable of representing patterns are detailed in the table. Meanwhile, the understandability of their method in that representation is indicated in a separate column in the table. As mentioned before, where there is a construct in the language that represents a pattern (+) is assigned to that pattern. If the pattern cannot be presented by the language, we use the (-) sign. If there is no construct designed in the language which exactly corresponds to a pattern but the pattern can be presented by combining language constructs, the sign (+/-) is used.

6.4.3 Ontological Completeness

Findings in [22] on ontological mapping of modeling languages and BWW ontological base model show that modeling languages are not able to represent all the ontological constructs documented in [65], [67].

6.4.4 Extendibility

BPEL is extendable and can accept new features to the language. For healthcare modeling, BPEL needs a security extension and it also needs an extension in order to support sub-processes. There are many extensions developed for BPEL e.g. BPEL4People [32] to include human user interface into the languages, and BPEL-SPE [32] to support sub-processes. The currently existing extensions, mainly BPEL4People, changed the language in many aspects. However, since BPEL is an executable language, the extensions should also be implemented in the supporting tools for its execution.

6.4.5 Notations

BPEL follows a fairly accepted standard (i.e., XML) for its notations. From the graphical notations' viewpoint, BPEL has a set of accepted non-standard format and graphical notations. However, BPEL's notation is not easily understandable by non-modeler actors since BPEL's textual and graphical notations are entwined; hence a model user must understand BPEL's background coding as well as its graphical notation. Moreover, this inseparability makes it rather cumbersome for the non-expert user to change graphical BPEL models.

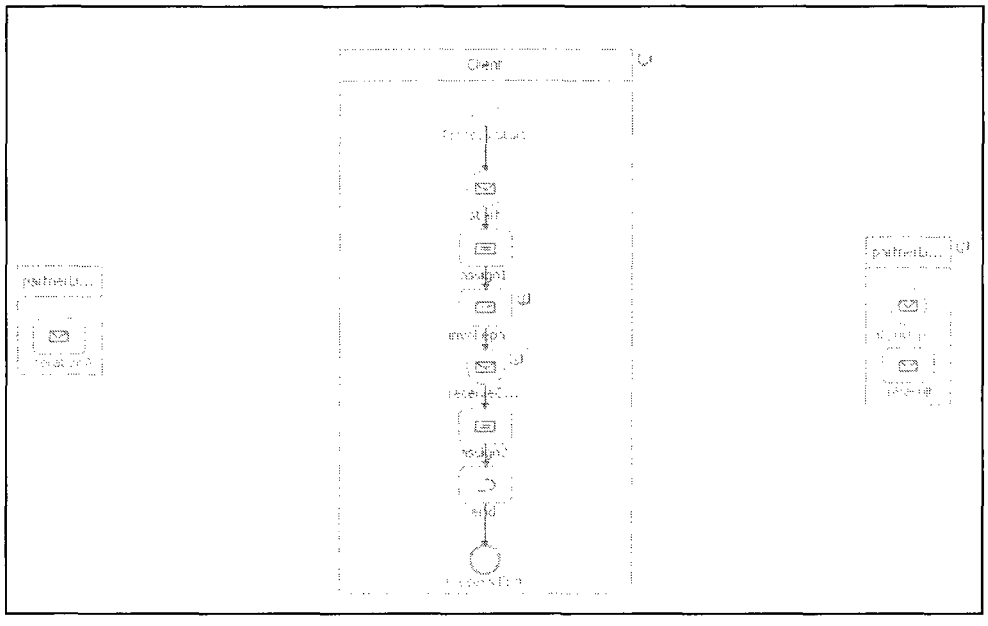


Figure 22: Super Process Implementation in BPEL

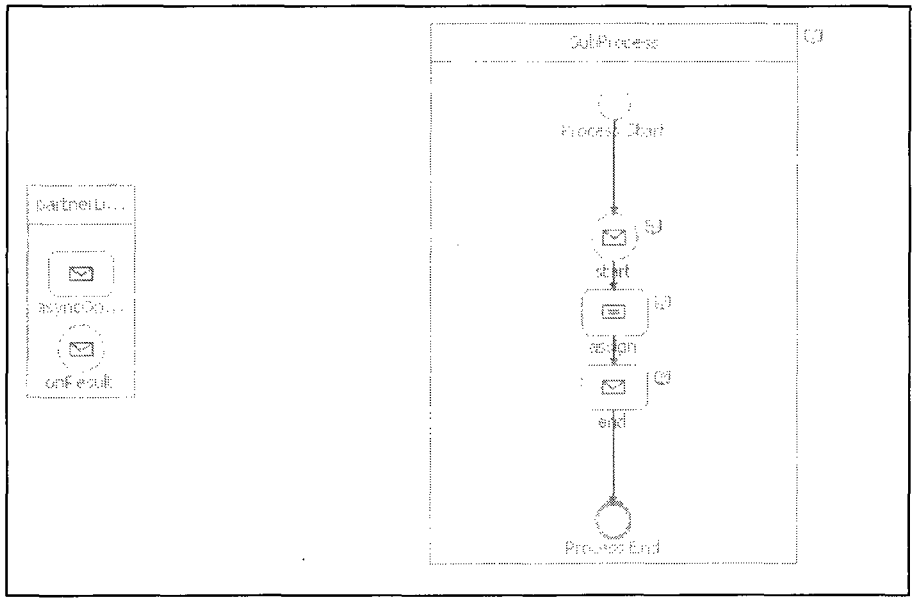


Figure 23: Sub-Process Implementation in BPEL

Listing 1: Implementation of Super-process

```
<sequence>
  <receive
    name="start"
    partnerLink="partnerLinkA"
    portType="wsdlNS:MyPortTypeClient"
    operation="operationA"
    variable="inputVar"
    createInstance="yes">
    <correlations>
      <correlation set="correlator" initiate="yes">
        <documentation>
          Correlation set is initiated with the values of the properties found in the message.
        </documentation>
      </correlation>
    </correlations>
  </receive>

  <assign name="assign1">
    <copy>
      <from variable="inputVar" part="inputType"/>
      <to variable="partnerInputVar" part="inputType"/>
    </copy>
  </assign>

  <invoke
    name="invokepartner"
    partnerLink="partnerLinkB"
    portType="partnerNS:MyPortType"
    operation="asyncOperation"
    inputVariable="partnerInputVar">

    <documentation>
      The process sends a message to the partner (AsynchronousSample process will receive it).
    </documentation>
  </invoke>
  <receive
    name="receiveCallback"
    partnerLink="partnerLinkB"
    portType="partnerNS:MyCallbackPortType"
    operation="onResult"
    variable="partnerOutputVar"
    createInstance="no">

    <documentation>
      The process receives a response from the partner.
    </documentation>

    <correlations>
      <correlation set="correlator"/>
    </correlations>
  </receive>
```

Listing 2: Implementation of Sub-Process

```
<sequence>
   <receive
      name="start"
      partnerLink="partnerLinkA"
      portType="wsdlNS:MyPortType"
      operation="asyncOperation"
      variable="inputVar"
      createInstance="yes">

      <documentation>
         The Receive activity makes the process to wait for a message to arrive.
      </documentation>
   </receive>

   <assign name="assign">
      <documentation>
         The Assign activity copies data from the input variable to the output variable.
      </documentation>

      <copy>
         <from variable="inputVar" part="inputType"/>
         <to variable="outputVar" part="resultType"/>
      </copy>
   </assign>

   <invoke
      name="end"
      partnerLink="partnerLinkA"
      portType="wsdlNS:MyCallbackPortType"
      operation="onResult"
      inputVariable="outputVar">

      <documentation>
         The Invoke activity allows the business process to call the client and send a response message.
      </documentation>
   </invoke>
</sequence>
```

Listing 3: Implementation of interfacing partnerlink

```
<wsdl:message name="requestMessage">
   <wsdl:part name="inputType" element="xs:typeA"/>
</wsdl:message>

<wsdl:message name="responseMessage">
   <wsdl:part name="resultType" element="xs:typeA"/>
</wsdl:message>

<wsdl:portType name="MyPortType">
   <wsdl:operation name="asyncOperation">
      <wsdl:input message="tns:requestMessage"/>
   </wsdl:operation>
</wsdl:portType>
```

Listing 4: Message and port type declarations

```
<message name="RAD-2">
   <part name="OrderInfoTypoe' type="xsd:string"/>
</message>
<message name="RAD-4">
   <part name="ScheduleInfoType" type="xsd:string"/>
</message>
<message name="RAD-6">
   <part name="ModalityProcedureProgressType" type="xsd:string"/>
</message>
<message name="RAD-7">
   <part name="ModalityProcedureCompletedType" type="xsd:string"/>
</message>
<message name="RAD-11">
   <part name="ImageAvailabilityQueryType' type="xsd:string"/>
</message>
<message name="ImageAvailability'>
   <part name="ImageAvailabilityType" type="xsd:string"/>
</message>
<message name="RAD-3">
   <part name="OrderStatusType" type="xsd:string"/>
</message>
```

```
<portType name="OrderFillerPortType">
   <operation name="ProcedureScheduled'>
      <input name="input1" message=":ns:RAD-2"/>
      <output name="output1' message="tns:RAD-4"/>
   </operation>
   <operation name="ModalityProgress'>
      <input name="input2" message=":ns:RAD-6"/>
   </operation>
   <operation name="ModalityCompletionOperation">
      <input name="input3" message=":ns:RAD-7"/>
   </operation>
   <operation name="QueryImageAvailability">
      <input name="input4" message=":ns:ImageAvailability"/>
      <output name="output3' message="tns:RAD-11"/>
   </operation>
   <operation name="UpdateOrderStatus">
      <input name="input5"/>
   </operation>
   <operation name="ScheduleProcedure">
      <input name="input6" message=":ns:RAD-2"/>
   </operation>
</portType>
```

Listing 5: Declarations of partners and partnerlinks

```
<partnerLinks>
   <partnerLink name="OrderFillerPL" xmlns:tns="..." partnerLinkType="tns:partnerlinktypeOrderFiller" partnerRole="OrderFillerPortTypeRole"/>
   <partnerLink name="OrderPlacerPL" xmlns:tns="..." partnerLinkType="tns:OrderPlacer" partnerRole="OrderPlacerPortTypeRole"/>
   <partnerLink name="Secretary" xmlns:tns="..." partnerLinkType="tns:ADT" myRole="ADTRegistraror"/>
</partnerLinks>
```

Listing 6: Central process

```
<sequence>
      <receive name="RegistrationInitiation" createInstance="yes" partnerLink="Secretary" operation="PatientRegistration"
xmlns:tns="http://j2ee.netbeans.org/wsdl/ADT" portType="tns:ADTPortType" variable="PatientRegistrationRequestreceived"/>
      <flow name="Flow1">
         <if name="If1" xmlns:tns="http://j2ee.netbeans.org/wsdl/OrderPlacer">
            .
            .
            .
            .
            .
         </copy>
      </assign>
      <reply name="RegistrationCompleted" partnerLink="Secretary" operation="PatientRegistration" xmlns:tns="http://j2ee.netbeans.org/wsdl/ADT"
portType="tns:ADTPortType" variable="PatientRegistrationOut"/>
   </sequence>
```

6.4.6 Modularity

The overload of transmitting information, the high amount of communication and the need for reusability make sub-processes an indispensable feature in healthcare process modeling. BPEL, generally, does not provide any support for sub-processes as a standard.

Using BPEL, the goal is to model processes in a web services environment. For implementing the whole model as an orchestration, the central process will regulate communication between services if actors are considered as service providers. In this system, however, actors need to perform internal processes. There are two ways to represent this system in BPEL. The first way is to provide an abstract central process with conceptual transactions between units in addition to actors' internal processes modeled in different BPEL processes with a reference to a central process (Figure 22 and Figure 23, and also Listing 1, Listing 2, and Listing 3 represent this method from graphical and textual views in which *Client* plays the role of super-process and *SubProcess* represents sub-process implementation). This makes the process simple and easy to understand without forgoing the model's execution feature. The second way is to use extensions. While extensions to BPEL do not affect its understandability by users familiar with extensions, understandability decreases for non-professional users. Moreover, the use of extensions decreases the portability of models.

6.4.7 Level of Detail

Different healthcare model users require different views of the processes. Different detail levels specify the intended use of the models. To a large extent, BPEL is inflexible in the level of detail that can be provided to the end user. It more closely resembles a programming language than a modeling language and does not allow for any form of analysis. BPEL process models are associated with coding which contains a high volume of detail. BPEL's limitation is its inseparability of graphical notation from background executable code. Healthcare managerial staff does not need to see detailed code while it might be necessary for IT staff. The inflexibility of BPEL at the detail level makes the modeling, on some levels, cumbersome and decreases the understandability of models to non-IT actors.

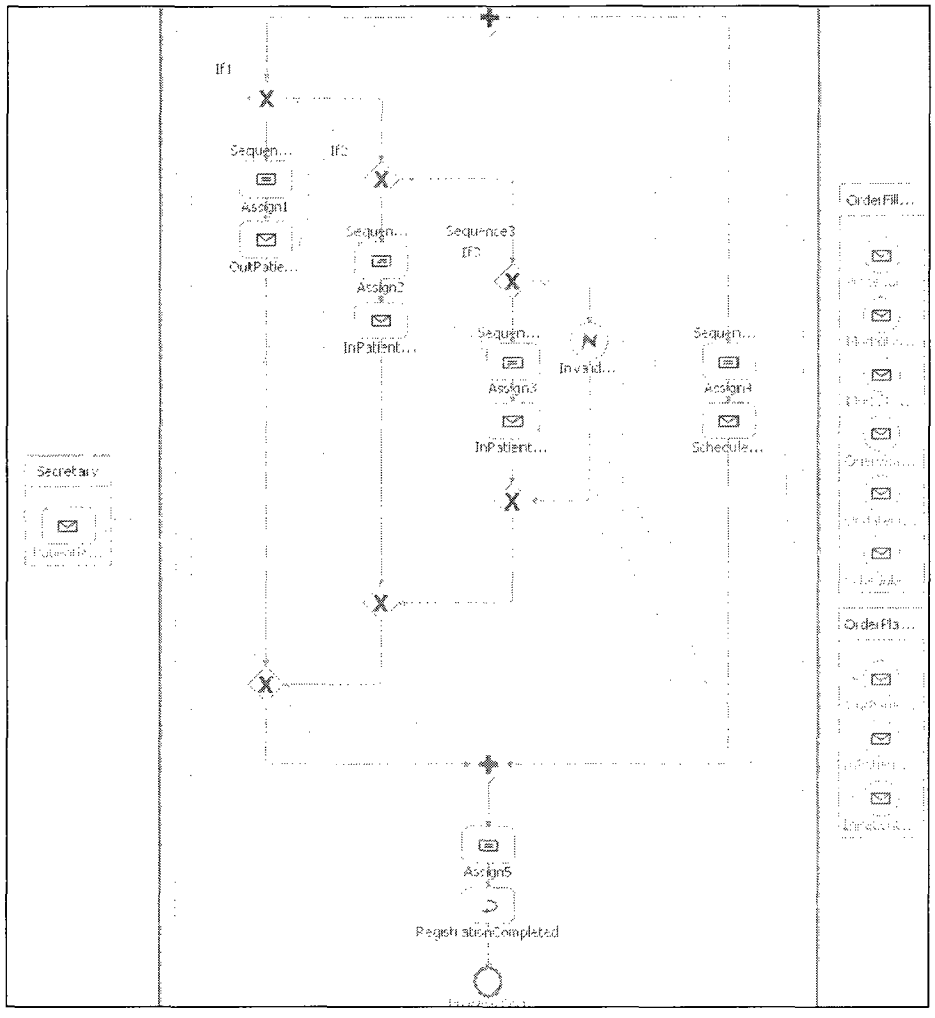


Figure 24: ADT sub-process and exception handling in BPEL

6.4.8 Exception Handling

BPEL provides a complete set of exception handling and compensation features. Exception handling features are very important in BPEL due to its ability of process execution. Healthcare processes demand adequate support for exception handling to increase process understandability. However, from a representation point of view, BPEL is limited in dealing with exceptions since there is no cancellation element in it. The

modeler cannot decide to end a process instantly, and the process continues until it gets to the ending point. Due to this limitation, the result would be a larger model with many more connections even though the model will be easier to follow as it terminates only at one final point.

6.4.9 BPEL Implementation

At this step, we derived a BPEL process specification from the *Patient Registration* activity diagram. The BPEL specification contains definitions of *types*, *variables*, *messages*, and *correlationSets*. Also, business *partners* and a *process* using basic and structured activities are defined. The WSDL file contains a *portType* and a *serviceLinkType Section* to define the web services. Finally, compensation activities are provided using scopes and security issues are outlined. Clearly, to start BPEL modeling processes requires a significant effort. Every part of the process model should be analyzed and pre-designed with a predefined structure. We apply the main steps of BPEL development procedure, and then we represent BPEL features in a test-bed to assess them in accordance with the evaluation performed earlier using our framework.

First, the partners section, the supported *serviceLinkType*, as well as messages, and data types should be carefully developed. The next step is the development of the central process which orchestrates the interaction flow in the whole process. To complete the modeling process, transactions, security, error handling, and compensation features will be applied next. Listing 4, Listing 5, and Listing 6 represent portions of the coding that has been done for the SWF process and Figure 24 illustrates the model's graphical view.

With respect to modeling, we faced all challenges that we expected and mentioned in the framework-based evaluation. The code sections of models are long and not user readable. They do not help in the presentation of security measures from a modeling perspective. The tight relationship between BPEL's graphical notation and its background coding

makes it difficult to optimize representational details for training purposes or management use.

One of the main issues that came to our attention is model reusability. BPEL models are not reusable in that for a small change in the process, the major part of the process will be affected and sometimes it would be necessary to remodel the whole process. This issue is even more important for processes where changes are more frequent. BPEL is less capable of adopting new processes or services after the modeling process is finalized. It is worth mentioning that we did not use any extensions to BPEL in order to measure the language's genuine capacities and features.

6.5 Assessment of WS-CDL

The Web Services Choreography Description Language (WS-CDL), a W3C recommendation, is an XML-based language for describing how peer-to-peer participants collaborate regardless of programming model and platform. WS-CDL describes processes from a global viewpoint, and documents their common and complementary observable behavior by modeling ordered message exchanges between processes. The following section is dedicated to investigating WS-CDL's features and assessing its ability to model healthcare processes.

6.5.1 Security and Privacy

WS-CDL does not contain any construct supporting task level or process level security representation in models. Although WS-Security [41] can be embedded in transmitted messages between services in WS-CDL, it does not provide any representational aspect, and cannot be identified by non-modeling experts.

Table 10: Pattern Analysis of WS-CDL

Pattern	WS-CDL Support	Ambiguity
Sequence	<sequence>	+
Parallel Split	<parallel>	+
Synchronization	<parallel>	+
Exclusive Choice	<choice>	+
Simple Merge	<choice>	+
Multiple Choice	<parallel> structure containing guarded work units	+
Synchronizing Merge	<parallel> structure containing guarded work units	+
Multiple Merge	Non-blocking <perform>	+/-
Structured Discriminator	Using parallel split with blocking condition	-
Arbitrary Cycles	N/A	-
Implicit Termination	Activating sub-choreographies using blocking <perform>	+
Multiple Instances without Synchronization	<parallel> AND non-blocking <perform>	+
Multiple Instances with a Priori Design-Time Knowledge	<parallel> AND blocking <perform>	+
Multiple Instances with a Priori Run-Time Knowledge	Instantiating sub-choreographies in a loop	-
Multiple Instances without a Priori Run-Time Knowledge	Instantiating sub-choreographies in a loop	-
Deferred Choice	<choice> without a guard work unit in its structure	+
Interleaved Parallel Routing	N/A	-
Milestone	State capturing variables AND non-blocking work units	+
Cancel Activity	Use of exception handling features	+
Cancel Case	Use of exceptions handled in the root choreography	+

Since WS-CDL is not aimed for execution of services, the security measures are less important. Some techniques exist to represent security features in WS-CDL with the help of other modeling languages [25] which do not introduce large differences in the use and portability of models. Since WS-CDL is XML-based, is mainly executed by support of Java services, and does not have a graphical view, the portability of models would not be

much affected. However, the issue of model understandability for users who are unfamiliar with these techniques and extensions remains unsolved. Moreover, in the theory of service composition, since corporate entities are often unwilling to delegate control of their business processes to their integration partners, WS-CDL (or service choreography in general) offers more privacy by means of participation contracts and elimination of a central process.

6.5.2 Pattern support

WS-CDL is a description language that is used to unambiguously describe observable service collaborations. WS-CDL is used to describe internal workflows of processes that involve multiple services, yet it does not support all workflow patterns documented in [71]. WS-CDL is also assessed against the existing identified patterns by [71], like the other two languages under evaluation, and the result [72] of evaluating each pattern in conformance with WS-CDL's capabilities is presented in Table 10.

6.5.3 Ontological Completeness

The findings of Green et al. about the ontological mapping of modeling languages illustrated that modeling languages are not able to represent all the ontological constructs documented in [65], [67], or if they were capable of representing some patterns, their representation might not be clear. Table 11 illustrates the ontological constructs are implemented into WS-CDL.

6.5.4 Extendibility

WS-CDL is extendable and can accept new features into the language. An execution extension to WS-CDL can relieve modelers from using Java services in order to implement WS-CDL choreographies.

Table 11: Ontological representation of WS-CDL, from Green et al. [22]

Construct	WS-CDL's support
Property	+
Class	-
State	+
Conceivable state space	-
State Law	-
Lawful State Space	-
Event	+
Conceivable Event Space	+
Transformation	+
Lawful Transformation	+
Lawful Event Space	+
Coupling	+
System	+
System Composition	+
System Environment	-
System Structure	+
Subsystem	+
System Decomposition	+
Level Structure	+
External Event	+
Stable State	-
Unstable State	-
Internal Event	+

It is important to mention that WS-CDL is incapable of showing the order and direction of transferred messages between services and processes, and all that WS-CDL describes is the ordering rules for the messages. Hence, as a process language, WS-CDL needs a notation that represents the flow of messages in the process.

6.5.5 Notations

Although WS-CDL follows a fairly accepted standard (i.e. XML) for its notations, WS-CDL has some drawbacks with regard to constructs and their naming. WS-CDL is barely equipped for graphical representation of processes. Without considering its lack of graphical notation, WS-CDL still suffers from possible confusions resulting from the naming of its constructs. Several WS-CDL constructs, like *RoleType*, *ParticipationType*, and *ChannelType*, give the wrong impression that there should be some possibility to create separate instances from them, but in fact, they function as entities. The existence of *Type* in their naming definition may create some confusion among language users [20]. Also, as an XML-based non-graphical language, specifications of even the smallest interactions become rather lengthy. The language also introduces a wealth of new design concepts (e.g., channel passing), constructs for direct expression of concurrent execution and nondeterministic choices, that are currently not well understood in the world of web services. The channel passing mechanism is poorly explained in the standard. It is not exactly clear which conditions the reference text puts on channel usage. Moreover, in general, the channel passing mechanism sounds overly clumsy. The intention of the language designers with regards to channel mechanisms is, in general, far from being clear.

6.5.6 Modularity

The overload of transmitting information, the great amount of communication and the need for reusability make sub-processes an indispensable feature in healthcare process modeling.

WS-CDL is well suited for reusability purposes. It coordinates the services without need of a central process, that makes the alternation of models more difficult, and choreographies can be created from parts contained in several different choreographies.

Therefore, existing choreographies can be combined to form new choreographies that may be reused in different contexts, and the same choreography definition is usable by different participants operating in different contexts. Furthermore, since the contracts are described in a modular fashion, reading and learning the processes is much easier. Based on that, WS-CDL can easily represent sub-processes, and the abovementioned scenario's internal and external one-to-many relationships. Besides, strict modularity of WS-CDL takes away all global variables, conditions or work units from the modeler, but still enables centralized storage to have global variables in order to avoid repetitive definitions.

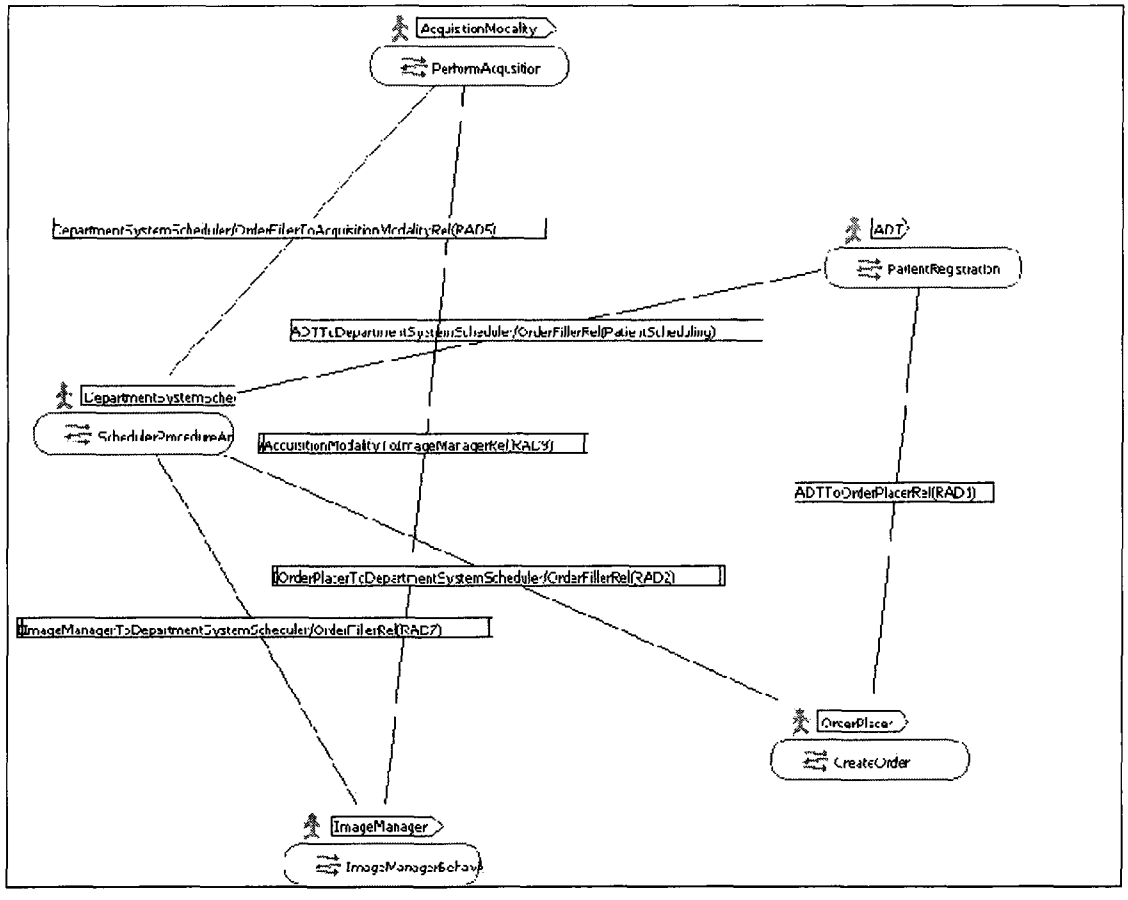


Figure 25: Roles and Relationships of SWF visualized by Pi4SOA

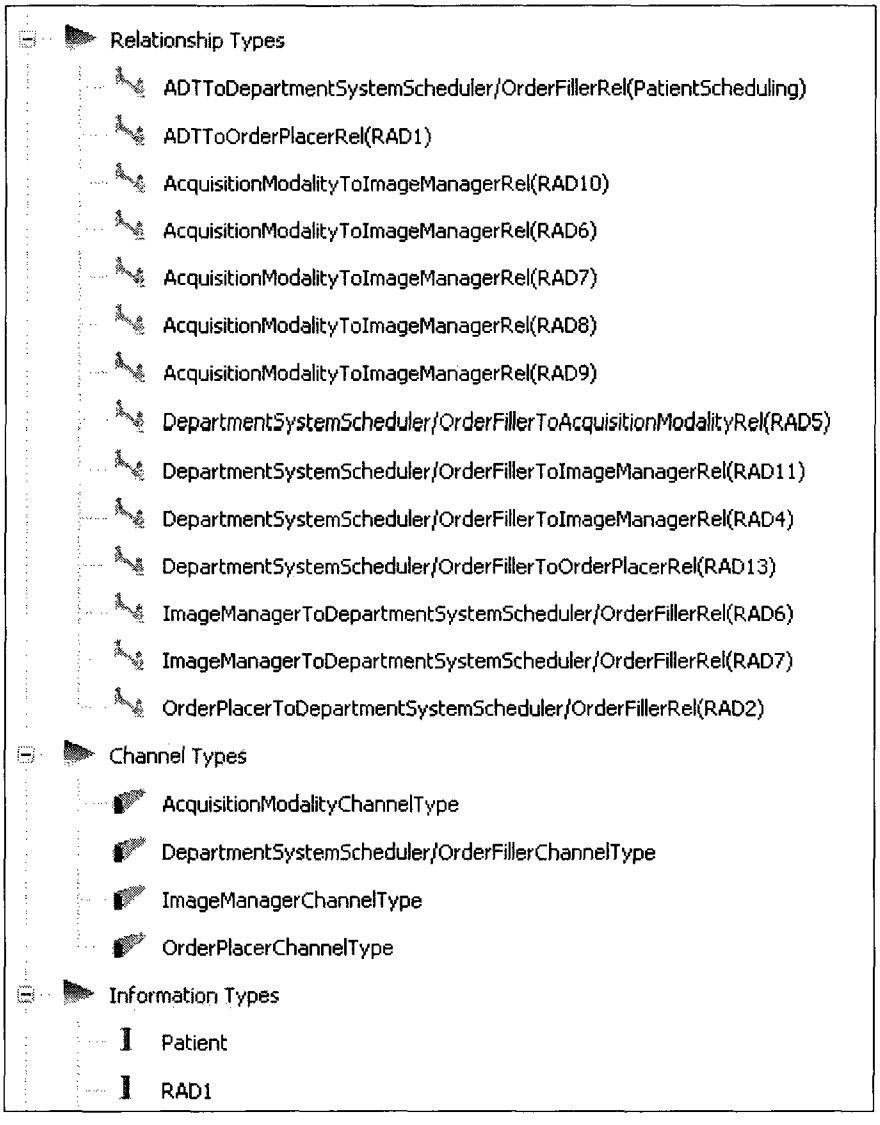


Figure 26: *RelationshipType* definition for SWF

6.5.7 Level of Detail

WS-CDL is genuinely modular, and capable of defining abstract service interactions. Hence, the model details can be hidden in required situations. Yet it is still too detailed for non-development purposes, and its lack of graphical representation makes it almost

unusable for training purposes, even though it can include human readable documentation and semantics for all components in the choreography.

6.5.8 Exception Handling

Exception handling mechanisms in WS-CDL are poorly explained in general. For instance, there is some confusion in the standard documentation regarding whether exceptions impact only certain role types in choreography, or they affect all role types. The likely intention is that choreography coordination is required to notify all role types, but this is inadequately specified in the standards documentation [20], [31].

6.5.9 WS-CDL implementation

The first step of modeling in all languages is definition of roles and relationships. Since the case study has not changed, the roles and relationships do not change from what existed in earlier models developed in this thesis. However, Pi4SOA, a plug-in added to Eclipse IDE creates WS-CDL modeling environment for Eclipse, and is used for assessment of WS-CDL in this thesis. Pi4SOA provides graphical model-based features to enable modeling roles and relationships of systems. Figure 25 represents the roles involved in the SWF.

During the previous step, the relationship types engaged in SWF are defined. Figure 26 represents the *RelationshipTypes* defined for the model represented in Figure 25.

The final step in the description of choreography is creating variables and values, as well as generating the choreography description. For this thesis we modeled a selected part of SWF represented in Figure 15 using Pi4SOA Scenario Description that is represented in Figure 27. This selection only deals with patient registration and care scheduling in the healthcare environment and SWF. Figure 28 illustrates the graphical representation of the choreography generated by Pi4SOA for the selected part.

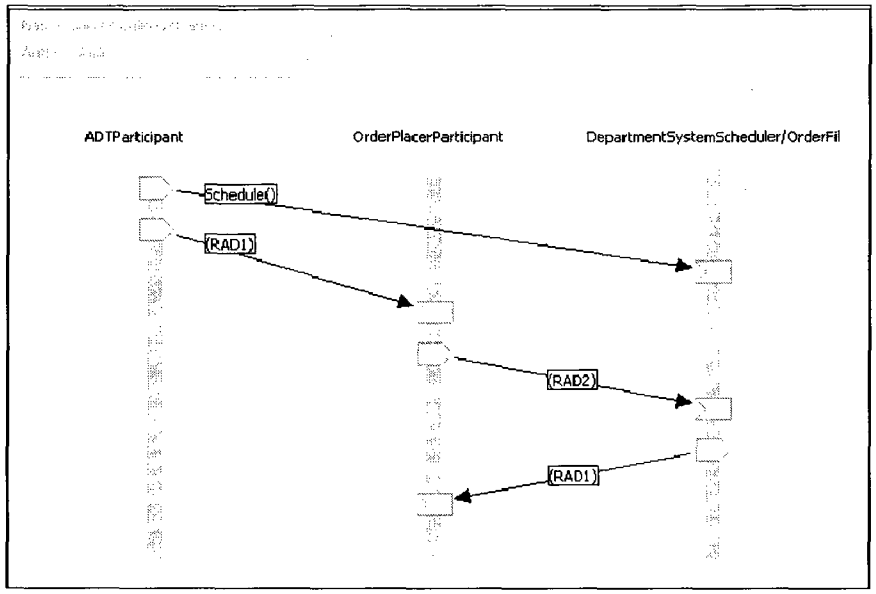


Figure 27: Scenario description for part of SWF designed for WS-CDL modeling

Listing 7 shows the main choreography part of the WS-CDL document for our SWF example. Although WS-CDL provides more features to support healthcare requirements, the long WS-CDL documents are rather hard to analyze and understand for non-professional model users, as can be seen in the code snap. Moreover, WS-CDL sometimes fails to provide description of workflows if proper WSDL files are not available. Finally, the lack of a standard integrated graphical notation makes the understanding of the described choreographies time consuming.

WS-CDL can be seen as a programming language with limited capabilities. It obviously lacks some features of a modeling language (i.e., graphical notation) while it has some extra features (i.e., limited execution) that separate it from other description languages.

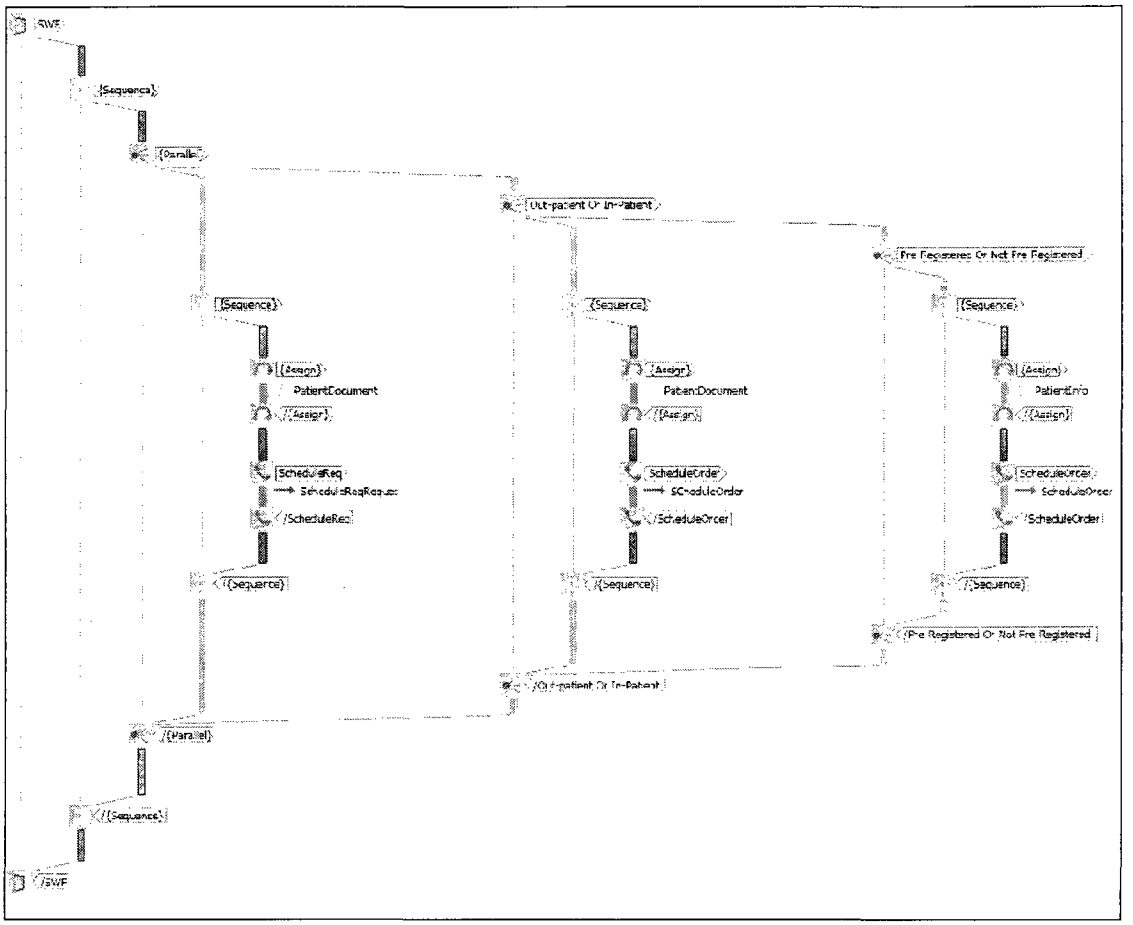


Figure 28: Graphical Choreography Description for SWF generated in Pi4SOA

Listing 7: Selection of a WS-CDL description

```
<choreography name="SWFChoreography" root="true">
    <relationship type="tns:ADTOrderPlacer"/>
        <variableDefinitions>
            <variable name="ADTOrderPlacerC"
                channelType="tns:ADTOrderPlacerChannel"
                roleTypes="tns:RequesterRole tns:ProviderRole">
            </variable>
            <variable name="RAD1"
                informationType="tns:RAD1Type"
                roleTypes="tns:RequesterRole tns:ProviderRole">
            </variable>
            <variable name="RAD3"
                informationType="tns:RAD3Type"
                roleTypes="tns:RequesterRole tns:ProviderRole">
            </variable>
            <variable name="fault"
                informationType="tns:FaultType"
                roleTypes="tns:RequesterRole tns:ProviderRole">
            </variable>
        </variableDefinitions>

        <choice>
                <sequence>
                        <interaction name="Registeration" operation="Register"
                            channelVariable="tns:ADTOrderPlacerC">
                            <participate relationshipType="tns:ADTOrderPlacer"
                                fromRoleTypeRef="tns:ReqRole" toRoleTypeRef="tns:ProRole"/>
                            <exchange name="RegisterationReq" informationType="tns:RegistrationReqType"
                                action="request">
                                <send variable="cdl:getVariable('ID','','')"/>
                                <receive variable="cdl:getVariable('ID','','')"/>
                            </exchange>
                            <exchange name="RegistrationRes" informationType="tns:RegistrationResType"
                                ...
                            </exchange>
                            <exchange name="UpdateResponse" informationType="tns:UpdateResponseType" action="respond">
                                <send variable="cdl:getVariable('UpdateResponse','','')"/>
                                <receive variable="cdl:getVariable('UpdateResponse','','')"/>
                            </exchange>
                            <exchange name="UpdateFault" informationType="tns:UpdateFaultType" action="respond" faultName="InvalidInfoFault">
                                <send variable="cdl:getVariable('faultResponse','','')"/>
                                <receive variable="cdl:getVariable('faultResponse','','')"/>
                            </exchange>
                        </interaction>
                </sequence>
        </choice>
</choreography>
```

Chapter 7: Implications for Healthcare Business Process Modeling

In the previous chapters we developed choreography and orchestration representations of our test case, using BPEL and WS-CDL, and identified advantages and disadvantages of both languages for modeling healthcare processes, presented in Table 12. We found that neither of the languages individually provides the full functionality to satisfy all the healthcare requirements. One of the evaluated languages might be suitable for development of specific models for specific modeling purposes in the healthcare environment. For instance, more graphical notations (BPMN) might be more suitable for developing models intended for training. The suitability of models, or their more corresponding features to some specific purposes, brings the idea of combining modeling languages to have all the features required for a complete model in an environment, especially a complex one like healthcare. As mentioned earlier, it is possible to combine two or more modeling languages to overcome one language's limitations by utilizing the combined advantages of all. The combination of languages provides more features than each individual language. This is what we present in this section.

In order to model a system using two different languages (or methodologies), we first needed to categorize the processes based on their suitability for modeling in each language (or methodology). The following criteria should help in the task.

Table 12: Overall Analysis of Evaluated Languages

	BPMN	BPEL	WS-CDL	Combination
Security & Privacy	-	-	-	-
Pattern Support	17	10	13	15
Ontological Constructs	16	12	16	17
Extendibility	+	+	+	+
Notations	+	+/-	+/-	+/-
Modularity	+	-	+	+
Level of Detail	+/-	+/-	-	+/-
Exception Handling	+	+	+	

- **Intra-departmental or inter-departmental relations:** Intra-departmental processes usually include a tight relationship between processes. There is often a major process which handles and manages the whole department's process. In this case, orchestration would be a better choice since creating standalone processes is difficult and in most cases these processes are dependent on each other. On the other hand, different departments have separate defined tasks and their tasks should not overlap. They usually operate as a standalone unit and there is no central dominant process over their operation. In this case, different departments should be modeled as choreographies.

- **Privacy:** If privacy is an important concern, then orchestrations would not be a good choice. Orchestrated processes share their procedures with the central process which exposes the process to all participant processes while choreography

provides a level of encapsulation for the process so the process remains private at the departmental level.

- **Purpose of modeling:** Considering there is no standard graphical choreography language, if the model is aimed to support training or managerial decisions, BPEL would be a better choice. However, as one of the discussed problems of BPEL is the high level of detail, if the above mentioned issues are the only reason for developing the models, graphical process modeling notations such as BPMN are better.

Once all processes are chosen, modeling can be done in either top-down or bottom-up approaches considering that the upper level will always be choreography. As we mentioned earlier in this section, the resulting model is partially graphical at the departmental level. Because the inter-departmental interactions are not included in each unit's model, models are smaller and less detailed, so BPEL models could be more understandable. Moreover, a combination of choreography and orchestration can solve one of the important concerns in the IT world which is reusability. Reusability is of major importance in complex business sectors such as healthcare due to their dynamicity, frequent change, large extent of process integration, and standardization. By developing reusable services and process models, modeling web service based can be done in a semi-automatic manner.

One of the major issues with BPEL is its lack of modularity. Lack of reusability always accompanies lack of modularity. Since BPEL processes literally cannot be divided into or composed from standalone processes, all its processes should be designed in a linear fashion (Figure 29). As a result, the addition, removal, or alternation of even small sections of processes forces a large change on involved processes. With the help of choreography, here WS-CDL, the BPEL processes can be divided into reasonably smaller parts, and then WS-CDL provides a means of interaction between the parts (Figure 29).

Due to the separation of standalone processes, they are less resistant to change since most alternations will be made on the target BPEL standalone process and its relevant WS-CDL file. Additionally, each standalone process can be used in further modeling without any dramatic changes. As a result, libraries of models will be available to modelers, and the semi-automatic process for defining web services workflows using web service-based composition becomes possible.

In light of all abovementioned benefits coming from the combination of the two methodologies, there are still some remaining drawbacks. The final model has within it each language's embedded problems. The codes are still long and detailed, so they can be used mostly for development purposes. Besides, mastery of two different methodologies is required for model creation and interpretation.

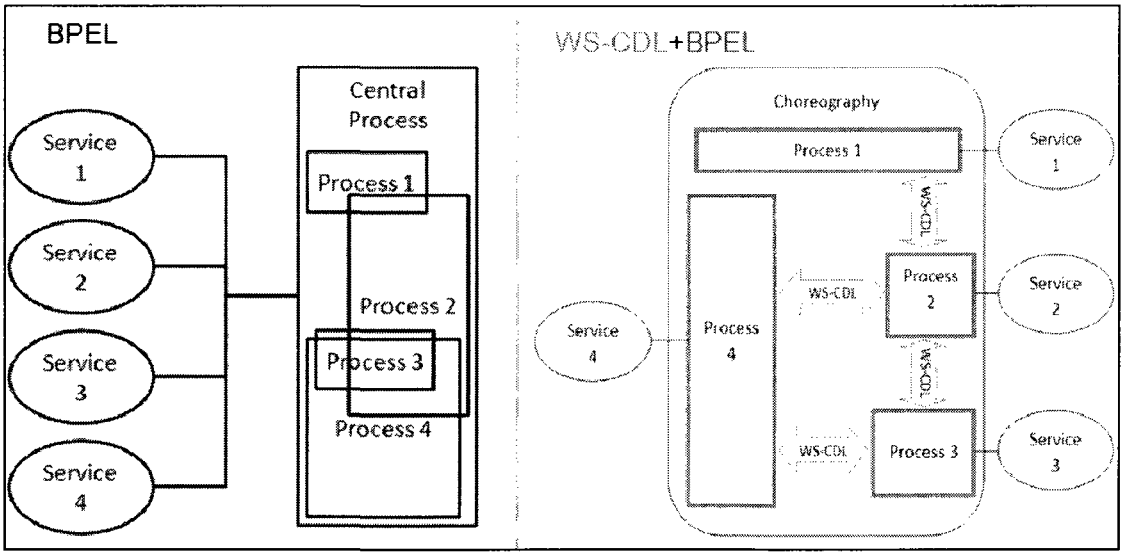


Figure 29: Combination of BPEL and WS-CDL

Chapter 8: Conclusions

Many research initiatives have focused on studying the ambiguities and difficulties of model and modeling language normalization with the aim of improving our ability to design information and communication technologies. Since the specifications of various business sectors and the inherent variety of their processes impose the utilization of a subset of modeling features, modeling languages should be evaluated separately for each sector.

This thesis presents our research on modeling healthcare processes within a service-oriented environment. We took into account the specificities of the healthcare sector as well as its actors in order to enhance business process modeling. We also aimed to improve the design of information and communication technologies to support complex domains such as healthcare.

Healthcare environment is a challenging area for both design and development. The complexity of healthcare processes, the diversity of actors and healthcare's special needs such as security and privacy are the source of those challenges. Although the challenges have been observed by healthcare system designers and developers, there has been a little effort to identify them systematically. We identified challenges associated with modeling healthcare processes as groundwork for identifying the possible solutions to the challenges.

We introduced a framework for evaluating business process modeling languages in healthcare, defined a set of requirements for healthcare process models, and performed a case study on a complex IHE transaction using BPMN, BPEL and WS-CDL. We based our framework on previous works done in different areas of modeling. We found that many previous works focused on one feature of modeling languages, and there is little work on combining all these requirements to form a framework for identifying languages' pluses and minuses all together. It is worth to mention that we developed some criteria, and introduced some others to match the framework more into complex systems. So far, based on studies on languages, our results show that, individually, none of the studied languages satisfies healthcare modeling requirements. Basically, when considering the advantages and disadvantages of each language, any of these languages is useful for different modeling purposes, yet none of them is completely suitable for non-modeler target users. Embedded features of all languages are not enough for modeling complex processes, so extensions to existing languages are necessary in order to generate models that can suit healthcare modeling requirements. However, extensions give rise to more problems in the model development and use.

As another solution to overcome challenges associated with modeling healthcare, combination of languages can be utilized. The combination of modeling languages increases the level of satisfaction for the healthcare modeling requirements, and also solves some problems that result from using a single modeling language. The most important problems solved by combining the two languages are reusability and part of process privacy issues. The final model, created by combination of the two languages, still lacks a complete graphical representation. However, we still recommend it for modeling healthcare processes because it addresses some fundamental healthcare modeling requirements. Healthcare is our domain of focus, but since most complex systems share a set of similar requirements and specifications, findings of this thesis can be applied in other complex domains.

There are a few limitations associated with our research and this thesis. First of all, this thesis, as an exploratory study, focused on identifying the requirements of healthcare business process models, and the features that languages should include to satisfy those requirements. The identified criteria are not ranked on the level of importance as mentioned in chapter 5. Moreover, the metrics developed for the evaluation of criteria are not quantified. On the other hand, each criterion is the object of more attention in modeling for some special purposes. For instance, for training purposes, readability and representativeness become important, while for system development more attention is dedicated to the ability to represent details. As future work, we will evaluate each criterion based on its importance in modeling healthcare environments, and also for different modeling purposes. Based on the evaluation results, the criteria will be ranked and quantitative metrics will be used to evaluate the modeling languages.

In this research we only assessed three languages for modeling healthcare. Although we chose the languages based on their specifications and technology, and we tried to cover most currently used technologies, it is possible that other languages better suit the healthcare environments. We based our research on complex systems and studied healthcare, as one example a complex system. Furthermore, we modeled part of the IHE framework for our evaluation, so the evaluation is limited to the IHE framework. We believe our research will be applicable when evaluating other healthcare frameworks, yet there is a need for more testing and analysis. This work was based on and limited to healthcare processes only. Further, research is also needed to see the extent this research is applicable to other complex environments.

As we have illustrated in this thesis, modeling languages are not complete. They only implement some of the features required in a process model. In our future work we will investigate the integration of modeling languages to overcome their individual limitations, and also combine their advantages. As it was introduced in Chapter 7,

combining two modeling languages can satisfy some of the requirements of complex healthcare processes discussed in this thesis.

References

[1] R. Anzbock and S. Dustdar. (2005, "**Semi-automatic generation of web services and BPEL processes - A model-driven approach**," in *Lecture Notes in Computer Science* Anonymous

[2] R. Anzbock and S. Dustdar, **Modeling and implementing medical Web services**. Data & Knowledge Engineering. vol. 55, pp. 203-236, 2005.

[3] R. Anzbock and S. Dustdar, "**Modeling medical e-services**," in *Lecture Notes in Computer Science - Business Process Modeling* , vol. 3080, Anonymous Heidelberg: Springer Berlin, 2004, pp. 49-65.

[4] R. Anzbock and S. Dustdar, **Modeling and implementing medical e-services**, Technical University of Vienna, Austria, 2004.

[5] R. Anzbock and S. Dustdar, **Medical services workflows with BPEL4WS**, infosys.tuwien.ac.at, Austria, 2003.

[6] A. Barros, M. Dumas and P. Oaks, A Critical Overview of the Web Services Choreography Description Language (WS-CDL). BPTrends. pp. Dec, 2008, 2005.

[7] A. Barros, M. Dumas and A. H. M. Ter Hofstede, "**Service interaction patterns**," in *Lecture Notes in Computer Science - Business Process Management* , vol. 3649, Anonymous Heidelberg: Springer Berlin, 2005, pp. 302-318.

[8] A. Barros and A. H. M. Ter Hofstede. Service interaction patterns: Towards a reference framework for service-based business process interconnection. Queensland University of Technology, Queensland University of Technology, 2005.

[9] Barry & Associates Inc., "**Service oriented architecture (SOA) definition**," 2008, http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html Accessed: 2008.

[10] T. Bellwood, "Understanding UDDI" Accesses: 2009, 07/01. http://www.ibm.com/developerworks/webservices/library/ws-featuddi/, 2002.

[11] A. Bobkowska, Modeling pragmatics for visual modeling language," in TAMODIA 2005. 2005, pp. 75.

[12] BPMI Org. 2009, Business process management initiative. http://www.bpmi.org/, Accessed: 2008

[13] Canada Health Infoway, "Infoway," 2008, www.Infoway.ca, Accessed: 2008.

[14] Capgemini. 2004, Business process modeling defined. *www.uk.**capgemini**.com, Accessed: 2008*

[15] R. Cover, "Business Process Execution Language for Web Services (BPEL4WS)," vol. 2009, July 10. 2008.

[16] R. Cover, "Business Process Modeling Language," vol. 2009, 29/04/2003. 2003.

[17] F. Curbera, R. Khalaf, F. Leymann and S. Weerawarana, Exception handling in the BPEL4WS language," in BPM 2003. 2003, pp. 276.

[18] F. Estrella, T. Hauer, R. McClatchey, M. Odeh, D. Rogulin and T. Solomoniodes. **Experience of engineering grid-based medical software**. International Journal of Medical Informatics vol. 76, no.8, pp. 621-632, 2007.

[19] S. W. Fraser and T. Greenhalgh, Complexity science: Coping with complexity: educating for capability. BMJ. vol. 323, pp. 799-803, 2001.

[20] L. A. Fredlund, Implementing WS-CDL," in JSWEB 2006. 2006

[21] A. Gemino and Y. Wand. A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering vol. 9, no.4, pp. 248-260, 2004.

[22] P. Green, M. Rosemann, M. Indulska and C. Manning, **Candidate interoperability standards: An ontological overlap analysis**. Data & Knowledge Engineering. vol. 62, pp. 274-291, 2007.

[23] P. Green, M. Rosemann and M. Indulska. Ontological evaluation of enterprise systems interoperability using ebXML. Knowledge and Data Engineering vol. 17, no.5, pp. 713-725, 2005.

[24] V. Gruhn and R. Laue, Complexity metrics for business process models," in 9th International Conference on Business Information Systems. 2006, pp. 1.

[25] M. Hafner and R. Breu, Realizing Model Driven Security for Inter-organizational Workflows with WS-CDL and UML 2.0. Model Driven Engineering Languages and Systems. vol. 3713, pp. 39-53, 2005.

[26] Health level seven, inc. http://www.hl7.org/ Accessed: *2008*

[27] Howard Smith and Peter Fingar. Business Process Management: The Third Wave. 2003, Available: www.bpm3.com

[28] IHE Inc., www.IHE.net, Accessed: *2008*

[29] IHE Inc, "IHE Technical Framework, Volume I, Integration Profiles," 2007.

[30] I. Jacobson and S. Bylund, The Road to the Unified Software Development Process. SIGS Reference Library, 2000.

[31] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon and C. Barreto, "Web Services Choreography Description Language Version 1.0," vol. 2008, Nov 9, 2005. 2005.

[32] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. V. Riegen, P. Schmidt and I. Trickovic, "WS-BPEL Extension for Sub-processes – BPEL-SPE," 2005.

[33] M. Laguna and J. Marklund, Business Process Modeling, Simulation, and Design. USA: Pearson Printice Hall, 2004.

[34] R. Lu and S. Sadiq, A survey of comparative business process modeling," in BIS 2007. 2007, pp. 82.

[35] W. Luo and Y. A. Tung, A framework for selecting business process modeling methods. Industrial Management & Data Systems. vol. 99, pp. 312-319, 1999.

[36] J. Mendling and M. Hafner, From WS-CDL Choreography to BPEL Process Orchestration. Journal of Enterprise Information Management. vol. 21, pp. 525-542, 2008.

[37] J. Mendling, H. A. Reigers and J. Cardoso, What makes process models understandable?" in Business Process Management. 2007, pp. 48.

[38] Michael P. Parazoglou and Pieter M.A. Ribbers, E-Business: Organizational and Technical Foundations. England: John Wiley, 2006.pp. 722.

[39] H. Mili, G. B. Jaoude, E. Lefebvre, G. Tremblay and A. Petrenko, Business process modeling languages: Sorting through the alphabet soup, UQAM, 2004.

[40] J. Mykkanen, A. Rikkinen, M. Surmonen, H. Karhunen and P. Laitinen. **Designing web services in health information systems: From process to application level.** International Journal of Medical Informatics vol. 76, no.2-3, pp. 89-95, 2007.

[41] A. Nadalin, C. Kaler, R. Monzillo and B. Hallam-Baker, Web services security: SOAP message security 1.1, WSS: SOAP Message Security (WS-Security 2004), www.OASIS-open.org, 2006.

[42] Y. V. Natis. (2003, 04/16/2003). **Service-oriented architecture scenario**. *2008(05/10)*

[43] NEMA Org., "**DICOM**," http://medical.nema.org/, Accessed: 2008.

[44] A. G. Nysetvold and J. Krogstie, Assessing business processing modeling languages using a generic quality framework," in Workshop on Exploring Modeling Methods in Systems Analysis. 2005, pp. 545.

[45] OASIS Org. 2009, OASIS: Advancing open standards for the global information society. *2009*

[46] R. F. Paigea, J. S. Ostroffa and P. J. Brookeb, Principles for modeling language design. Information and Software Technology. vol. 42, pp. 665-675, 2000.

[47] P. E. Plesk and T. Greenhalgh, Complexity science: The challenge of complexity in health care. BMJ. vol. 323, pp. 625-628, 2001.

[48] P. E. Plesk and T. Wilson, Complexity science: Complexity, leadership, and management in healthcare organizations. BMJ. vol. 323, pp. 746-749, 2001.

[49] M. Rani, A. K. Chawla and S. Batra, Web service choreography description language (WS-CDL): Goals and benefits," in COIT. 2007,

[50] J. Recker and J. Mendling, On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages," in 18th International Conference on Advanced Information Systems Engineering. 2006, pp. 521.

[51] A. Rodriguez, E. Fernandez-Medina and M. Piattini, A BPMN Extension for the Modeling of Security Requirements in Business Processes. IEICE Transactions on Information and Systems. vol. E90-D, pp. 745-752, 2007.

[52] M. Rosemann, J. Recker, M. Indulska and P. Green, A study of the evaluation of the representational capabilities of process modeling grammars," in CAiSE, LNCS. 2006, pp. 447-461.

[53] S. Roser and B. Bauer, A categorization of collaborative business process modeling techniques," in Seventh IEEE International Conference on E-Commerce Technology Workshops. 2005, pp. 43.

[54] D. Rossi and E. Turrini. What your next workflow language should look like, Presented at
2nd InternationalWorkshop on Coordination and Organization.

[55] P. Routino, "Models" http://www2.uiah.fi/projects/metodi/15b.htm, Accessed: 2009, Last modified: Aug 3, 2007

[56] W. Sedera, M. Rosemann and G. Gable, Measuring process modelling success," in 2002, pp. 331.

[57] Sparx Systems. The business process model. Sparx Systems, 2004. Available: www.sparxsystems.com.au

[58] W. M. P. Van Der Aalst. InterorganizationalWorkflows: An approach based on message sequence charts and petri nets. Eindhoven University of Technology, Eindhoven, 1999.

[59] W. M. P. Van Der Aalst, A. K. Alves de Medeiros and A. J. M. M. Weijeiters, "**Process equivalance: Comparing two process models based on observed behavior**," in ***Lecture Notes in Computer Science - Business Process Management*** , vol. 4102, Anonymous Heidelberg: Springer Berlin, 2006, pp. 129-144.

[60] W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede, N. Russell, H. M. W. Verbeek and P. Wohed, Life after BPEL?" in Formal Techniques for Computer Systems and Business Processes. 2005, pp. 35-50.

[61] W. M. P. Van Der Aalst and A. H. M. Ter Hofstede. YAWL: Yet another workflow language. Information Systems vol. 30, no.4, pp. 245-275, 2005.

[62] W. M. P. Van Der Aalst. **Don't go with hte flow: Web services composition standards exposed**. IEEE Intelligent Systems no.Jan/Feb, 2003. Available: http://www.tm.tue.nl/it/research/patterns/ieeewebflow.pdf

[63] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski and A. Barros. Workflow patterns. Distributed and Parallel Databases vol. 14, no.1, pp. 5-51, 2003.

[64] W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede and P. Wohed, **Pattern based analysis of BPML (and WSCI)**, Queensland University of Technology, QUT, 2002.

[65] Y. Wand and R. Weber, On the ontological expressiveness of information systems analysis and design grammars. Information Systems Journal. vol. 3, pp. 217-237, 1993.

[66] W. Wang, H. Ding, J. Dong and C. Ren, A comparison of business process modeling methods," in Service Operations and Logistics, and Informatics. 2006, pp. 1136.

[67] R. Weber, Ontological Foundations of Information Systems. Melbourne, Australia: Coopers & Lybrand and the Accounting Association of Australia and New Zealand, 1997.

[68] S. A. White, Introduction to BPMN, IBM Corp. www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf, Accessed: 15/04/2009

[69] P. Wohed, W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede and N. Russell, "**On the sustainability of BPMN for business process modeling,**" in *Lecture Notes in Computer Science - Business Process Management* , vol. 4102, Anonymous Heidelberg: Springer Berlin, 2006, pp. 161-176.

[70] P. Wohed, W. M. P. Van Der Aalst, M. Dumas and A. H. M. Ter Hofstede, **Pattern based analysis of BPEL4WS**, Queensland University of Technology, QUT, 2002.

[71] Workflow Patterns Initiative. 2007, Workflow patterns. http://www.workflowpatterns.com/ Accessed: *2008(07/24)*

[72] World Wide Web Consortium. 2008, www.W3C.org, Accessed: *2008(06/26)*