HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Business Process Management: State-of-the-Art Technologies

Master's Thesis

Tuomas Piispanen

February 15, 2008

Instructor: Dr. Sasu Tarkoma
Supervisor: Prof. Petri Vuorimaa

HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author
Tuomas Piispanen

Date
Feb 15, 2008

Pages
77

Title of thesis
Business Process Management: State-of-the-Art Technologies

Professorship
Content Production

Professorship Code
T-111

Supervisor
Prof. Petri Vuorimaa

Instructor
Dr. Sasu Tarkoma

Business Process Management (BPM) is today concerning the majority of companies because business processes play such a centric role in modern businesses. BPM is fundamentally a management discipline that aims to improve the performance of enterprise's business processes, and hence the overall performance of the company. By applying the principles of BPM with a sense of purpose, an enterprise can make considerable cost savings and radically improve its competitiveness in the market.

Although BPM has been traditionally considered as a discipline of economics, it is today more and more of a term that is related to information technology, and in particular, to application integration. This can be explained by the fact that in today's enterprises information systems are more often responsible for the execution of individual work tasks of business processes. In modern BPM approaches these information systems are seamlessly integrated with the assist of a state-of-the-art BPM system in order to reach the goal of BPM - the improved process performance. In a modern approach IT tools play also a centric role in carrying out the other relevant activities of BPM including business process modeling, monitoring, analysis, and optimization.

It is notable that especially in the 00s BPM has gained a lot of new momentum as new kind of methods and technologies have emerged. In particular, the emergence of XML and Web Service standards have had a significant effect on the development of BPM-related information technology.

The purpose of this thesis is to provide an overview of the latest technologies and methods of (computer-aided) BPM.

Keywords
BPM, business process modeling, process automation, BPM systems, business intelligence, Web Services, Web Service composition, SOA, B2Bi, BPMS, WfMS, ESB, BAM, BPEL, BPMN, UML, EAI, Netbeans, J2EE, Questioner

TEKNILLINEN KORKEAKOULU Informaatio- ja luonnontieteiden tiedekunta Tietotekniikan laitos	DIPLOMITYÖN TIIVISTELMÄ

Tekijä Tuomas Piispanen	**Päiväys** 15.2.2008
	Sivumäärä 77

Työn nimi
Liiketoimintaprosessien johtaminen – katsaus uusimpiin teknologioihin

Professuuri Sisällöntuotanto	**Koodi** T-111

Työn valvoja
Prof. Petri Vuorimaa

Työn ohjaaja
TkT Sasu Tarkoma

Liiketoimintaprosessien johtaminen koskee tänä päivänä lähes kaikkia yrityksiä, koska prosessit ovat yhä tärkeämmässä roolissa osana lähes mitä tahansa nykyaikaista liiketoimintaa. Liiketoimintaprosessien johtaminen (lyh. prosessijohtaminen) tarkoittaa lyhyesti menetelmäoppia ja johtamisstrategiaa, jonka avulla pyritään tehostamaan yrityksen prosesseja ja sitä kautta koko yrityksen toimintaa. Määrätietoisella prosessijohtamisella voidaan saavuttaa yrityksissä huomattavia kustannussäästöjä ja kilpailuetua markkinoilla.

Liiketoimintaprosessien johtaminen ei ole uusi ilmiö, vaan sen johtamisoppeja on sovellettu jo 1900-luvun alkupuolelta lähtien, aina siitä asti kun prosessiajattelu alkoi saamaan jalansijaa yrityksissä. Vaikka perinteisesti liiketoimintaprosessien johtaminen tunnetaan paremmin käsitteenä kauppatieteistä, on siitä tullut viime vuosikymmeninä yhä enemmän tietotekniikkaan ja prosessiautomaatioon liittyvä termi. Tämä johtuu siitä, että liiketoimintaprosesseihin liittyy yhä enemmän tietojärjestelmiä, jotka vastaavat yksittäisten työvaiheiden suorittamisesta prosesseissa. Moderniin prosessijohtamisen lähestymistapaan kuuluu integraatiojärjestelmien käyttöönottoa, joiden avulla yksittäiset tietojärjestelmät voidaan integroida suuremmiksi yksiköiksi suorittamaan suurempia tehtäväkokonaisuuksia – eli yrityksen liiketoimintaprosesseja. Tietotekniikalla on keskeinen rooli modernissa prosessijohtamisessa prosessien suorittamisen lisäksi myös niiden mallinnuksessa, valvonnassa, analysoinissa ja optimoinnissa.

On huomattavaa, että erityisesti 2000-luvulla prosessijohtaminen on saannut paljon uutta pontta uusien menetelmien ja teknologioiden myötä. Erityisesti avoimet XML- ja Web-palvelu -perustaiset tekniikat ovat myötävaikuttaneet merkittävästi prosessijohtamiseen liittyvän tietotekniikan kehittymiseen.

Tämän työn tarkoitus on tutkia modernin (tietokoneavusteisen) prosessijohtamisen soveltamisessa käytettäviä menetelmiä ja uusimpia teknologioita.

Avainsanat
prosessijohtaminen, mallinnus, prosessiautomaatio ja -järjestelmät, Web-palvelut, Web-palveluiden koostaminen, SOA, EAI, B2Bi, BPMS, WfMS, ESB, BAM, BPEL, BPMN, UML, Netbeans, J2EE, Questioner

Acknowledgments

First of all, I would like to thank HIIT (Helsinki Institute of Information Technology) and particularly my supervisor Mr. Sasu Tarkoma for kindly permitting me to undertake this project in Finland's top information technology research institute, HIIT. This has been really a great opportunity for me.

I am really grateful for all people who have been supporting me or providing their valuable feedback or comments on the way. I want to thank for this especially Mr. Sasu Tarkoma but also Mr. Ken Rimey and the rest of the HIIT Services4All research group members. I also thank Mr. Petri Vuorimaa for acting as a supervisor for this thesis. Of course I want to also thank my parents and all the coffee producers for satisfying my endless thirst for coffee. Writing this thesis would have been very difficult without coffee – even impossible.

Preface

This thesis has been the last milestone on the way completing my computer science studies in the Department of Computer Science and Engineering, Helsinki University of Technology, Finland. During my studies I have been focused on topics such as Internet technologies, multimedia, content production, information systems, and enterprise application integration.

I have written this thesis during a 12 months period in HIIT (Helsinki Institute of Information Technology) between Dec 2006 and Feb 2008. The research of this thesis has been part of a bigger research project called Services for All (S4All). The S4All is an international research project that has been on-going since 2006 and is about to finish in 2008. The project has been coordinated by ITEA (International Technology Education Association). The goal of the project has been “a world of user-centric Web Services that are easy to create, share and use”. This thesis has been a small contribution from me to this large-scale project.

Table of Contents

Abstract.....i

Tiivistelmä.....ii

Acknowledgments.....iii

Preface.....iii

Table of Contents.....iv

Figures Index.....vi

Tables Index.....vi

List of Abbreviations.....vii

1 Introduction.....1

1.1 Business Process Management.....1
1.2 Research Objectives.....2
1.3 Previous Research.....3
1.4 Structure.....3
1.5 Experimental Part: Questioner Application.....4

2 Evolution of Application Integration Technologies.....5

2.1 Enterprise Application Integration (EAI).....5
2.2 Evolution of Middleware.....7
2.3 Web Services.....9
2.4 Web Service Composition.....11
2.5 Web Service Composition Languages.....13
2.6 Service Oriented Architecture (SOA).....14

3 Business Process Management Essentials.....16

3.1 Evolution of BPM.....16
3.2 Foundational Technology.....17
3.3 Architecture.....18
3.4 Business Process Lifecycle.....19
3.5 Workflow Management.....20
3.6 BPM – A Convergence of EAI and Workflow Management.....21
3.7 Standardization.....21

4 Business Process Modeling.....25

4.1 Overview.....25
4.2 Diagramming and Other Modeling Techniques.....25
4.3 Business Process Modeling Languages.....26
4.4 Granularity of Process Models.....29
4.5 Modeling Processes with BPMN.....31
4.6 Modeling Processes with UML Activity Diagrams.....32
4.7 Differences Between BPMN and UML Activity Diagrams.....34
4.8 File Formats for Process Models.....34

5 Business Process Execution Language (BPEL).....37

5.1 Overview.....37
5.2 Technology.....38
5.3 Features.....39
5.4 How to Develop a BPEL Process.....41
5.5 BPEL Editors.....46
5.6 Shortcomings.....48

5.7 Extensions....49
6 Business Process Run-Time Software and Tools....50
6.1 Workflow Management Systems (WfMS)....50
6.2 Business Process Management Systems (BPMS)....52
6.3 Enterprise Service Bus (ESB)....53
6.4 Business Activity Monitoring Tools (BAM)....57
6.5 BPEL Engines....58
6.6 Open Source BPM Platform: Netbeans 5.5....60
7 Advanced Topics....62
7.1 BPMN-to-BPEL Mapping....62
7.2 B2B Automation: Introducing Choreographies....62
7.3 Combining Choreographies and Orchestration....63
8 Experimental Part: Questioner Application....65
8.1 Overview....65
8.2 Original Intended Use....66
8.3 Other Uses....66
8.4 Features....68
8.5 Server-side Implementation....69
8.6 Client-side Implementation....71
8.7 Discussion....73
9 Conclusions....75
9.1 Summary....75
9.2 Observations....75
9.3 Outlook on BPM....76
9.4 Further Research....76
References....78
Appendix A....83
Appendix B....89
Appendix C....90
Appendix D....92

Figures Index

Figure 1: BPM, enterprise strategy, processes, IT, and people....2
Figure 2: Legacy nightmare....6
Figure 3: Point-to-point and hub-and-spoke integration architecture....6
Figure 4: Evolution of middleware....7
Figure 5: Service interaction patterns....9
Figure 6: WS protocol stack and composition layer....11
Figure 7: Web Service orchestration....12
Figure 8: A Web Service choreography....12
Figure 9: Layers of SOA....15
Figure 10: Architecture of BPM....19
Figure 11: Business process lifecycle....19
Figure 12: Evolution of BPM standards....24
Figure 13: Granularity of a process model....29
Figure 14: Granularity of a modeling language....30
Figure 15: BPMN core elements....31
Figure 16: Booking a trip....32
Figure 17: UML Activity Diagram elements....33
Figure 18: Order-delivery process....33
Figure 19: BPEL process and the corresponding workflow diagram....38
Figure 20: Currency conversion process....41
Figure 21: Skeletal currency conversion BPEL process....42
Figure 22: WSDL of the currency conversion process....43
Figure 23: Completed currency conversion BPEL process....46
Figure 24: Oracle JDeveloper BPEL Designer....47
Figure 25: Hub-and-spoke and bus architecture....54
Figure 26: Architecture of ESB and itinerary-based routing....55
Figure 27: IBM Websphere Business Monitor....57
Figure 28: Oracle Business Process Manager administration console....59
Figure 29: Developing BPEL processes with Netbeans....60
Figure 30: Collaborating in a choreography....64
Figure 31: Questioner components....65
Figure 32: Questioner with an automated business process....66
Figure 33: Organizing events with Questioner....67
Figure 34: Opinion polling with Questioner....68
Figure 35: Questioner architecture....70
Figure 36: Questioner client program....71
Figure 37: Question message formats....73
Figure 38: Developing the Questioner mobile client program....73

Tables Index

Table 1: Summary of common BPM standards....22
Table 2: Classification of modeling languages....25
Table 3: BPEL activities....44
Table 4: BPEL editors....47
Table 5: BPEL engines....58

List of Abbreviations

API	Application Programming Interface
B2B	Business-to-Business
B2Bi	Business-to-Business integration
BAM	Business Activity Monitoring
BPDM	Business Process Definition Metamodel
BPEL	Business Process Execution Language for Web Services (also WS-BPEL, BPEL4WS)
BPELJ	BPEL for Java
BPM	Business Process Management / Business Process Modeling
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Modeling Notation
BPMS	Business Process Management System
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
EAI	Enterprise Application Integration
EJB	Enterprise Java Bean
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
HTTP	Hypertext Transport Protocol
IDE	Integrated Development Environment
J2EE	Java 2 Enterprise Edition
LRT	Long-Running Transaction
MOM	Message-oriented Middleware
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
RPC	Remote Procedure Call
SMS	Short Message Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
WS-	Web Service-
WS-CDL	Web Services Choreography Description Language
WSDL	Web Service Description Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XPath	XML Path Language
XPDL	XML Process Definition Language

1 Introduction

In the first Chapter of this thesis we introduce the concept of Business Process Management. This chapter also discusses the goals and structure of this thesis.

1.1 Business Process Management

Business Process Management (BPM) is an old management discipline which goal is to improve the performance of enterprise's business processes. The concept of BPM is actually quite difficult to define precisely because there are so many varying definitions for it. In the following is listed a few alternative definitions:

- **Whatis.com:** *"a systematic approach to improving an organization's business processes. BPM activities seek to make business processes more effective, more efficient, and more capable of adapting to an ever-changing environment."*
- **Webopedia.com:** *"describes activities and (or) events which are performed to optimize a business process. These activities are aided by software tools."*
- **Gartner:** *"a management practice that provides for governance of a business's process environment toward the goal of improving agility and operational performance. BPM is a structured approach employing methods, policies, metrics, management practices and software tools to manage and continuously optimize an organization's activities and processes."*

At the core of BPM is the business process. A business process may be defined (according to whatis.com) as "a set of coordinated tasks and activities, conducted by both people and machines, that will lead to accomplishing a specific organizational goal. .. They occur at all levels of an organization and can also occur between organizations. .. Business processes may not always be clearly recognizable in organizations. .. Typical business processes include such as receiving orders, ordering materials, manufacturing products, shipping products, invoicing customers, and processing claims."

It is often said that the adoption of BPM principles is what separates the good companies from the rest of the pack. By applying the principles of BPM with a sense of purpose, an enterprise can gain some considerable business benefits. In the short-term, BPM helps companies improve profitability by decreasing costs and increasing revenues. In the long-term the benefits are even bigger as BPM enables competitive advantage by improving organizational agility. The organizational agility means that an organization is better equipped to respond to change – the only thing that is constant in business. [1][2]

BPM is not a new discipline because its principles have been applied basically ever since the early 20th century. It is generally thought that the systematic management of business processes started in the 1920s in the contemporary companies. This was a consequence of companies adopting the Frederick Taylor's famous theory of scientific management, also known as Taylorism. It is a theory that laid down the fundamental principles of large-scale manufacturing through assembly-line factories. [3]

Although traditionally BPM has been considered as a discipline of economics, today it is also a term that is closely related to information technology. This can be explained by the fact that in many companies the information systems are responsible for executing the bulk of the tasks of which business processes are composed of. To improve the performance of such processes, new information technology is needed in an enterprise to seamlessly integrate these systems together. In today's BPM approaches, IT tools are used also to model, monitor, measure, analyze, and optimize business processes.

Figure 1 depicts the relationship of BPM and IT. In modern organizations the enterprise strategy is driven through processes, and processes further driven through IT. The figure also depicts how the BPM is a collaborative effort by both business and IT people in today's organizations.

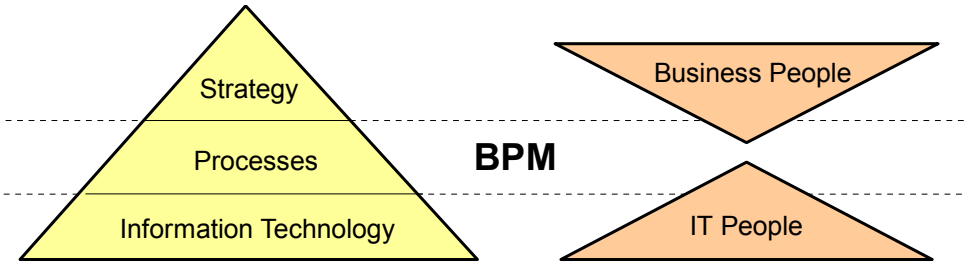


Figure 1: BPM, enterprise strategy, processes, IT, and people

It is notable that especially in the few last years BPM has gained much new momentum as new kind of methods and technologies have emerged. Particularly, the emergence of Web Service standards have had a significant effect on the development of BPM technologies. Web Services offer BPM an Internet-enabled application integration platform that is based on open standards. This has meant that today lesser BPM solutions are based on proprietary technologies. The Web Services have further enabled to develop many new BPM standards that build upon them. [1][4]

To apply BPM in practice, the state-of-the-art solution is to introduce a new kind of enterprise information system – a BPM system. BPM systems are fundamentally process-centric application integration platforms but unlike the traditional EAI systems that are based on proprietary technologies, BPM systems build upon open standards such as XML, Web Services, and BPEL. [1][4]

1.2 Research Objectives

The main objective of this thesis is to get a big picture of what is currently going on in the arena of BPM. To get into this, we have to present a number of different concepts, methodologies, and technologies of BPM on the way. We are going to study which of the methodologies and technologies offer today the best combination.

The second main object of this thesis is to find out how to automate a complete business process in practice. We want to know what are the exact steps that needs to be taken each time a business process is automated. The steps include some important activities such as business process modeling and execution.

One of the specific objectives of this thesis is to examine the relationship between Web Service composition and BPM. We are going to be especially interested in the BPEL composition language because it represents today the de facto language to specify executable business processes in enterprises.

Furthermore, it should be noted that we are also interested in bringing out all kind of new ideas related to BPM. This means if we encounter some useful new ideas in the white papers or articles we are going to read, we will discuss them here.

Below is listed some secondary objectives of this thesis:

- **Standardization -** The multitude of the BPM standards is often surrounded by many misunderstandings and ambiguity and finding good sources for information is difficult. We are going to discuss what BPM standards are used today, what standardization organizations are out there, and what roles do they have.
- **Buzzwords -** What is meant by the cryptic three- and four-letter buzzwords such as SOA, EAI, B2Bi, BPMS, WfMS, ESB, BAM, BPEL, BPMN, UML, EAI, J2EE?

- **Historical aspects** - How has BPM changed and how has it been applied in practice during the years?
- **BPM products** - What kind of BPM-enabled information systems may be utilized in enterprises to automate business processes?
- **Open technologies**: - The technologies that are either based on open standards or are available as Open Source or freeware are of special interest of this thesis.

1.3 Previous Research

Many books have been written about the BPM during the years. However, the problem is that most of the books have been written purely from an economics perspective. Only a few books also deals with the IT side of BPM. Many of the IT-related books have further become outdated as new technologies have emerged. The life expectancy of such a book is usually quite short.

One of the most important books in this area is essentially the Smith and Fingar's "Business Process Management: 3rd Wave" [1]. It has been very often cited from several BPM-related white papers and magazine articles. Some other important books include Business Process Management: A Rigorous Approach by M. Ould [5], Business Process Management Systems: Strategy and Implementation by J. Chang [6], Essential Business Process Modeling [7] by M. Havey, Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology by Mark and Bell [8], and Process Aware Information Systems by M. Dumas et al. [9]. All of these books have been written by authors with some substantial merits in BPM.

In addition, there are a few good Web journals, magazines and blogs that are good sources for up-to-date information of BPM technologies. Below we have listed some of our favorites:

- Align Journal – www.align.com
- SOAWorld Magazine – soa.sys-con.com
- Techtarget - searchsoa.techtarget.com
- ebizQ guide – www.ebizq.net
- Column2 (Sandy Kemsley's blog) – www.column2.com
- BPMS Watch (Bruce Silver's blog) – www.brsilver.com/wordpress/
- Perspective in Process (Phil Gilbert's blog) – blog.lombardicto.com
- Go Flow (Keith Swenson's blog) – kswenson.wordpress.com

1.4 Structure

Chapter 2 discusses the evolution of application integration technologies. Application integration technologies serve as the foundation for the today's BPM technologies. The chapter discusses, inter alia, Web Services, middleware, EAI, SOA, Web Service composition, and workflow automation. Chapter 3 discusses some of the essential theoretical and technical concepts of BPM. Chapter 4 discusses business process modeling and the different aspects of it. Several different ways and languages are presented to address the challenge of business process modeling. Chapter 5 is focused merely on Business Process Execution Language (BPEL). Chapter 6 discusses some state-of-the-art business process run-time software and tools that may be used to apply BPM in today's enterprises. Chapter 7 discusses some miscellaneous advanced topics of BPM that have arisen in the previous chapters. Chapter 8 is focused on the experimental part of this thesis – the Questioner application which is fundamentally a reusable Web Service component for automated business processes enabling human interactions. Chapter 9 concludes the thesis, gives some finishing thoughts, and provides a short outlook on BPM.

1.5 Experimental Part: Questioner Application

A very pivotal part of this thesis is the Questioner application that we have built for the purpose of the experimental part of this thesis. The Questioner application is fundamentally a specialized SMS gateway to send question-formatted messages to a mobile phone. It has been intended as a reusable human interaction component for automated business processes. The idea is that an automated business process can send simple yes/no type of questions, at run-time, to people who are responsible for the process, e.g., to managers, administrators, or operators. The question recipients are allowed to reply to the question only with predefined answer choices. For example, Questioner can be used in an automated business process to query an administrator for details of how to recover from error conditions. Questioner application is fundamentally a client/server application consisting of a J2EE/BPEL server and a simple J2ME application in the user's phone. For more details, see Chapter 8.

2 Evolution of Application Integration Technologies

In this chapter we discuss the Enterprise Application Integration (EAI) which is a concept closely related to BPM. We first discuss the concept of EAI and then focus on the different technologies of it. We are especially interested in the Web Service technologies because they represent today the core technology of BPM. In this chapter we also present the important concept of Service-Oriented Architecture.

2.1 Enterprise Application Integration (EAI)

Enterprise Application Integration (EAI) refers to the problem of integrating applications or information systems within an enterprise. EAI is fundamentally not a technology per se, but a collection of tools, techniques and technology. Below is listed a few definitions for EAI:

- **Thefreedictionary.com** – *"The use of middleware to integrate the application programs, databases, and legacy systems involved in an organization's critical business processes."*
- **Techtarget.com** – *"A business computing term for the plans, methods, and tools aimed at integrating, modernizing, consolidating, and coordinating the computer applications in an enterprise. The main goal of EAI is to integrate the heterogeneous enterprise application infrastructure to act like a single enterprise-wide system that is better capable of serving the enterprise's business needs."*

The need for EAI has emerged whereas the number of applications in enterprises has risen dramatically in the past decades. A today's enterprise can consist of dozens or even hundreds of applications of many generations and from each of which is having its important role in the enterprise's business activities. The problem is that the applications in an enterprise often take the form of islands of automation. This means the applications are unable to communicate easily with each other due to partial or full isolation.

In order that an enterprise can take the full value of its applications, the applications must be integrated following a structured EAI approach. If applications are not integrated following a structured approach but rather on an ad-hoc basis, point-to-point connections grow across an organization. Dependencies are added on an impromptu basis, resulting in a tangled mess that is difficult to maintain. This is commonly referred to as spaghetti, an allusion to the programming equivalent of spaghetti code. [10][11]

Integrating applications in enterprises is often a very difficult task. The main reason for the difficulty is usually the notorious "legacy system problem". A legacy system is fundamentally an antiquated computer system in an enterprise that does not interface with new technologies and of which there is a general lack of understanding in the enterprise. Integrating such systems is usually very slow and expensive because any supportive resources cannot be found anymore. In today's enterprises there can be dozens of such legacy systems. In Figure 2 is shown a situation from which an enterprise may have to start its integration project. [10][11]

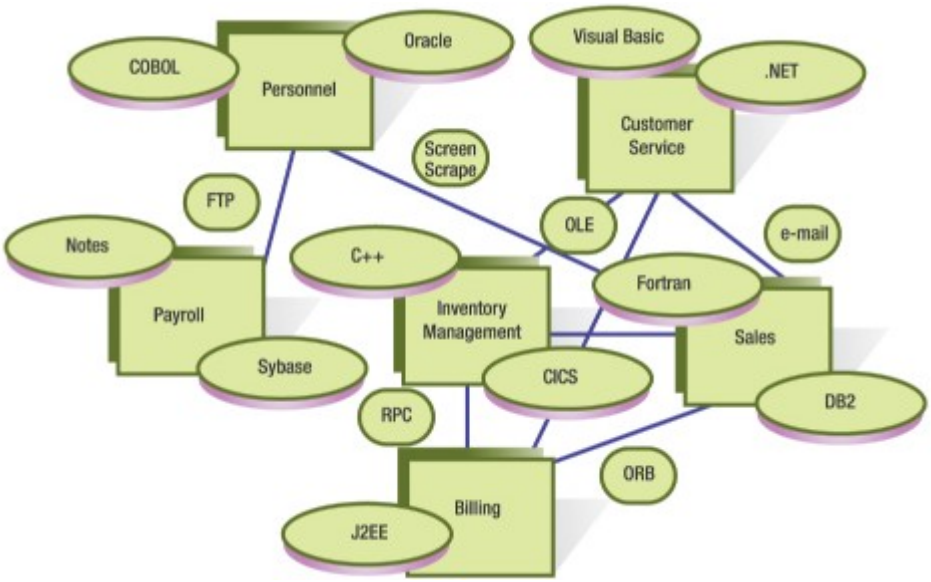


Figure 2: Legacy nightmare

EAI is applied in practice by introducing a so called EAI platform. The main goal of EAI platforms is to integrate the heterogeneous enterprise application infrastructure to act like a single enterprise-wide system in which applications and data are shared across the organization. The platforms can serve the enterprise's business needs and enable to build business processes upon it. In order to apply BPM efficiently in an enterprise, some sort of a EAI platform is basically always needed. [11]

The today's EAI platforms include application servers, EAI systems, and modern process-centric systems such as BPMSs or ESBs. All of these solutions are fundamentally hub-and-spoke architectures consisting of a centralized hub (or broker) in which the applications are connected as spokes. The spokes are connected through connectors that are constructed and deployed on top of existing applications. Connectors are needed to address the heterogeneity of applications, or to bridge differences in protocols and message types. In Figure is shown the fundamental difference between a point-to-point (ad hoc) and hub-and-spoke integration architectures. [12]

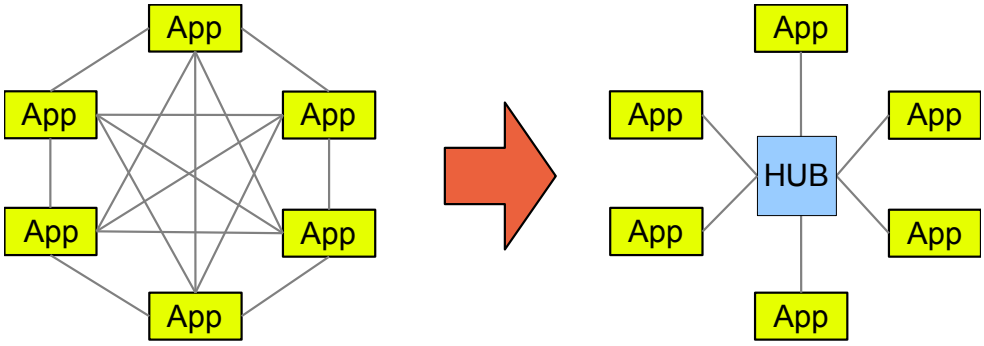


Figure 3: Point-to-point and hub-and-spoke integration architecture

B2Bi (business-to-business integration) is a term that is closely related to EAI. It is about application integration between enterprises – unlike EAI which is application integration within an enterprise. A need for interenterprise application integration may arise for example in between trading partners or supply chain participants. Today, B2Bi is however often seen as only extending EAI because the traditional enterprise boundaries are blurring at the time of business outsourcing, joint ventures, virtual corporations, and others. The merger of EAI and B2Bi is also much because of Web Services which have become the standard middleware in both domains.

2.2 Evolution of Middleware

In computer industry, the term middleware refers to any technology that ties two or more applications together and enables them to talk to each other using same language. While EAI is more of a concept or a comprehensive solution for application integration, middleware is the technical component of it that "glues" the connected applications together and realizes the integration. The used middleware does not have to be necessarily a piece of software but it can be also some simpler means such as an e-mail or a shared file.

Very many kind of different middlewares have been developed and used in the past few decades. Today the middlewares that are being used in enterprises are often very complicated software systems. The complicacy is largely due to the large number of the applications the middleware has to integrate in enterprises. It can be also explained by that today the middlewares are often also required to provide a bunch of additional features to support advanced transactions, data transformations, routing, etc. [13]

Figure 4 illustrates the time spans of the (software-based) middlewares that have been used in the past decades, their scope, and what kind of software building blocks they are capable of integrating [13]. The middlewares have been further explained in the list below (according to [13], [14]). The middlewares are in a somewhat chronological order in the list.

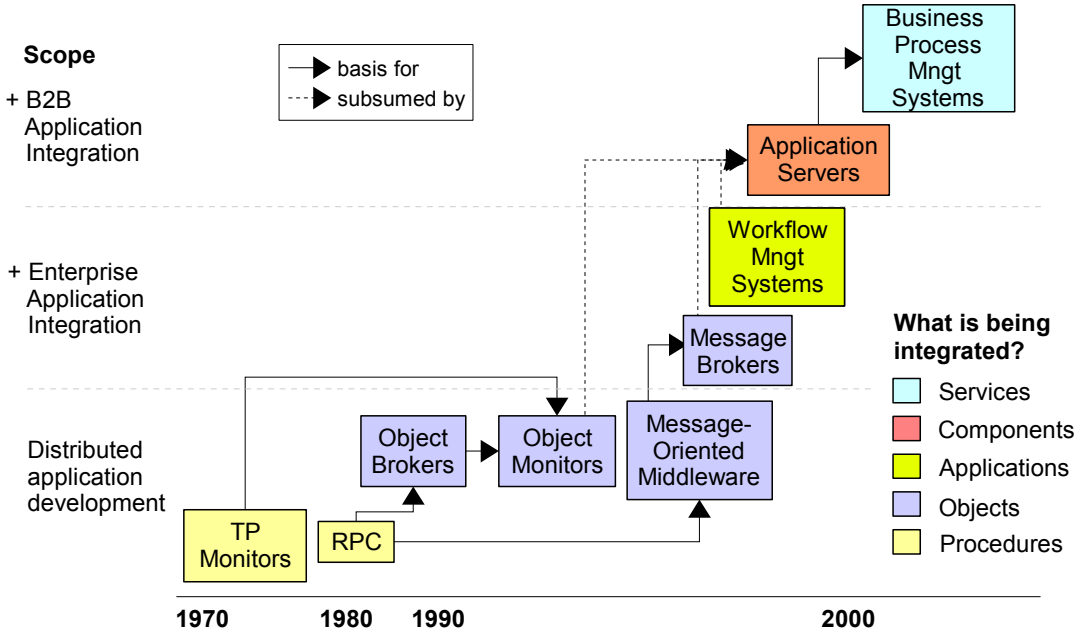


Figure 4: Evolution of middleware

Shared file – A shared file is a file in a network location that is accessible by two or more applications. The applications share data by reading and writing this shared file. The use of a shared file is very limited but often sufficient for the simplest integration cases. In practice, the shared file is most often a simple text file but it may be also a binary file. The data of the text files can be structured by such means as comma delimitation or XML.

Screen scraping – Screen scraping is about "reading" data electronically from application screens. Screen scraping software has been used already in the 1960s with mainframe computers. Although screen-scraping represents a very rudimentary way of application integration, it has been used very successfully to integrate applications and mainframes in the past.

Database connectivity – In this middleware method two or more applications are sharing a database from/to which they can both read and write. The idea is fundamentally the same as with the shared file methods. Use of this method is however not really recommended because it leads to "tight couplings" between applications and the database. Such couplings are difficult handle in the long run and often leads to legacy system type of problems.

APIs – An Application Programming Interface (APIs) is a programmatic interface to an application that enables programmers to control the application or to access the information stored into it. APIs are usually application- and programming language specific.

E-mail – Standard e-mail is often used as a middleware because very many applications can relatively easily send and receive e-mails. This method is best-suited only for some of the more simple cases.

Distributed objects – Object technologies were developed in the first place to address the limitations of APIs. Integrating applications using distributed objects is often considered easier because it requires less low-level programming. The interfaces, data types, and invocations are somewhat more standardized in distributed objects than in APIs.

RPC/RMI-based systems – Remote procedure call is a traditional middleware technology that allows an application to call a function (procedure) across machine boundaries without the programmer needing to code the low-level details of transactions. The original idea of RPC was to enable a programmer to call remote procedures as easily as a local procedures. RMI (Remote Method Invocation) is an object-oriented equivalent for RPCs.

TP Monitors – Transaction processing monitors represent one of the oldest and best-known form of middleware. TP Monitors are pieces of software (with databases) that enable transactions with RPCs. TP Monitors have been used traditionally to assure the transactional integrity in distributed database environments. They have been also used often to perform load balancing between distributed servers.

Object Brokers – Object Request Brokers (ORB) is called a software that provides transparent interaction mechanisms between distributed objects. The objects of a distributed object environment can be located and invoked via an ORB. ORBs allow objects to communicate with various protocols and to reside in different platforms. The most popular class of object brokers are those based on the CORBA architecture.

Object Monitors – Object Monitors are fundamentally TP monitors extended with object-oriented interfaces (merged to object brokers).

Message-Oriented Middleware (MOM) – MOM is a middleware that is based on asynchronous messaging (not on synchronous procedure calls anymore). The core components of MOM systems are message queues to which the applications are enabled to subscribe. The main benefit of MOM is that the integratable applications do not have to be necessarily in the same LAN but can be within reach of slower and less reliable connections.

Message brokers – Message brokers are specialized MOMs that are also capable of transforming, routing and filtering messages as they move through queues.

Workflow management systems (WfMS) – Workflow management systems are based on message brokers but differ in that the integration logic is not hard-coded into applications but separated from them. In WfMSs, the separated integration logic is represented in the form of a graphical workflow diagram. The main benefit of using diagrams is that they enable to make changes to the logic more easily.

Application servers – Today's application servers are quite similar to plain old TP monitors. Both of them are centralized transactional middlewares and are focused on scalability and fault tolerance. The differences to TP monitors include that they are Web-enabled, they are based on more modern programming languages (J2EE/.NET instead of C/COBOL), they provide a large set of application connectors, and that they are based on component models.

Web Services – Web Services is an XML-based middleware to integrate applications over Internet. Although Web Services have been quite much hyped, the idea is still the very same as it was already with RPC/CORBA a long time ago. Web Services are discussed in more depth later in this chapter.

2.3 Web Services

Web Services have become the most relevant middleware platform for enterprise application integration in this decade. The platform is based on the notion of service. A service represents fundamentally "a unit of work that an application can do on behalf of another application". [15]

Web Services is an "Internet-enabled" middleware which means it can be applied for both EAI (Enterprise Application Integration) and B2Bi (Business-to-Business integration), and in LANs and WANs (Internet), respectively.

The term Web service is essentially only an abstract notion that refers to a combination of an application and a service wrapper. A service wrapper is a piece of software that wraps its host application into a service. The "wrapping" refers to transforming the native outgoing messages of the application into a standard Web Service format (SOAP/XML). Correspondingly, the "unwrapping" refers to transforming the incoming Web Service-formatted messages into the application's native message format. A service wrapper are built using the same programming language in which the application has been implemented.

Web Services are today so widely used mainly because they are based on open and royalty-free standards. Because of the XML-based standards, programmers are enabled to easily consume Web Services using several programming languages. Another benefit of XML-based standards is that they are also neutral about the underlying application platforms (J2EE/.NET) and operating systems. [11]

The basic technology of Web Services is relying on two standards: WSDL (Web Service Description Language) and SOAP (Simple Object Access Protocol). WSDL description language is used to describe the public interfaces of Web Services. A WSDL description of a Web Service enables a programmer to gain a clear understanding of its purpose and of how one is enabled to invoke it. SOAP protocol's role is instead to act as a standard communication protocol of Web Services. SOAP messaging is basically about sending XML over HTTP.

Figure 5 depicts the Web Services model, or the interaction patterns of how connect to a Web service, suggested by the W3C's Web Service specification. Figure 5a represents the most usual pattern in which a service consumer connects directly to a service provider. Figure 5b represents a situation in which a service directory is also involved. The role of the service directory is to simply connect the service consumers and providers. Through a service directory the service consumers are able to find new services and to which the service providers are enabled to publish descriptions of their services. In practice, service directories are today still quite rarely used. [15]

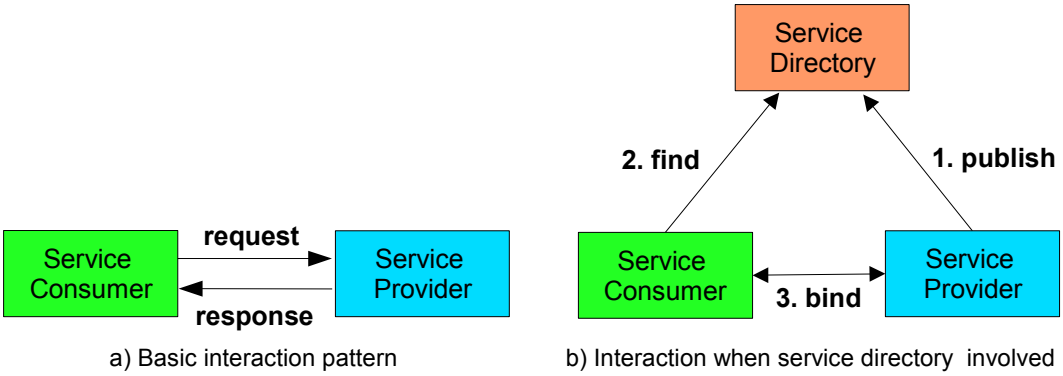


Figure 5: Service interaction patterns

2.3.1 Core Standards

WSDL (Web Services Description Language) is an XML-formatted language by W3C for describing Web Service interfaces in a standardized way. Each Web Service is required to have its respective WSDL description. In a WSDL document is described only the public interface of the service. A WSDL document enables a Web Service consumer to gain a clear understanding of the service's purpose and of the technical details of how to invoke it. The type system of WSDL is based on XML Schema. [15]

Each WSDL file contain at least following information: what operations the service provides, what kind of message formats (message structure and data types) must be used in order to invoke those operations, what kind of response messages the service may send as a reply, which transport protocol is used (binding information), and where is the service located in the network (URI endpoint). It is notable that the WSDL does not have to reside necessarily on the same network address as the actual service endpoint.

SOAP (Simple Object Access Protocol) is the standard communication protocol of Web Services. SOAP messaging is fundamentally about sending XML documents over the common Internet protocol – HTTP. Each SOAP message captures a complete XML document inside it which contains the actual data. SOAP messages are in general very simple and they may be easily read with the human eye. [15]

In technical level, SOAP messages are carried as payloads of HTTP/POST requests. SOAP messages can be bound also to protocols other than HTTP, for example, to SMTP or FTP. A SOAP message, or in SOAP terms – an envelope, consists of a body and header. The header part of the message is optional and it is actually quite rarely used. In the header part it is possible to specify some advanced routing, security, or authentication mechanisms.

The biggest advantage of SOAP over former middleware protocols is that it can penetrate firewalls as SOAP messages are sent through the standard HTTP port (80). Today the traditional middleware protocols, e.g., RPC, get easily stuck to the company firewalls because they are using protocol-specific ports.

UDDI (Universal Description, Discovery and Integration) is a set of specifications to define a Web Service directory. UDDI directories are databases consisting of information about Web Services. UDDIs themselves are also exposed as Web Services and thus, they may be accessed through a SOAP protocol and they have their respective WSDLs. The main goal of UDDI is to simply enable service consumers to find service providers and, on the contrary, to allow service providers to advertise their services to the consumers. As UDDIs include also the WSDLs of the services, the service consumers are enabled to immediately bind to the services they find from the UDDI. [15]

Two kind of UDDI directories are possible: public registries available in Internet for everybody; and private directories in private networks – available for a limited number of users only. While the goal of public UDDIs is to enable B2B, the private UDDIs act more as catalogs of services in enterprises. Public UDDIs are today somewhat rare. [16]

UDDI directories are often compared to telephone book's white, yellow and green pages: white pages consist of information about the companies having their services in the registry; yellow pages lists all the Web Services in the registry classified into categories; green pages provide technical details of the services. [17]

2.4 Web Service Composition

Web Service composition is fundamentally what EAI is today. WS composition is about composing Web Services into new (composite) Web Services, or in EAI terms, creating new enterprise applications by integrating the existing ones. A prerequisite for WS composition is that the most of enterprise's applications have been exposed as Web Services. In other words, the IT architecture of an enterprise should be SOA-enabled. [18]

A composite service is basically a service that is composed of more than one service. This is as opposed to an atomic service which implementation is self-contained and does not invoke any other services. A composite service acts as both a service provider and as a service consumer of its child services. Its role is to aggregate the information it has received from its child services. A composite service can be considered as one kind of a service coordinator, a coordinator that also itself is a service. [19]

From business perspective, WS composition offers a great potential to bring added value for business. This is because the existing application infrastructure can be used to build new applications that can better meet the business requirements and that are more meaningful for business. The new applications or the composite services provide more coarse-grained and simplified functionalities than the atomic services. [19][20]

WS composition is closely related to business processes. This is explained by the fact that the composition logic inside every composite service is fundamentally a process. The logic specifies the exact order in which the involved services are invoked, similarly as processes consist of tasks that are executed in some specific order. In many cases, composite services actually implement business processes and are used to execute some large-scale business activities. In WS compositions the different kind of process tasks are all interfaced as Web Services.

In practice, there are two fundamentally different kind of techniques (or paradigms) to compose Web Services: orchestration and choreography. The main difference is that in orchestration there is a central coordinator that coordinates the service invocations while in choreography there is not. The two paradigms will be discussed later in this section. [19]

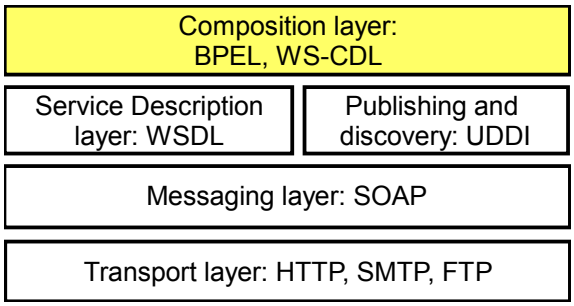


Figure 6: WS protocol stack and composition layer

To build composite Web Services, it is required to use a dedicated programming language for this purpose. Such languages are usually called as WS composition languages. They are quite similar to the usual programming languages but with a difference that they are XML-based languages. They are also somewhat more simpler languages with less programming constructs. Figure 6 depicts how the composition languages form a new layer to the standard Web Services protocol stack. [21]

2.4.1 Orchestration

Orchestration is the more common paradigm of the two different composition paradigms. It is fundamentally about composing Web Services using a central coordinator for this purpose. Because it is required to use a central coordinator, orchestration is better suited for EAI scenarios rather than for B2Bi. [19][21]

The central coordinator's role is to be in control of all services and of their invocations. The coordinator is fundamentally a server system which role is to execute the composition logic of the deployed composite services. The coordinator is often called as an execution engine or an orchestration server.

In orchestration, only the coordinator is aware of the composition, the involved services do not know that they are involved. Figure 7 depicts how the coordinator coordinates the service invocations in orchestration.

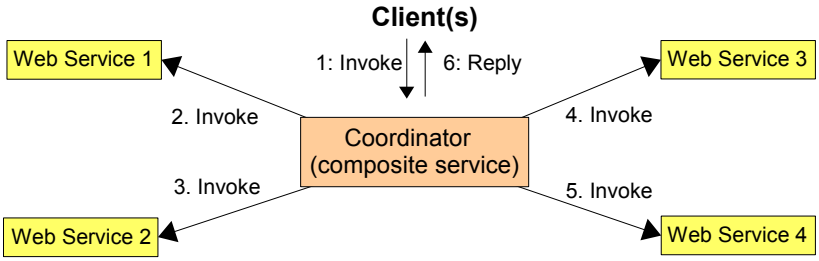


Figure 7: Web Service orchestration

2.4.2 Choreographies

In the choreography style composition there is no central coordinator (server) controlling the invocations. While the orchestration was about composing the private services of a single company, the choreography is about composing the public services of many companies. In a choreography definition is described the global, publicly observable view of a multi-participant process exchange. In choreographies each of the participating services are aware of the process as a whole and know exactly when to collaborate and with whom. Figure 8 depicts an invocation pattern of a choreography in which the participating services are invoking each other. [19][21][22]

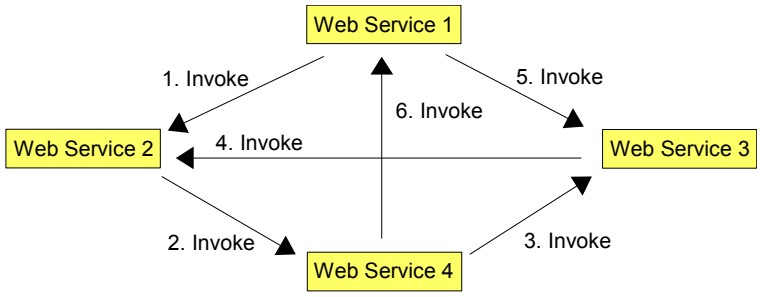


Figure 8: A Web Service choreography

In practice, choreographies are only used to describe B2B processes between companies. A typical choreography is often a long-running transaction such as a purchase order process that involves several companies. The main benefit of using a choreography is that it eliminates the need for a central coordinator and thus, none of the involved companies are able to control the process over the others.

2.5 Web Service Composition Languages

Web Service composition languages are programming languages for building composite Web Services. Most of them are XML-based programming languages that have been designed exactly for this purpose. Programming with WS composition languages is about building new applications from the existing enterprise applications, using the existing ones as basic building blocks for the new. The resulting composite applications are also Web Services themselves (and that can be further composed into new composite services). [19]

In addition to application integration, WS composition languages can be also used to specify business processes. This can be explained by the fact that the composition logic inside every composite service is fundamentally a process. And because this logic can be arbitrary complex, it may well represent a complex business process of an enterprise.

In technical level, the main function of a composition language is to specify the order in which the involved Web Services are invoked. Often, their role is also to provide facilities to support business process specific features such as transactions, events, exceptions, data mappings, etc. Figure 6 depicts the role of the composition languages in the standard WS protocol stack.

Composition languages may be based either on orchestration or choreography style composition. The difference is that orchestration requires the use if central coordinator to execute composition definitions while choreographies do not. The composition languages also differ in whether they are based on a structured or graph-based programming model. While in structured models the control flow is defined by nesting activities into each other, in graph-based models the activities are connected via graph-based links that are like "goto" instructions of where to go next in the process. It should be noted that some composition languages, e.g. BPEL, support both programming models. [19]

The main advantage of using WS composition languages over conventional programming languages, such as Java, C#, or any other, is that the programming logic may be designed graphically (as workflows) using a graphical tool. This enables also the non-programmers to use the language without needing to have an explicit knowledge of the low-level programming constructs. Also, it usually greatly facilitates the maintenance of the code. Moreover, the composition languages are also often considered as much simpler languages as they have only a limited number of basic programming constructs. Composition languages are also better equipped to support XML handling and to support business process features such as transactions. [23]

2.5.1 Standards

Below is listed some WS composition languages that can be used to compose Web Services. It is notable that currently the de facto WS composition languages for orchestration and choreography, are BPEL and WS-CDL, respectively [23]. The rest of the listed standards can be described as outdated. In Figure 12 is shown the different composition languages and their versions in a time-horizon.

BPEL/BPEL4WS (Business Process Execution Language [for Web Services]) is currently the de facto orchestration language and has the strongest industry support behind it. It is being developed by OASIS (a coalition of major software vendors). The latest version 2.0 was published in 2006. BPEL will be more detailed in Chapter 5. [24]

BPML (Business Process Modeling Language) is an orchestration language by BPMI (Business Process Management Initiative). BPML is closely related to the WSCI choreography language. They can be actually used together and they both share the same underlying process execution model and a similar syntax. Today BPML represents an obsolete standard as it has become superseded by BPEL. [19]

WSCI (Web Services Choreography Interface) is a choreography language developed by Sun, SAP, BEA, and Intalio in 2002. It is a predecessor of the WS-CDL language. Today, WSCI standard is no longer used. [15]

WSFL (Web Services Flow Language) is an old orchestration language by IBM which is no longer used as it has become superseded by BPEL. The latest version of WSFL (1.0) was introduced in 2001. [19]

XLANG stands for eXtensible Language and is an orchestration language by Microsoft. Before BPEL, it was the language that was used in the Microsoft's BizTalk Server integration software. [19]

WS-CDL (Web Service Choreography Description Language) is the successor of WSCI and represents currently the most prominent choreography language. WS-CDL enables to describe the publicly observable behavior of B2B processes. WS-CDL is standardized by W3C and its first version was released in 2005. [15]

WSCL (Web Service Conversation Language) is a lightweight choreography language proposed by HP for specifying B2B processes. Its latest version was published in 2002 and it represents today a standard that is no longer used. [15]

2.6 Service Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is an enterprise application integration architecture that builds upon Web Services. SOA is fundamentally not a technology but rather an architecture that leverages many different technologies. In SOAs the most of enterprise's business applications have been exposed as Web Services. The service-exposed applications build a homogeneous application platform in which applications are accessed through standard interfaces and using a single protocol only. Upon such a platform it is possible to easily implement automated business processes and composite applications. [25][26]

The main goal of SOA is a flexible IT infrastructure of an enterprise that is capable of quickly adapting to changing business requirements. A prerequisite for this is that the services of a SOA are meaningful, reusable, and loosely coupled, or so called “business services”. By “meaningful” is meant that services should provide only coarse-grained business operations that match complete business tasks. Such services may be more easily utilized compared to if they were only system APIs that were exposed as services. “Reusable” means that a service should not be limited to one context only but should be instead such that may be used in various different business scenarios. “Loosely coupled” refers to services that are self-contained and not bound geographically nor technologically to any certain platform. [25][27]

WS composition is playing a key role in SOA because it enables to build new applications and business processes from the services the SOA has exposed. WS composition is basically like the configuration component of a SOA that enables to easily build new applications by using the existing applications as basic building blocks for them but also to refine its current application on the fly. This brings a great amount of flexibility to an enterprise and allows it to more quickly respond to changing conditions. [11][25]

Figure 9 depicts the four basic layers that build the SOA architecture. In terms of EAI, these layers represent an n-tier architecture. The bottom layer depicts the existing IT infrastructure of an enterprise. In the second layer, the enterprise components (application server applications) group the individual applications into more meaningful entities. The role of the services layer is to expose the enterprise components and other resources as Web Services for standard application access. In the top layer, the atomic services have been composed to composite services that provide more meaningful services for the various business needs. The composite services may represent also business processes. [28]

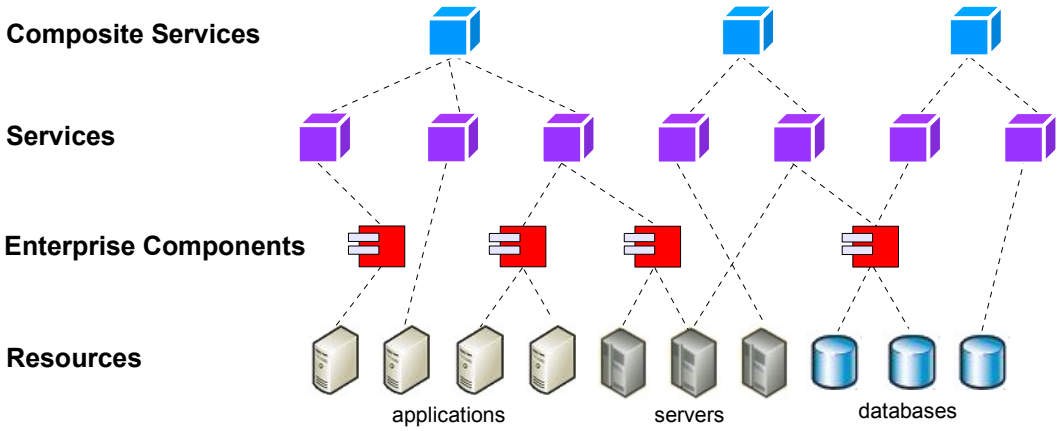


Figure 9: Layers of SOA

Lastly, it should be noted that a common misconception of SOA is that it is always bound to Web Services. This is basically not true because there are also some non-Web Service integration architectures that can be considered as SOAs. Such platforms include at least CORBA, Jini, OSGi, UPnP, and OpenWings. The last three of them are targeted at specific application integration domains such as homes, cars, or embedded systems. The first two are targeted at EAI but have not become very common platforms. [29]

3 Business Process Management Essentials

In this chapter we are going to discuss some important topics of business process management. This chapter serves as a prelude to the subsequent chapters in this thesis. In chapter we are going to discuss the evolution, foundational technology, architecture, and standardization of BPM. We are also going to introduce the concepts of business process lifecycle and workflow management.

3.1 Evolution of BPM

According to [1], [3], [4] and [30], the evolution of BPM may characterized by three major phases or "waves".

Pre-information age (1920s-1960s)

It is most often considered that the systematic management of business processes began in enterprises in the 1920s when the process thinking started to become popular. This was a consequence of the Frederick Taylor's famous theory of scientific management, also known as Taylorism. Taylorism is fundamentally a theory that laid down the fundamental principles of large-scale manufacturing through assembly-line factories. The theory focuses on production efficiency by breaking every action, job, or task into small and simple segments which can be easily analyzed and taught. The factory workers were trained at that time to follow specific steps that required narrowly focused skill and endurance. The goal was to gain a maximum efficiency from both machine and worker, and maximization of profit for the benefit of both workers and management.

First wave – process improvement (1970s-1980s)

In the end of 1960s, the emergence of computers enabled new kind of opportunities to improve business processes. Many enterprises started to adopt more systematic approaches to manage their business processes. At that time, the main focus was on measurable processes and speed that was combined into "Just-in-time" manufacturing.

The first kind of information systems for process management were the industrial computer-aided manufacturing systems. They revolutionized a large number of factories in the 1970s. The roots of modern BPM systems are fundamentally in these systems as they evolved gradually into a more generic type of systems that could be applied also to non-industrial processes. The earliest commercial BPM systems or actually workflow management systems were introduced in the 1980s. The earliest workflow management systems were build around document management. They were fundamentally employee productivity tools that enabled routing of office documents between employees in offices.

Second wave – process reengineering (1990s)

The second wave was driven by Michael Hammer and James Champy's new management trend called Business Process Reengineering (BPR). The idea of BPR was that processes should be manually reengineered through a one-time activity to achieve a radical improvement in business performance. BPR was fundamentally a management approach in which technology (IT) was not considered as a major issue but was having more of a supportive role.

A preferred solution to cast in changes in processes at that time was to use ERP suites. The problem with ERP suites was that they did not really address BPM. To install an ERP suite meant that the enterprise's processes must be fitted to the "best practices" dictated by the contemporary BPR evangelists. The "best practices" were usually those that were common to the whole industry and did not allow companies to differentiate from their competitors. Today, introducing an ERP is often considered as a bad approach to BPM.

In the mid-90s, after the emergence of Internet and e-business, many businesses were forced to become more responsive to constant change because business opportunity windows become ever shorter. This increased the importance of business processes and BPM in enterprises even more. Also the need to connect enterprise applications grew more apparent as the process tasks were more and more often performed by information systems. This lead to a rising interest towards EAI systems in many enterprises.

Third wave – processes in constant change (2000-)

In the third wave, or in the current wave, business processes are freed from their concrete castings in technology. They are made the central focus in all of enterprise's integration efforts. The primary goal of BPM in the third wave is to better equip the enterprise with capabilities to respond to change. This is made possible by agile and flexible business processes that are like first-class citizens of all enterprise assets. Continuous process improvement and value chain monitoring play also a very important role.

The technology strategy in the third wave is not a piecemeal replacement of legacy applications with a new system such as ERP but to preserve the existing applications. The goal is to separate (and centralize) only the process logic from the applications but nothing else. As a response, a new enterprise system category called Business Process Management System (BPMS) has appeared. A BPMS is fundamentally a process-driven EAI system that is light-weight and enabled to execute the process logic.

The push for the third wave has been largely affected by the emergence of Web services. Also the advances in other Web technologies have made it easier to apply BPM in practice, e.g, the shift from stand-alone programs to Web applications has enabled to easily integrate also humans to business processes (through Web portals).

3.2 Foundational Technology

The hype around BPM in recent years has arisen much because of Web Services and related technologies. These technologies build the foundation of systems that are capable of executing business processes, i.e., Business Process Management Systems (BPMS) and Enterprise Service Buses (ESB) (see Chapter 6). Below we have explained the relationship between BPM and these technologies. [31][32][33][34]

Web Services - The main benefit of Web Services in BPM is that they provide an application integration platform that is based on open standards. As we know from the previous chapter, BPM today is more or less a process-oriented approach to application integration. Until recently, at the time when there were no common application integration standards, the majority of BPM (EAI) solutions were based on proprietary technologies. Such BPM solutions were not only very complex but also very expensive. Today, Web Services enable organizations to eliminate the constraints of such proprietary infrastructures. Web Services standards are especially welcomed in BPM approaches aiming at integrating business processes between companies and over Internet (BPM for B2Bi). Without platform- and language independent Web Service standards, it would be virtually impossible to implement such processes.

Another major benefit of Web Services in BPM is that they enable to build automated business processes that are loosely coupled. Loosely coupled process definitions are such in which application invocation details are kept to minimum. To such processes it is easier to implement changes. If the process was instead tightly coupled then implementing changes is often much more difficult. For example, if the process definition consisted of a multitude of database queries then it would be very time-consuming and costly to implement even the smallest changes to the definition. This is the main reason why it is better to have Web Services as intermediaries between process definitions and applications.

SOA - BPM's relationship with SOA is as close as it is with Web Services. Figure 10 depicts the relationship between BPM and SOA. SOA acts as an application integration platform which responsibility is to provide service for the BPM layer. SOA layer is like an inventory of services to be used for an enterprise's BPM purposes. In addition, the SOA layer also has to provide some additional facilities for the BPM layer such as security, reliable delivery, transactional capabilities, and of course a service directory (UDDI). It is generally thought that a SOA is not a real "SOA"if it is not combined with BPM because then it is not possible to take the full value of it. [35]

WS composition (languages) – Web Services themselves are not sufficient for implementing automated business processes and applying BPM. For this purpose it is required to introduce Web Service composition languages that enable to compose services into business processes. Fundamentally, as each service of a a SOA usually corresponds to an individual business task, by service composition it is possible to assemble these tasks rapidly into end-to-end business processes.

From the two composition paradigms (orchestration and choreography), orchestration is a more common paradigm and better suited for the process automation purposes. From the orchestration languages the BPEL language is most commonly used in the today's BPM solutions.

Application Server Technologies – Application server technologies play also a major role in today's BPM. Application servers provide the "plumbing" for application integration and building SOAs. In addition, the majority of today's BPM systems have been implemented upon some application server. From the different application server technologies (J2EE, .NET), J2EE has been more often used to implement BPM solutions.

WS-* standards – Many of the today's BPM solutions are relying also on the second-generation Web services technologies including WS-Coordination, WS-MetadataExchange, WS-ReliableMessaging, WS-Security, and WS-Transaction. The role of the second-generation technologies is to broaden the feature-set of first-generation Web Service standards (WSDL, SOAP, UDDI).

3.3 Architecture

In Figure 10 is shown what BPM is all about today [36]. The architecture represents how a business process may be automated in practice using BPM and Web Service technologies. In the the uppermost layer is a business process diagram consisting of sequential tasks (or process steps). In the second uppermost layer the diagram has been transformed into an executable process definition. The coarse-grained process tasks have been broken down into more fine-grained execution tasks. The execution tasks consist of, inter alia, service invocation details. The executable definitions are fundamentally service orchestrations, or composite services, that orchestrate enterprise applications that have been exposed as services by SOA. Enterprise components (such as EJBs) are used in many enterprises to logically group applications into more meaningful entities.

The roles and technologies that are related to each layer are shown in red and blue text. In this architecture, the division of roles is such that business people (executives, business analysts) are responsible for the coarse-grained process designs (diagrams) and IT people for all the rest.

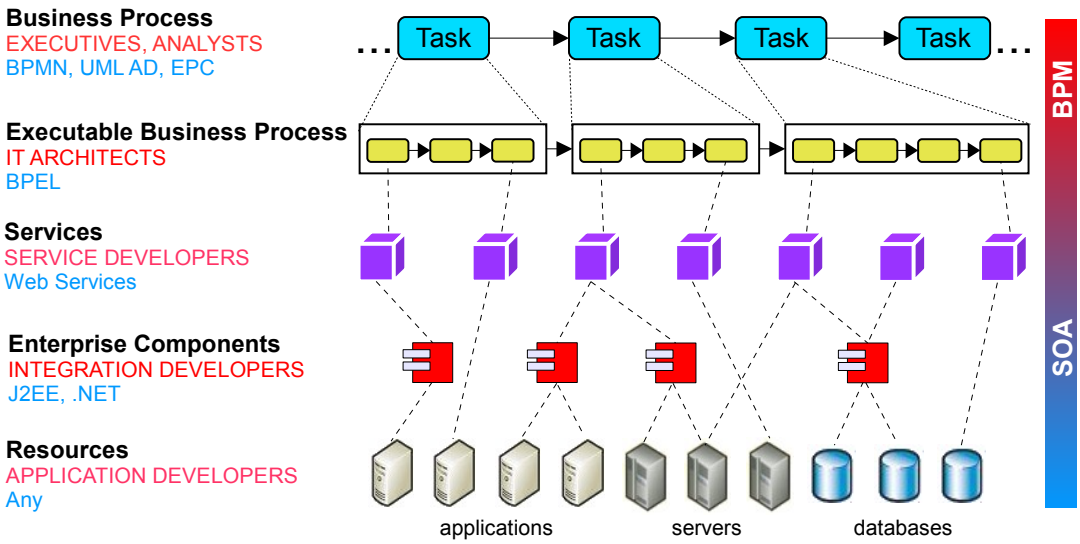


Figure 10: Architecture of BPM

3.4 Business Process Lifecycle

To evolve into a process-based organization, an enterprise has to move from the management of static processes to the management of business process lifecycles. Only by supporting business process lifecycles, and thus, constant improvement, an organization is enabled to find its optimal operational performance. In addition, an organization that is static, structured, and predictable can become more dynamic or such that is better equipped to respond to change. [31]

The business process lifecycle concept of BPM is illustrated in Figure 11. A lifecycle is a continuous loop that consists of four phases (according to [37], [38], [39]) including modeling, automation, management, and optimization phases. Basically, to enable constant improvement of business processes in an enterprise, one should iterate the different phases all over again – starting from the modeling phase and ending into the analysis (optimization) phase.

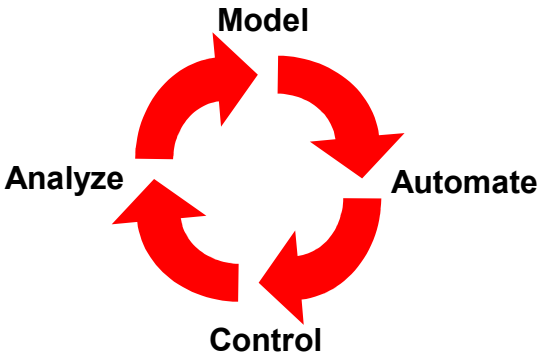


Figure 11: Business process lifecycle

In the first phase, or in the modeling phase the business process is given a design in a conceptual level. The output of the modeling phase is typically a process diagram. A diagram captures the process logic at a high abstraction level enabling the process to be easily understood with the human eye. The diagram serves in essence as a communication tool between people. It enables to explicitly describe the process rules, responsibilities, and purpose to the different process stakeholders.

In the automation phase coarse-grained process models are transferred into executable process definitions, usually by IT developers. The goal is to enable the execution of process logic through an information system. Before it is possible to execute the process, all the necessary process resources including applications, data sources, and people must be linked to the process definition. The process execution environment is usually a system such as a BPMS (see Chapter 6) or similar.

In the control phase the deployed processes are monitored and administered by the process managers and administrators. Monitoring is required to ensure that the process is carried out as defined and to take corrective actions as and when required. Corrective actions are required when delays occur, exceptions arise, resources are unavailable, and so on. To be able to control processes, most BPM systems usually bundle a management console for this purpose.

In the analysis phase the deployed processes are analyzed for their performance and cost effectiveness. The goal of this phase is process improvement. The analysis is based on measuring key performance indicators and other metrics (for this purpose, there are some dedicated software). Measurements can show how often certain process paths are taken, what elapsed cycle times are, what costs have been incurred, and similar results. As mentioned, the results of the analysis phase usually results in new business process requirements that serve as the basis for the next cycle's modeling phase.

3.5 Workflow Management

Workflow Management is fundamentally a concept that is very closely related to BPM. Workflow management is dealing with more or less about the same thing as BPM – its goal is control flows of work in an enterprise so that work gets done at the right time by the right person or application. The only main difference between the two is fundamentally that workflow management is having a smaller scope. [40]

The term workflow is used to denote a "sequence" of tasks that is represented as a graphical diagram. A process may be distinguished from a workflow by the fact that it has more well-defined inputs, outputs and purposes. Workflows apply more generally to any systematic patterns of (work) activities. A traditional example of workflows is routing of important work documents from one employee to another in an office. [38][41]

The term workflow management may be defined as “the combination of software, tools and services to automate knowledge-based tasks, on an ad hoc or production-based level. .. During automation information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

Workflow management differs from BPM mainly in that it is more of a technical concept than a management discipline. BPM implies also the non-technical issues like analysis, organizational impact, from the viewpoint of a manager. The flows of work or processes are usually considered of being of smaller scale in workflow management. In workflow automation, the process lifecycle thinking is neither playing such a centric role as it is in BPM.

Workflow management has been applied in practice traditionally through Workflow Management Systems (WFMSs). A WfMS is fundamentally an enterprise system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic [42]. WfMSs have been used in enterprises in the first place to route information between employees. They have been in use ever since the 1980s. It is notable that WfMSs have basically started the era of modern business process automation. Today's BPM systems can be considered as having evolved from these systems and of being the latest step in the evolution. For more about WfMSs, see Section 6.1 “Workflow Management Systems”.

3.6 BPM – A Convergence of EAI and Workflow Management

According to Khan [38], BPM today is actually all about EAI and workflow management having converged together. In the following we explain briefly why and how this has happened.

The term BPM did not become a widely recognized term until the late 1990s. With the advent of advanced EAI solutions in the late 1990s, the idea of managing the entire life cycle of a business process through IT became finally viable. This is backed by the fact that the analysts firms, including Gartner, began tracking the BPM market not before the year 2000. Formerly, the same analysts were tracking EAI, workflow management, business process re-engineering, and document management. After the year 2000, or during this decade, BPM has matured tremendously both as a market and as a category of products and services. [43]

Compared to EAI, BPM is a more process-oriented approach to application integration. Traditionally, EAI has been conceived of being only a middleware technology that provides the "plumbing" to route data between applications. Many of the EAI solutions have had a very application centric way to view the world. Only recently, the idea of integrating applications into business processes, together with other enterprise assets including people, has emerged. This has resulted in that today the EAI systems provide many similar features to WfMSs in order to support business processes. [44]

Compared to workflow management, BPM is having a broader scope. While workflow management is more of a technical concept, BPM is viewing things more from the enterprise management perspective. We can also say that while workflow management is focused on executing some short sequences of tasks (usually human tasks), BPM is more focused on automating complete business processes consisting also of application tasks. Unlike workflow management, BPM is also concerned with issues such as process monitoring, analysis, and optimization.

The main reason for the convergence of EAI and Workflow Automation to under the umbrella of BPM is Web Services. As Web Services have become the most prominent integration technology in both of the domains, it has become possible to build sophisticated business integration systems that combine the best sides of the former EAI and workflow automation systems. [43]

The convergence of EAI and workflow products is also noticeable in the BPM marketplace. A wide array of traditional EAI and workflow vendors have jumped recently into the BPM market – each desperately attempting to sell their products as "BPM enabled". It is notable that while most of the vendors promise to deliver more or less the same, in reality, the products differ significantly in their technical approach and functional capabilities. Many of the vendors are actually only selling their former EAI or workflow products but renamed. [43]

3.7 Standardization

The standardization of BPM technologies can be said to be quite tangled. This can be explained by the fact that today there are a number of different standardization organizations and vendors who are concerned with the standardization of BPM. The fragmented standardization and the lack of cooperation has lead to many misunderstood, incompatible, misapplied, and excessively hyped standards. Many definitional confusions also feed the fragmented market. [20][40][45]

Only a few of the BPM standards have become widely adopted by the software industry. Some of the most widely adopted standards include BPMN, BPEL, UML, and WS-CDL. These standards offer yet only some partial answers to some particular problem areas, e.g., BPEL only standardizes the executable process definitions, BPMN only the process diagrams, and so on. Much more should be done to have more widely adopted and interoperable BPM standards.

Below is listed the five major standardization organizations that have had the biggest influence on the standardization of BPM in recent years.

OMG – Object Management Group got involved in BPM or in workflow technology in the end of 90s when they released the Workflow Management Facility specification in 1998. Today, OMG is known especially for the BPMN and UML modeling language standards. Currently, OMG is working with the Business Process Definition Metamodel (BPDM) specification which is a file format and a metamodel for BPMN. [46]

OASIS – Organization for the Advancement of Structured Information Standards is an industry group that has provided several XML-based standards for Web Services and business integration. OASIS is best known for the BPEL standard. OASIS has also contributed to the ebXML B2B integration framework. [24]

W3C – World Wide Web Consortium has been focused on Web Service choreographies and published a couple of choreography language standards. The WS-CDL choreography language is currently the de facto standard in this area. [15]

WfMC – Workflow Management Coalition is the oldest and probably the most mature standardization organization. It was founded in 1993 and was the first organization to promote workflow standards. Today, WfMC consists of over 300 members ranging from workflow vendors and users to analysts and university/research groups. According to WfMC [42], their mission is "to promote and develop the use of workflow through the establishment of standards for workflow terminology, interoperability and connectivity between workflow products". WfMC is best known for its interface specifications for workflow management systems, and for the XPDL process diagram file format. [42]

BPMI – Business Process Management Initiative is today a non-existent organization as it has merged with OMG in 2005. BPMI has yet published several standards that are relevant still today. BPMI started off as an industry consortium to promote BPM standards in 2000. The most relevant standard they have published is the BPMN process modeling language (published in 2004). BPMI is also known for the BPQL (Business Process Query Language) standard. [46]

3.7.1 Summary of BPM Standards

In Table 1 is presented a summary of the standards the BPM standardization organizations have published in the last years [15][20][24][42][45][46].

Acronym	Standard	Organi-zation	Type	Version / year / status
BPDM	Business Process Definition Metamodel	OMG	metamodel and exchange format, BPMN serialization	draft
BPEL	Business Process Execution Language	OASIS	execution language	2.0 (2007)
BPML	Business Process Modeling Language	BPMI	execution language	deprecated
BPMN	Business Process Modeling Notation	OMG	notation	1.0 (2003)
BPQL	Business Process Query Language	BPMI	BPMS query language for administration and monitoring purposes	planning (abandoned?)
BPRI	Business Process Runtime Interface	OMG	BPMS API for administration and monitoring purposes	planning
BPSM	Business Process Semantic Model	BPMI	metamodel of BPDM	planning
BPSS	Business Process Schema Specification	OASIS / ebXML	choreography (and collaboration) language	1.01 (2001)
BPXL	Business Process Extension Layers	BPMI	BPEL extension for transactions, human w orkflow , business rules, roll-backs etc.	planning (abandoned?)
BTP	Business Transaction Protocol	OASIS	choreography language	deprecated (2002)
IDEF3	Integrated Definition Standards	U.S Air Force	classical process notation used in economics	final (1970s)
Petri Net	Petri Net (mathematical definition)	Carl Petri	notation, mathematical graph-based modeling technique	final (1962)
UML AD	Unified Modeling Language Activity Diagram	OMG	notation	2.0 (2002)
WAPI	Workflow API	WfMC	API for WfMC BPMSs (in C-language)	2.0 (1998)
WfXML	Workflow XML (for SOAP, v.2.0)	WfMC	choreography (and collaboration) language	draft
WRM	Workflow Reference Model	WfMC	architecture	1.1 (1997)
WS-CDL	Web Services Choreography Description Language	W3C	currently a leading choreography language	1.0 (2005)
WSCI	Web Services Choreography Interface	W3C	choreography language	deprecated
WSCL	Web Services Conversation Language	W3C	choreography language	deprecated
WSFL	Web Services Flow Language	IBM	execution language	deprecated
XLANG	Web Services for Business Process Design	Microsoft	execution language	deprecated
XPDL	XML Process Definition Language	WfMC	execution language	1.2 (2005)

Table 1: Summary of common BPM standards

3.7.2 Evolution of Standards by Organization

Figure 12 depicts a bunch of relevant BPM standards in a time-horizon grouped by different organizations [47].

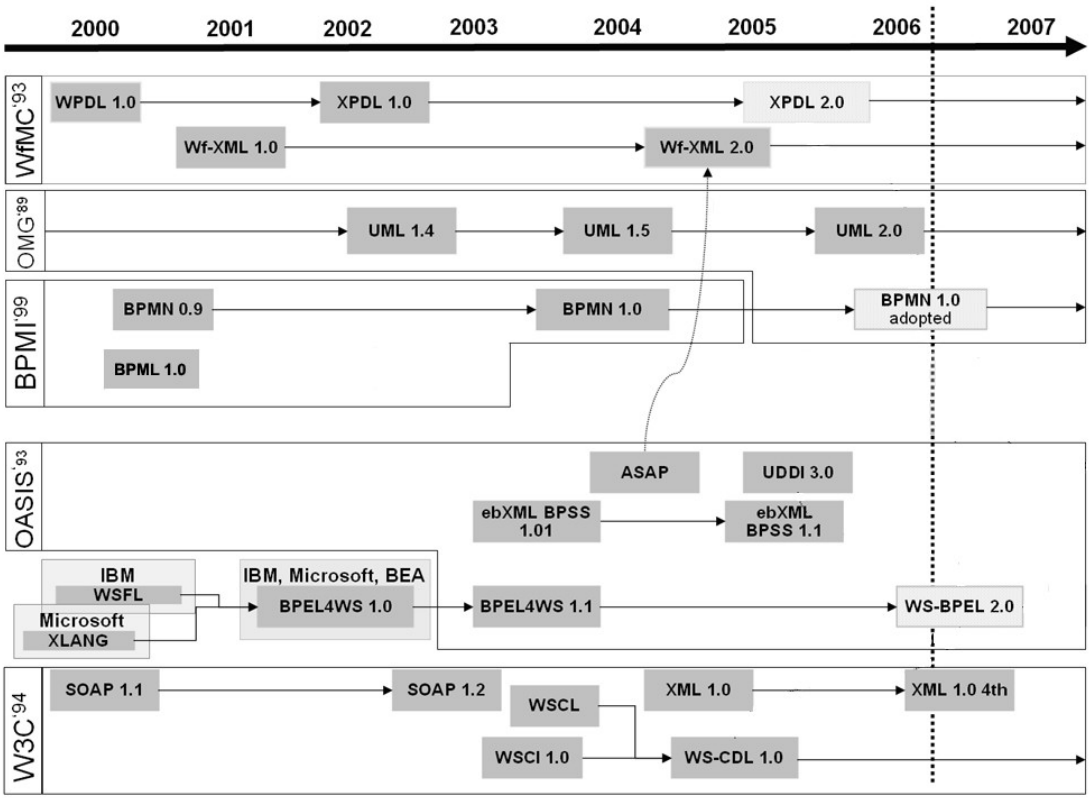


Figure 12: Evolution of BPM standards

4 Business Process Modeling

In this chapter we discuss business process modeling. We discuss different modeling techniques and languages, and about their differences. We also introduce the concept of granularity of process models and discuss what would an ideal process modeling language be like. In the later sections we focus on the two most popular business process modeling (diagramming) languages, BPMN and UML Activity Diagrams. Lastly, we discuss how to save business process models to file and what kind of file formats there are.

4.1 Overview

Business Process Modeling is about documentating a business system by diagramming or using some other modeling technique. Modeling is an activity usually performed by business analysts or executives in companies. The goal of business process modeling is to facilitate human understanding and communication or to support process improvement. Business process modeling is sometimes confused with Business Process Management because they are sharing the same acronym – BPM.

In business process lifecycle concept of BPM, the modeling phase is often considered as the most important phase. As we remember from the previous chapter the lifecycle starts from the modeling phase and ends to the the analysis phase, and that we get the next round's modeling requirements from the previous round's analysis phase.

Business process modeling should be thus a continuous activity. The processes of a company can stay consistent with the prevailing business conditions basically only if the process models are constantly redefined and refined. Otherwise a company is probably giving away in process performance and in its competitiveness. Constant modeling, remodeling, and optimization are often said as the keys to success.

A common misconception about modeling is that it refers to only diagramming of processes. This is not the whole truth because there are also some other methods that can be used to model business processes. For example, one can specify the same process also as a set of plain text rules, as programming code, or as mathematical formulas.

Although most people understand the same thing with terms modeling and diagramming, in theory, there is a fundamental difference between the terms. Namely, while a diagram is only one of the alternative ways to represent the model, the model represents all the information that have been captured from the business process (using any modeling technique). In other words, while a model is an abstract concept, a diagram is something more concrete. A diagram can be thought of viewing the information of the model only from one angle.

4.2 Diagramming and Other Modeling Techniques

Using a diagram is by far the most popular way to model a business process. Diagramming enables to capture the process logic at a high abstraction level. The main benefit of diagrams is that they are easily understandable by human viewers. Compared to other notations, e.g., if the process was described as text, a diagram provides a much clearer view to its purpose, rules, and responsibilities. The diagrams serve in essence as communication tools that facilitate the communication of processes to people.

According to Bog [39] and many others, the most common business diagrams today are BPMN (Business Process Modeling Notation) and UML (Unified Modeling Language). BPMN is a newer standard of the two (released in 2004) and has been developed exclusively for business modeling purposes. UML, instead, is considered as maturer and more widely adopted standard. The drawback of UML is that it is rather a technical standard and aimed more at software engineers than business people.

4.2.1 Other Techniques

Writing programming code may be considered as process modeling as it is, after all, all about writing sequential instructions together with conditional statements – or writing process logic. From a technical viewpoint business processes differ from programming scripts only in that they are of larger scale. Java, C, Assembler, and any other programming language can be fundamentally used to define (automated) business processes. The only difference between traditional programming languages and dedicated process modeling languages such as BPEL is that the latter one provides such constructs that are better suited to be used with business processes.

Process modeling in the mathematical level is process modeling at its purest. Process theories and algebras such as Petri Nets, Abstract State Machines, and Pi-Calculus allow to capture the process logic as mathematical formulas. They offer the means to unambiguously capture the logic of any kind of processes – not only of business processes. It is notable that most of today's process diagramming techniques and execution languages are based on either of these theories. For example, BPMN and EPC are based on Petri Nets, WS-CDL on Pi-Calculus, and BPEL on both of these theories.

Rule-based modeling is about specifying processes using a set of logical rules. Many rule-based languages have been developed for this purpose in recent years, e.g., RuleML, which is an emerging industry standard for rule-based process modeling. The rules are usually specified using predicate logic (a mathematical language to represent knowledge) with rule languages.

4.3 Business Process Modeling Languages

In this section we first present a classification of the modeling languages before we present the actual languages.

4.3.1 Classification of the Languages

In Table 2 is presented the majority of contemporary process modeling languages. The languages have been classified in regard to four parameters. The classification is based to the classification in [48]. The aim of the classification is to emphasize the fundamental differences between the modeling languages. Our classification differs from [48] in that we have used also the "is executable" parameter. It was added because the executable processes are of special interest of this thesis.

	has a notation	has a file format	has a formal semantics (metamodel)	is executable
BPDM*	**x**	**x**	**x**	
BPEL		**x**	**x**	**x**
BPMN	**x**			
BPMN (2.0)*	**x**	**x**	**x**	**x** (?)
EPC	**x**		**x**	
IDEF3	**x**			
Petri Net**	**x**		**x**	
Pi Calculus**			**x**	
RAD	**x**			
UML AD	**x**	**x**	**x**	
XPDL		**x**		

* not released yet

** is a theory, not a standard

Table 2: Classification of modeling languages

From the above classification we can see that there are several languages providing at least a notation. From these quite many are also having a formal semantics. Much lesser languages are having a file format and even lesser are executable. Basically, the more sophisticated a language is, the more it is having checks in our classification. Below is explained shortly the different parameters:

Has a notation – It is not always obvious that a modeling language comes with a diagram or any graphical notation at all. There are several modeling languages that do not come with a diagram and are nor linked to any other diagramming language. For example, BPEL is such a language.

Has a formal semantics (metamodel) – If the modeling language is based on a formal metamodel, or is having a formal semantics, then all the meanings of the terms and concepts of the language have been explicitly specified in the language's specification. The metamodel of the language can be considered like a unified vocabulary and grammar of the language. It enables the different users of the language to understand similarly of what is really being said in the models. The metamodel is specified in the language's specification as text, pseudo-code, and/or (meta-)diagrams.

Some of the diagramming languages are not based on a formal metamodel. For example, traditional notations such as IDEF3 and RAD specify only the shapes of the diagrams and nothing else. This kind of languages leave the interpretation of the diagrams to the modeler and thus, what is being understood by the model can vary between users and context. Sometimes it is difficult to say that whether a language provides a formal metamodel or not, e.g., BPMN is said to have a semi-formal metamodel.

In practice, we can see the benefits of a metamodel-enabled language when using a graphical modeling tool. If the tool supports the language's metamodel, instead of being just a set of stencils, it can provide the modeler with modeling-time guidance. For example, the tool may warn or prevent the user from drawing a line into a wrong place on the diagram. For this, the tool verifies each change to the diagram against the language's metamodel which has been built in to the tool.

Another major benefit of the modeling language's metamodel is that it allows the transferring of process models into other languages. For example, the emerging BPMN metamodel, BPDM, enables a modeler to perform mappings between BPMN and UML Activity Diagrams. In an ideal case, different diagrams would be only different views of the same metamodel.

Has a file format - Each modeling standard either comes with a built-in file format or not. The benefit of a built-in file format is that it enables the easy interchange of process models across modeling tools. If the modeling language doesdoes not come with a file format, the problem is that one is required to use proprietary file formats of the design tools. Proprietary formats are often problematic because they tends to cause vendor lock-ins. A vendor lock-in means that it is not possible to change the modeling tool without significant change costs. For more about file formats, see section 4.8 “File Formats for Process Models”.

Is executable - An executable process modeling language is such that is metamodel-enabled and has a file format. The file format of an executable language is however such that can be executed through a process execution engine. In other words, it can be said that such a file format provides execution semantics. The execution semantics refers to the run-time details of the process or how the process should be executed by an execution engine.

The executable modeling languages are fundamentally programming languages but that differ from the traditional programming languages in that they have different kind of computational foundation. The executable process languages (e.g., BPEL) are having their foundation on process theories such as Pi-Calculus or Petri Net, while the traditional programming languages are based on very different theories.

4.3.2 The Languages

In this section we have described in more detail the languages that were listed in Table 2. The languages of Table 2 represent the most current business process modeling languages. The list of the Table 2 has been built according to [39][48], [49] and [50].

Event Driven Process Chain (EPC) - EPC language aims at being an easy-to-use language that both IT and business people should be able to use. The EPC language has been developed within the framework of ARIS by Prof. Wilhelm-August Scheer in the early 1990s. Today, it is still used by many companies for modeling, analyzing, and redesigning business processes. The basic building blocks of EPC diagrams are "events" and "activities". EPCs are used widely in SAP R/3 and ARIS systems. [49]

Integrated Definition Method 3 (IDEF3) - IDEF3 is a classical school example of process modeling. IDEF3 method has been originally to model sequences of a system but it may be used also for modeling business processes. IDEF methods (0-16) were developed in the U.S. Department of Defense in the 1970s. They are still quite commonly used. [49][51]

Petri Net - Petri net is a graphical and mathematical modeling tool for modeling any kind of processes. Although Petri Net is also a graphical notation, it is in the first place a mathematical tool. Petri Net serves as the mathematical foundation of several modeling languages including BPMN, UML Activity, BPEL and WSFL. A Petri Net graph is a directed graph that consists of two different nodes, places and transitions from which places represent possible states of the system and transitions represent actions that can cause the change of state. The theory of Petri net was invented by Carl Petri in 1962. [39][49]

Pi-Calculus - Like Petri Net, Pi-Calculus is also a mathematical modeling tool. As such it serves as the mathematical foundation of several modeling languages including WS-CDL, XLANG, WSCI, BPEL, and BPML. Compared to Petri Net, Pi-Calculus does not provide a graphical notation but it is merely an algebraic language that allows one to express any computation. Pi-Calculus is fundamentally an alternative computability theory to Church's lambda-calculus and Turing's theory of computation. The main design goal of Pi-Calculus has been to allow to describe concurrent computations whose configuration may change during the computation. The use of Pi-Calculus for business process modeling requires a senior level of mathematical training. Pi-Calculus was invented by Professor Robin Milner and his team in the 1990s. They received the respected Turing Award for their efforts. [39]

Role Activity Diagram (RAD) – Although RAD diagrams have been originally designed for modeling human interactions, they may be used also for business process modeling. RAD diagrams are often said to be simple and relatively easy to read but still capable of providing a powerful modeling technique to express complex process behaviors. The basic building blocks of RAD diagrams include "roles", "activities", "interactions", and "external events". The Role Activity Theory was introduced by Anatol Holt et al in the early 1980s. [49][52]

Business Process Execution Language (BPEL) – see Chapter 5

Business Process Definition Metamodel (BPDM) – see Section 4.8

Business Process Modeling Notation (BPMN) – see Section 4.5

Unified Modeling Language Activity Diagrams (UML AD) – see Section 4.6

XML Process Definition Language (XPDL) – see Section 4.8

4.4 Granularity of Process Models

It is important to see that different business process models capture different amounts of process detail and information. There are fundamentally coarse-grained process models and there are fine-grained process models. Coarse-grained models (e.g., diagrams) capture the high-level process behavior, while fine-grained models capture also the low-level implementation details. Thus, the more fine-grained the model the more information and details is captured in it. In this way, we can build a kind of classification into which we can put models, modeling techniques and modeling languages in regard how much details they capture or are able to capture. By the term granularity of a process model is meant exactly this.

The required level of granularity from the model is dependent on by whom is the model used and for what purpose. Namely, it is not always desirable to have a model that provides the modeler with exhaustive details. For example, a business analyst or a manager who is responsible for the overall process design is probably not wanting to see any fine-grained implementation details of the process but only the coarse-grained behavior that is relevant for their work (diagramming, optimization, etc.). Basically, if too much technical details are shown for the modeler at one time, the modeler usually gets only confused, especially if he/she does not have much technical knowledge. Instead the process developers, responsible for the process implementation, need to see also the technical details in order to be able to make necessary adjustments. [53]

Figure 13 depicts an "ideal" process model (inspired by Wikipedia-"Process Modeling") that is capable of expanding to higher granularity levels to cover new aspects of process modeling. By new aspects we mean, for example, a situation where we want to transfer a free-form diagram (or a sketch) into a more formal diagram to be edited with a computer-aided design tool or even if we want to execute it in an information system. In the figure, the circle size represents the amount of information that is captured into the model. As we can see the the amount of information is directly proportional to the level of granularity in the model. In the model we have included five granularity levels. The levels may thought also as representing the maturity of the process model. The maturity rises each time the model is transferred to the next granularity level, e.g., a free-form is transformed into a formal diagram.

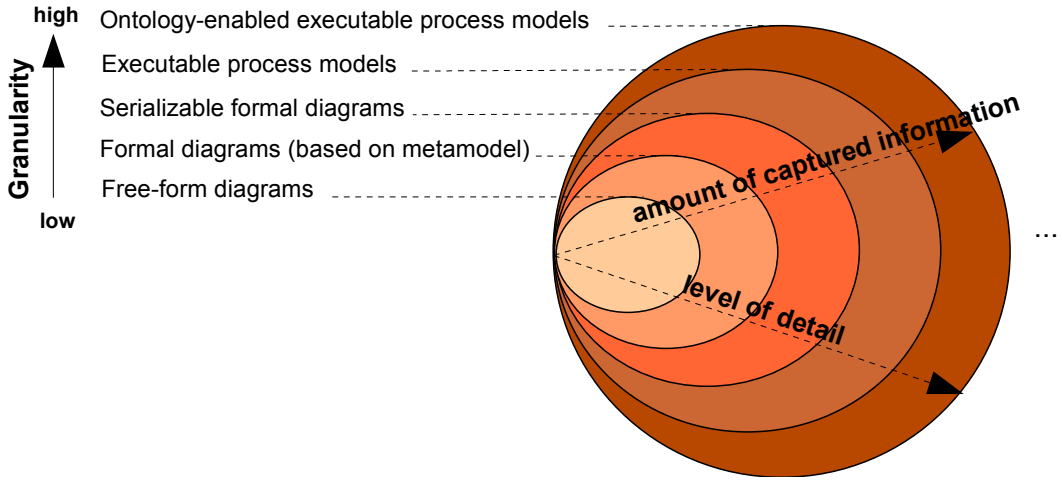


Figure 13: Granularity of a process model

What is important to see in the Figure is that process modeling is more or less vertical by nature. This means it is fundamentally possible to build complex executable models incrementally from simpler models. In such an ideal scenario the original free-form diagram is used as the base for more advanced models. Each time the diagram or the model advances to higher granularity level, it is only specified with more details.

4.4.1 An Ideal Business Process Modeling Language

In Figure 14 is shown how some of the current process modeling languages align with the granularity model. The model of the figure is the same as in Figure 13 but seen from the perspective of modeling languages. This time the circle size represents the granularity of a modeling language, or how much details it is possible to capture from the process using some language. Basically, the bigger the circle the more sophisticated modeling language it is in question.

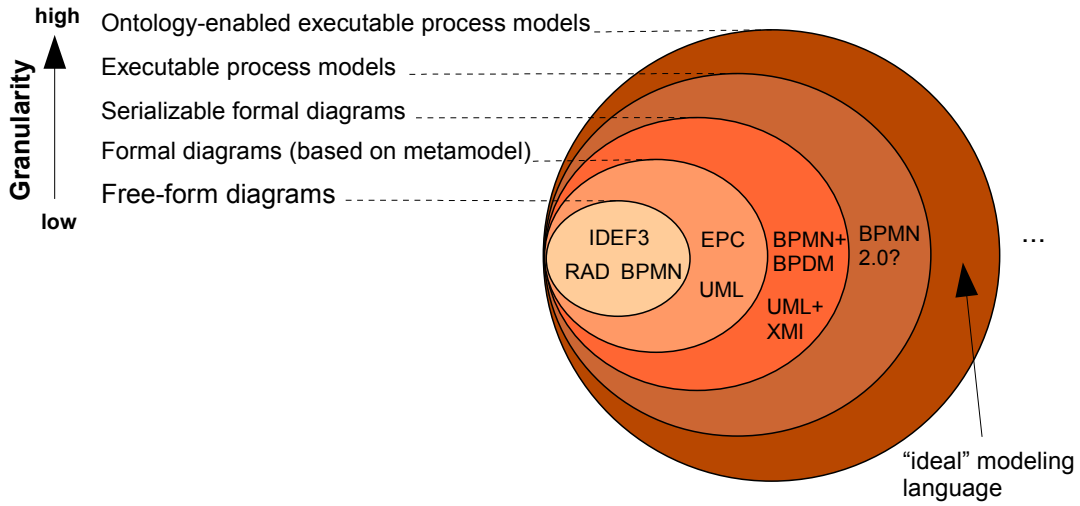


Figure 14: Granularity of a modeling language

Typically, many of today's modeling languages are only point solutions to some particular need or aspect of modeling. They are not vertical in the sense they could cover many modeling aspects at the same time. This is the reason why we are often required to use many different languages at the same time, e.g., to diagram processes, to serialize them into file, to specify executable processes, etc.

The problem of using many modeling languages is that the model gets scattered into many separate models. This in turn requires that the models have to be kept synchronized, or that each change to any of the models has to be propagated to the other models. Synchronization is often difficult to achieve in practice, e.g., keeping the BPEL processes in sync with the BPMN diagrams.

In an ideal case, only a single modeling language would be sufficient for all modeling needs. The same language could be used by all the different roles who are involved in business process modeling including business people and IT people. This would enable also different roles to work with the very same model - rather than different roles having to work with their respective models. There would be thus no synchronization problems in this case.

To enable different roles to work with the same model and to use the same language, the “ideal” language should support information hiding techniques to enable the hiding of low-level details from the high-level design views (of business people).

So far there has not been implemented such an “ideal” language. Yet, it would be completely possible to develop such a language. In the beginning all it requires is a well-defined diagram and a file format for it (preferably XML-based). The file format should be essentially extendible so that it is possible to make further specification to the language later on to support new aspects of modeling, e.g, to support execution semantics.

4.5 Modeling Processes with BPMN

Business Process Modeling Notation (BPMN) is an emerging standard of OMG (Object Management Group). It is a simple graphical notation standard for representing business processes. The notation includes a number of different shapes such as circles, rectangles, diamonds, and many kind of connecting lines. BPMN specifies fundamentally only the diagram shapes but nothing else, e.g., it does not provide a file format in which the diagrams could be saved. Today, BPMN is already quite a popular standard although it was originated only in 2004. Many think that it has already become a de facto standard for business process modeling in enterprises. [54][55]

BPMN is essentially a graphical notation that is mainly aimed at business users (business analysts, executives) rather than IT people. It allows people who aren't programmers to graphically depict the steps involved in business processes, easily and understandably. Compared to traditional modeling notations, such as UML, BPMN is clearly more intuitive with less technical details. [55]

BPMN serves yet also the developer needs who are responsible for automating business processes. BPMN has been designed so that it is possible to relatively easy map a BPMN diagram into an executable process, e.g., into a BPEL definition. There are actually some BPMN-to-BPEL mapping tools that can be used for this purpose. However, it should be mentioned that mapping to BPEL is not always a trivial exercise.

In Figure 15 is shown the BPMN core elements. The flow objects consists of events, activities, and gateways: events trigger actions from process flows, activities perform units of work within the process or sub-process; and gateways represent points of decision within the process. The connecting objects are different kind of connectors between flow objects. Swimlanes are used to organize activities into categories. Artifacts are metadata objects that carry additional information in diagrams, they are used to make the diagrams more readable. The data objects may be used to tell about what kind of data is required or produced by the diagram activities. [54]

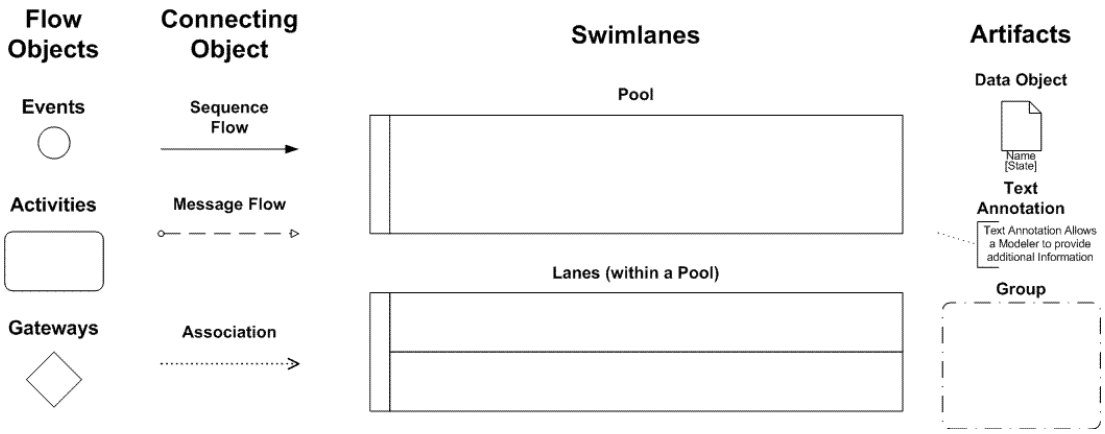


Figure 15: BPMN core elements

A simple BPMN example is shown in Figure 16. The diagram depicts a process of a travel agency in which the process goal is to book a trip including air and hotel. To successfully book a trip, it is required that it is possible to book both the air and the hotel for correct dates. Because neither the airline nor the hotel can guarantee the availability until payment, it is required to pay the hotel before the air can be booked. However, in the case the hotel has been already booked but the air is not available, then the hotel must be canceled in order to get compensation. The process represents basically a simple transaction with compensation.

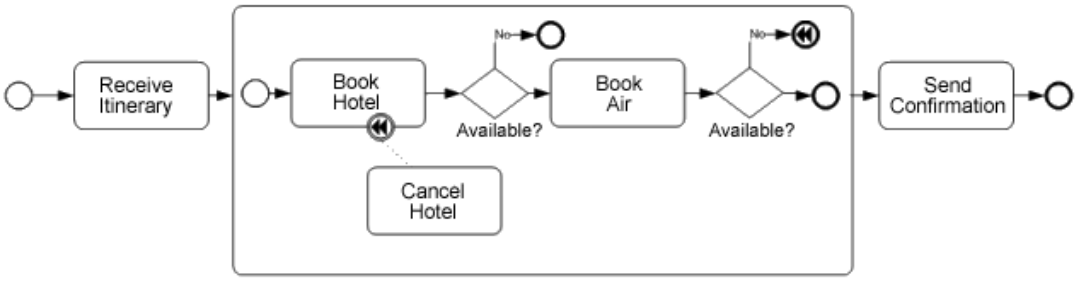


Figure 16: Booking a trip

4.6 Modeling Processes with UML Activity Diagrams

UML (Unified Modeling Language) is a graphical modeling standard developed by OMG (Object Management Group). Its first version (1.0) was released in 1997 and the latest (2.1.1) in 2007. The standard consists of eight basic diagrams including class, use case, state, sequence, activity, deployment and component diagrams. In addition there are a few others but that are somewhat more rarely used. The UML standard has been developed in the first place to describe and design software systems, in particular, software systems built using an object-oriented style. The roots of UML are thus in software engineering. However, despite this, the notation may be also used for business process modeling. [46][56]

From the eight diagrams of UML, the activity diagrams are best-suited for business process modeling. The activity diagrams represent sequences of tasks, or activities in UML terms. Activity diagrams are in many ways the object-oriented equivalent of plain old flow charts. The great strength of activity diagrams compared to flow charts is that they support parallel behavior that is typical to business processes. Activity diagrams are actually quite similar to BPMN diagrams and the basic shapes including circles, rectangles, diamonds and connecting lines look very much alike. [46][56]

Figure 17 summarizes the shapes of the UML activity diagrams. The core elements of activity diagrams are actions that perform individual tasks within processes. Actions may have multiple inputs and outputs and may change the incoming data. The entry and exit points of processes are represented with their respective start and end node symbols. Conditional behavior can be modeled using branch nodes, and the fork/join component can be used to fork control flow into concurrent threads and to merge these threads at a later point. Data flow and communication can be modeled using objects and signals. A dashed line with an arrowhead is used to indicate data flows. Swimlanes may be used to distinguish different roles in processes. Edges are connectors between the elements and can be annotated with guard conditions that must be satisfied in order to transition to take place. [46][56]

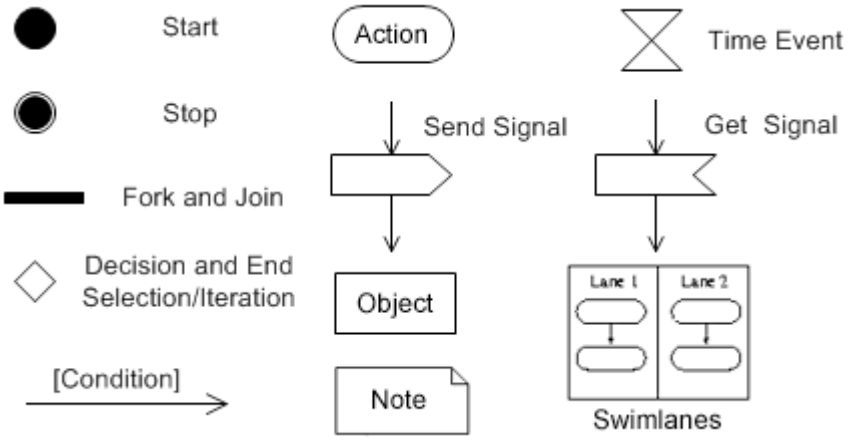


Figure 17: UML Activity Diagram elements

In figure 18 is an example of an activity diagram depicting a simple order-delivery process of some company. In the figure, once the order is received the activities split into two parallel sets of activities. One side fills and sends the order, and the other side handles the billing. On the fill order side there is also a conditional branch that selects the delivery style depending on whether the order is a priority order or not. The dashed arrows explain the UML elements that have been used in the example process.

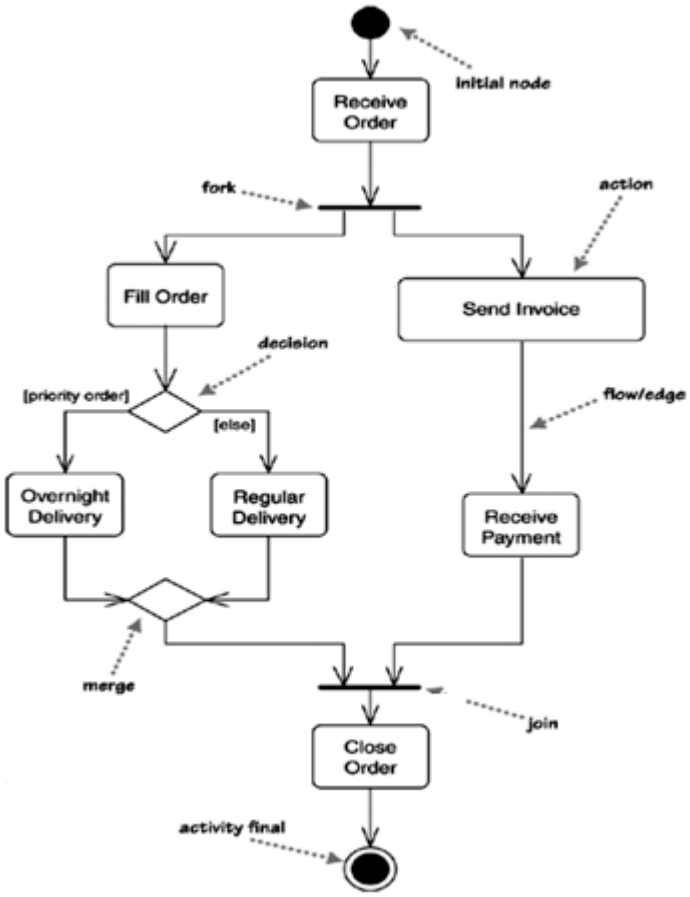


Figure 18: Order-delivery process

4.7 Differences Between BPMN and UML Activity Diagrams

In a general level, the both diagrams are very similar. The shapes look the same and both notations are also based on alike concepts. According to Eloranta [57], the representation power of the diagrams is also almost the same – BPMN only slightly more powerful. However, there are yet also some fundamental differences between the languages. Below we have listed some of the most important of them. [57][58][59]

- UML is fundamentally a general-purpose modeling language for software-intensive systems with thirteen diagram types, whereas BPMN is a domain-specific modeling language for business processes with only one diagram type.
- Although the basic shapes are very similar (e.g., rounded rectangles for activities, diamonds for decisions, etc.), they are yet named somewhat differently, for example, a Start Node in UML is a Start Event in BPMN.
- Although in BPMN there are less core elements, the total number of elements is yet bigger because of the variations of core elements. For example, the variations enable to use a multitude of different kind of gateways for controlling the flows and events in processes.
- The metamodel of UML is fundamentally more formal than the metamodel of BPMN language.
- Unlike BPMN, UML has an official file format, XMI, for serializing the diagrams in to file. BPMN does not provide any file format for the diagrams.
- UML ADs may be considered as somewhat more technical and difficult because the notation includes some object-oriented concepts. Instead BPMN diagrams may be easly designed also by non-technical people who are unaware of objects. The terms and elements of BPMN are fundamentally very down-to-earth and non IT-related.
- An advantage of BPMN over UML is that it is possible to quite easily map a BPMN diagram into an executable BPEL definition. This is not the case with UML ADs from which the mapping is more difficult.
- An advantage of UML over BPMN instead is that it is so widely used today in the software industry. Because UML has already been used for over 10 years, it has become a well-established standard. In contrast, BPMN is still an emerging standard and not as well-known as UML.
- What is notable is that the both diagrams are standardized today by the same organization – OMG. Because of this, some people have predicted that the BPMN and UML standards will merge into a single standard.

4.8 File Formats for Process Models

In this section we present three file formats that can be used to save UML and BPMN diagrams to file. Each of the formats are XML-based and define how to explicitly serialize an UML diagram or a BPMN diagram into XML. From the three file formats, BPDM and XPDL are used to save BPMN diagrams, and XMI and XPDL to save UML Activity Diagrams.

The most important reason why to use standard file formats is to avoid vendor lock-ins. It is easy to end up into a vendor lock-in situation when saving diagrams to proprietary file formats. The proprietary formats cannot be opened with any tools but only with the tool of a particular vendor. Switching from such tools to another tools is often difficult.

4.8.1 Fundamental Differences between File Formats

The common file formats for process modeling can be classified roughly into three categories: to those that capture the diagram details, those that support also the metamodel, and those that also capture the execution semantics. The classification of the file formats is thus fundamentally linked with the granularity model represented in Section 4.4. [60][61] [62]

The simplest type of a file format is such that saves only the graphical details of a diagram, or is capable of capturing only the shapes and lines of the diagram and their positions (XY coordinates) from the modeling tool's drawing canvas. XPDL is such a format, for example.

A more sophisticated format is such that also supports the metamodel of the modeling language. Such formats not only capture the diagram details but also the process semantics, or the metamodel of the diagram. Such formats enable to port process models between the tools of different vendors. This effectively reduces the risk of vendor lock-ins.

The most sophisticated file formats are those that can in addition also store the execution semantics of the model. No such file formats have been actually developed yet. Such formats would serve not only as file formats for diagrams but also as executable process definitions. It would be thus possible to execute them through a process execution engine. Execution semantics enable different engines to unambiguously execute same process definitions.

To conclude, process file formats are actually somewhat analogous with image file formats. Namely, the diagram file formats are like raster image formats (e.g., JPEG) and the metamodel-enabled file format are like vector image formats (e.g. EPS). As we know, the raster images are more difficult to be edited afterwards than the vector images because the design-time information is not saved to the file. The analogy chain can be expanded also to executable file formats that are like vector animation formats (e.g. SVG). The execution semantics can be compared to the render-time scripting information of animations.

4.8.2 BPDM

BPDM (Business Process Definition Metamodel) is an emerging file format and metamodel for BPMN by OMG. The standard has not been yet officially released (as of 10/2007) but is expected to be released very soon (finalization underway). [46]

The main role of BPDM will be as the “official” BPMN file format. BPDM specifies how to serialize unambiguously a BPMN diagram into XML (by means of XML schema). Until today, BPMN design tools have been relying on proprietary file formats (of tool vendors) because BPMN standard has been lacking an official file format. This has meant the diagrams that have been saved in one tool could not have been opened in other tools. A partial solution for this has been to use XPDL format but the problem with it is that it only saves the diagram but not the metamodel. [60]

The other important role of BPDM is going to be as the formal metamodel for BPMN. As such, it specifies the explicit semantics for BPMN diagrams. It also enables BPMN diagrams to be transformed into other diagrams such as UML Activity Diagrams. Metamodels were discussed earlier in Section 4.2. [60]

It should be noted that BPMN and BPDM are actually going to merge in the next version of BPMN. The version 2.0 of BPMN is going to be based entirely on BPDM. When released, the current BPDM standard will become deprecated. [60]

4.8.3 XPDL

XPDL (XML Process Definition Language) is a file format for several types of process diagrams. It is a standard by WfMC (Workflow Management Coalition) and was released in 2002. In XPDL specification (latest version 2.0) is specified how to serialize from 101 different diagram types to XML. Although most of the supported 101 diagrams are those of proprietary tools, common standards such as UML and BPMN are yet also supported. It is notable that XPDL is currently the only noteworthy format for serializing BPMN into XML. [61][62]

The main goal of XPDL is to provide interchangeability of diagrams between different tools, in the diagram level. XPDL saves essentially only the diagram shapes and their coordinates but what it does not save is metadata. Thus, it can only promise that the diagram look the same in different design tools. Use of other formats (proprietary) is required to save also the design-time metadata. [62][101]

There is however a small workaround to save also the metadata using XPDL, namely, by saving it to the extended attributes of the diagram elements. This technique has been even used to save execution semantics to XPDL (at least the Fujitsu Interstage execution engine is capable of taking XPDL directly). However, the problem with this approach is that such definitions are no longer standard-based and thus nor portable. This approach works only with a specific tool-engine combination. [62]

4.8.4 XMI

XMI (XML Metadata Interchange) is the “official” file format of UML. It defines the serialization of UML models into XML. It is a standard by OMG (like UML) and today a standard that is adopted by a large number of vendors providing UML tools. In the UML-XMI specification is specified with pseudo-code rules and XML Schema how to unambiguously transfer any type of UML diagram into XML. [46]

Because XMI enables the standardized serialization of UML diagrams into XML, it is a single good argument why to use UML Activity Diagrams for business process modeling instead of BPMN. As we remember, BPMN does not provide an official file format (yet).

It should be noted that XMI is used also by other modeling languages than UML for XML serializations. This is because XMI has been originally developed to be a universal XML serialization for all MOF-compatible modeling languages. MOF (Meta Object Facility, by OMG) is fundamentally a modeling language for defining modeling languages. The MOF may be thus thought as the meta-metamodel of UML. [63]

5 Business Process Execution Language (BPEL)

In this chapter we discuss the Business Process Execution Language (BPEL). We first introduce some basic concepts and features of the BPEL programming language and then discuss how it may be used in practice to automate a business processes in enterprises. We also discuss BPEL editors, some known shortcomings of the language, and the developed extensions to address these shortcomings.

5.1 Overview

Business Process Execution Language (BPEL, WS-BPEL, BPEL4WS) is an XML based programming language used for the definition and execution of business processes using Web Services. BPEL may also defined also as an orchestration-style Web Service composition language that provides a way to compose several Web services into new composite services or business processes. Today, BPEL represents the de facto language of the software industry for specifying executable business. [64][65][100]

BPEL is fundamentally an EAI technology because it standardizes the integration logic and the process automation between the Web services of an enterprise. As such it allows to integrate enterprise applications, which have been exposed as Web Services, from the perspective of a business process. In addition, BPEL can be considered also as a B2Bi technology because it allows to integrate applications also between enterprises.

As a programming language BPEL is comparable to any conventional programming language such as Java because it is Turing complete. BPEL language supports many of the basic features of programming languages including creating sequential or parallel process flows, setting conditional expressions, constructing loops, declaring variables, copying and assigning values, and defining exception handlers. In addition, BPEL also supports some sophisticated business process features such as long-running transactions and run-time events. However, in regard to the number of programming constructs, BPEL is much simpler than the common programming languages. [21][64]

The main benefit of BPEL is fundamentally that it is an open standard and may be used under royalty-free terms. The openness means BPEL is not dependent on certain platforms such as .NET or J2EE, or programming languages such as Java or C#. Instead it is based on neutral Web Service and XML standards. The openness allows easy portability of BPEL definitions between different tools or execution engines. [24]

The first version of BPEL was released in 2002 by a cross-company consortium consisting of IBM, BEA and Microsoft. The first version was born out of the merger of IBM's WSFL and Microsoft's XLANG specifications. The version 1.1 was released in March 2003 after SAP and Siebel had joined the consortium and proposed several modifications. In April 2003 BPEL was submitted to OASIS (Organization for the Advancement of Structured Information Standards) where a new technical committee for BPEL was formed. Since 2003, over 200 other companies have joined the committee. The latest version (BPEL 2.0) was published in spring 2007. [24][64][65]

5.2 Technology

In technical level, BPEL is about composing Web Services into new (composite) Web Services. Each BPEL process is fundamentally a composite service that invokes a set of other Web Services in order to provide the expected outputs. The composition logic of a BPEL service represents the business process logic. In the composition logic is defined the exact order in which the involved services are invoked and how their responses are aggregated to the output of the process. [21][64]

BPEL is a WS composition language layered on top of WSDL. All the inbound and outbound process interfaces are described in WSDL 1.1. The data model of BPEL is built on top of XML Schema 1.0 types. XML Schema is used, inter alia, to specify the structures of messages within processes. XPath 1.0 is used for data manipulations in BPEL processes, e.g., to extract data from messages or to define conditions of conditional statements. The communication with the external world takes place through SOAP protocol. [24]

Figure 19 depicts a BPEL process definition and the corresponding process diagram (or workflow). One of the main benefits of BPEL is that the process definitions may be depicted as visual diagrams in the design-time tools. The participating Web Services are called as partners in BPEL processes. The process tasks or the basic units of work within BPEL processes are called as activities. Activities represent essentially the basic building blocks of BPEL processes. 64]

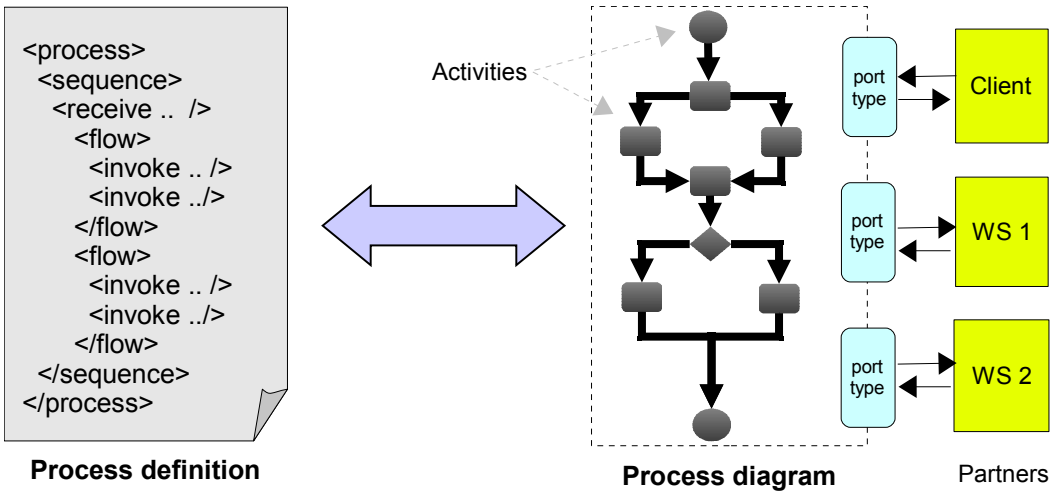


Figure 19: BPEL process and the corresponding workflow diagram

In order to be able to execute BPEL processes, it is required to utilize a BPEL execution engine. Execution engines are systems capable of executing the logic of BPEL processes. They are usually complex server-side systems and consist of a large number of deployed processes. 66]

Lastly, it should be noted that it is possible to build BPEL processes recursively. As mentioned earlier, BPEL processes are also Web Services themselves and having their corresponding WSDLs. This means it is possible to use BPEL processes as building blocks in other BPEL processes. This is the only modularization technique supported by the BPEL language, namely, subprocesses or other kind of subroutines are not supported by BPEL. 67]

5.3 Features

In this section we present some of the most interesting features of BPEL (2.0). We are not going to discuss all the features but only about the most interesting ones that differentiate the BPEL language from the conventional programming languages. The features have been gathered from [64], [68], [69], [70], and [100].

Service-based modularization – The most significant feature of BPEL is probably that modularization is achieved through Web Services. Unlike in traditional programming languages, there are no such things as sub-routines or function calls in BPEL. Instead, the only way to simulate sub-processes in BPEL is basically to outsource them as Web Services. The outsourced Web Services are then implemented as yet another (service-exposed) BPEL processes. Using this kind of recursive service composition approach, it is possible to build arbitrary complex business processes. This approach has yet also its shortcomings (see Section 5.6).

Synchronous & asynchronous processes – Two kind of BPEL processes are possible: synchronous and asynchronous. Synchronous processes are such that immediately replies the client when invoked. This requires the client is blocked until the results are returned. Asynchronous processes are instead such that can take a long time to complete and do not block their clients on calls. The process makes a callback on the client only when the results are available. The disadvantage of using asynchronous communication pattern is that clients become more complex as they are required to implement a callback interface (requires the use of threads). The server's role is also more complicated as it is required to record process states and keep track of different discussions. In BPEL, there are also some "partly asynchronous" interaction patterns, e.g., a request with an explicit response, a request with no response, and a one-way notification.

Two programming models – BPEL supports two different programming paradigms: structured and graph-based. While in structured paradigm the control flow is defined by nesting activities into each other, in graph-based paradigm the activities are connected via graph-based links that are like "goto" instructions of where to go next in the process. In graph-based paradigm, the links may have boolean expressions assigned to them called transition conditions that decide if the pointed activity should be executed or not. Both paradigm are computationally equivalent. In practice, the structured paradigm is more often used.

Scopes – Like any traditional programming language, also BPEL supports variable scopes. Scopes split processes into parts in which it is possible to define variables with a limited visibility. Scopes may be defined in BPEL with <scope>. Nested scopes are also possible. In addition, to every scope it is possible to specify separate fault and event handlers.

Exception handling – Exception handling mechanism is implemented in BPEL in similar way as, e.g., in Java. Exception handlers are specified to scopes using the <faulthandler> block. Exceptions, or faults in BPEL terms, are thrown using the <throw> activity and caught using the <catch> or <catchall> activities. Faults may be caught based on fault names, types of the containers they have been associated with, or a combination of them. BPEL also supports standard scope based escalation of uncaught faults and the re-throwing of faults.

Transactions – Transaction mechanism work hand-in-hand with exception handling in BPEL. Transactions are implemented in practice using scopes and fault handlers. To exemplify, if a sequence of activities is wanted to be grouped into a transaction then it must be surrounded with scope tags. The scope tag tells the steps enclosed in it that they should either all complete or fail. If any of these steps fail then the scope's fault handler, or the compensation handler, is invoked. The compensating actions specified to the fault handler are fundamentally instructions of how to recover from a failed transaction. It should be noted that the transactions of BPEL processes are usually rather long-running because many business transactions often take a long time to complete.

Concurrency and non-determinism – Building simple concurrent flows in BPEL processes is relatively easy. Concurrent flows are built in BPEL using the <flow> activity. To express more complex concurrency scenarios, <flow> provides also the ability to express synchronization dependencies between activities. For example, with <flow> it is possible to specify that a certain activity or several activities cannot start before some another activity has finished.

Event handlers – BPEL supports the handling of two kind of events that may occur during a process is being executed: incoming messages and timed alarms. Message events are those that occur when partners send messages to running processes. Alarm events are those that are triggered at a specific time or after a timeout has expired. In BPEL event handlers are defined using <receive> or <pick> activities.

Correlation - Correlations are used to match an inbound message to a right process instance running on the server. This is required because there may be running simultaneously several instances of the same process in a BPEL engine. Correlation is about identifying the different conversations the server is involved in. The right conversation for each inbound message is found by using correlation identifiers in messages. The identifiers may be thought like cookies used in many Web pages. Correlations are defined in BPEL with <correlation> and <correlationSets> tags. At run-time, the matching of identifiers is automatic. The BPEL engines leverage WS-Addressing for this purpose. Only asynchronous processes have to be correlated because in synchronous processes the conversation context is maintained on the stack or across a TCP connection.

Abstract processes – In addition to the usual executable processes, in BPEL it is also possible to define abstract processes. Abstract processes define only the externally visible behavior of a business process. They are used merely to describe the public behavior of the process to the process partners (to partner enterprises). In B2B scenarios, the abstract BPEL processes may serve as contracts of message sequencings between the partner enterprises.

5.3.1 Original BPEL Design Goals

The development of the BPEL language was guided by ten fundamental design goals which are summarized in the following list. They are listed here because we think they summarize well the original intentions of the developers of the BPEL language. [65]

1. Define business processes that interact with external entities through Web Service operations defined using WSDL 1.1, and that manifest themselves as Web services defined using WSDL 1.1. The interactions are “abstract” in the sense that the dependence is on portType definitions, not on port definitions.
2. Define business processes using an XML-based language. Do not define a graphical representation of processes or provide any particular design methodology for processes.
3. Define a set of Web service orchestration concepts that are meant to be used by both the external (abstract) and internal (executable) views of a business process. Such a business process defines the behavior of a single autonomous entity, typically operating in interaction with other similar peer entities. It is recognized that each usage pattern (i.e. abstract view and executable view) will require a few specialized extensions, but these extensions are to be kept to a minimum and tested against requirements such as import/export and conformance checking that link the two usage patterns.
4. Provide both hierarchical and graph-like control regimes, and allow their use to be blended as seamlessly as possible. This should reduce the fragmentation of the process modeling space.
5. Provide data manipulation functions for the simple manipulation of data needed to define process data and control flow.

6. Support an identification mechanism for process instances that allows the definition of instance identifiers at the application message level. Instance identifiers should be defined by partners and may change.
7. Support the implicit creation and termination of process instances as the basic lifecycle mechanism. Advanced lifecycle operations such as "suspend" and "resume" may be added in future releases for enhanced lifecycle management.
8. Define a long-running transaction model that is based on proven techniques like compensation actions and scoping to support failure recovery for parts of long-running business processes.
9. Use Web Services as the model for process decomposition and assembly.
10. Build on Web services standards (approved and proposed) as much as possible in a composable and modular manner.

5.4 How to Develop a BPEL Process

This section is a short guide how to automate business processes in practice using BPEL. Because the development process of a BPEL definition is fundamentally a gradual process, this section explain this process step by step. These same steps needs to be taken basically every time a new BPEL definition is to be developed. This section is based on BPEL tutorials of [21], [65], [70], [71] and [72].

We are going to develop a simple currency converter BPEL application. The application's purpose is to convert a given amount of euros to dollars. Figure 20 depicts the process diagram of the application. The Bank Web Service's role is to provide the process with the current current euro to dollar exchange rate. The given amount of euros is then multiplied with the exchange rate to get the amount of dollars. The amount of dollars is then returned as the process output.

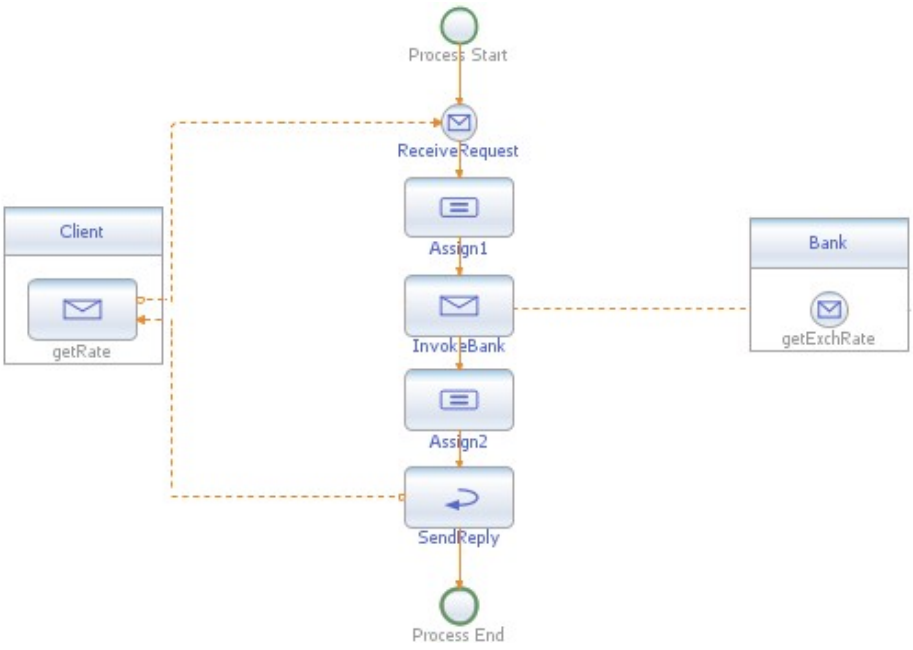


Figure 20: Currency conversion process

Step 1 – Create a skeletal process

As a first thing we create a skeletal BPEL process (a new .bpel file) that serves as the basis for any further definitions in the later steps. Figure 21 depicts a standard skeleton process that can be used as a starting point for any new BPEL processes. The common code blocks to all BPEL processes include (in addition to the <process> block) <import>, <partnerLinks>, <variables>, and <sequence> blocks. <import> statements are used to import WSDLs and other required artifacts into the process; <partnerLinks> to define the entry and exit points of the process; <variables> to define variables that are needed to pass data forward when the process is being executed; <sequence> to define the actual process logic (or orchestration logic).

```
<process>

   <import/>
       <!-- file imports (WSDLs, XML Schemas, etc.) -->

   <partnerLinks>
       <!-- clients and Web Services involved in the process -->
   </partnerLinks>

   <variables>
       <!-- definitions of process variables -->
   </variables>

   <sequence>
       <!-- the actual process logic -->
   </sequence>

</process>
```

Figure 21: Skeletal currency conversion BPEL process

Step 2 - Inventory the involved Web Services

It is important that before we go further we first inventory the Web Services that we are going to orchestrate in our BPEL process. If we are going to orchestrate external Web Services, we should download and store their WSDLs. Also, if we have not already exposed our internal applications as Web Services, we should do it now. To avoid difficulties in the later steps, it is also important that we gain a clear understanding to the invocation details of the Web Services already at this point. In our example we are orchestrating only one Web Service – the Bank service.

Step 3 – Develop a WSDL for the process

In this step we create a WSDL definition for our process. The WSDL describes the interface of the process to the outside world. It is wise to define it before defining the BPEL process logic. In the WSDL file we define essentially the operations the process is to provide for its clients and the input and output messages of the operations. In the WSDL we may also specify whether we want to use a synchronous or asynchronous (using a callback) communication style. The WSDL of the example process is shown in Figure 22. It is fundamentally very simple because it consists of only one port type, one operation (getRate), and two messages (getRateRequest, getRateReply). The role of the getRate operation is to bound the two messages into a simple synchronous transaction.

```
<definitions
    targetNamespace="http://hiit.fi/Piispanen/CurrencyConverter/WSDL"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:plink="http://schemas.xmlsoap.org/ws/2004/03/partner-link/"
    xmlns:tns="http://hiit.fi/Piispanen/CurrencyConverter/WSDL">

    <types/>

    <message name="getRateRequest">
        <part name="euros" type="xsd:float"/>
    </message>
    <message name="getRateReply">
        <part name="dollars" type="xsd:float"/>
    </message>

    <portType name="CurrencyConverterPortType">
        <operation name="getRate">
            <input name="input1" message="tns:getRateRequest"/>
            <output name="output1" message="tns:getRateReply"/>
        </operation>
    </portType>

    <binding name="CurrencyConverterBinding" type="tns:CurrencyConverterPortType">
       <!-- simplified -->
    </binding>

    <service name="CurrencyConverterService">
       <!-- simplified -->
    </service>

    <plink:partnerLinkType name="CurrencyConverterPartner">
        <plink:role name="provider" portType="tns:CurrencyConverterPortType"/>
    </plink:partnerLinkType>

</definitions>
```

Figure 22: WSDL of the currency conversion process

Step 4 – Define namespaces

Next, we should define the namespaces to distinguish the elements belonging to different entities and languages in the BPEL process definition. Namespaces must be specified for all BPEL processes because they may involve several WSDLs and other code blocks, such as XML Schemas or XPath expressions, that must be differentiated from each other. Namespaces ensure that different kind of elements may be referenced unambiguously within a process definition without having to worry about conflicting element names. To exemplify, namespaces can be considered as a similar concept to packages in Java. Namespaces (and their prefixes) are defined in the beginning of the process, in the <process> tag.

Step 5 - Define Partner Link Types

Partner link types are used to define the relationship of the BPEL process and its partners (involved clients and external services). Partner link types are defined to the WSDLs of BPEL processes. In a single partner link type is defined what role the process plays when it interacts with one of its partners. A BPEL process may be either a service provider or a service consumer, or sometimes both. To exemplify partner link types, in a partner link type BuyerSellerLink could have been defined whether the process is either a buyer or a seller when it interacts with one of its partners. In synchronous operations, there has to be only a single role in the partner link type because the operation is invoked in a single direction only. In asynchronous callback operations both roles must be used. In our example there have been specified two partner link types, one for the process and one for the bank (see the WSDLs).

Step 6 – Define Partner Links

Partner links (<partnerLink>) are implementations of partner link types. They are defined to the actual BPEL process definition. Partner links define what external Web Services may be invoked from the process but also the clients of the process. If any external Web Service operations need to be invoked from a process, they are always consumed through partner links. Partner links can be considered as entry and exit points of processes. In our example we have only two partner links: one for the client and one for the bank.

Step 7 – Declare process variables

Process variables allow the data to flow from one activity to another in BPEL processes. The variables are used to store the incoming messages to an activity, and to build the outgoing messages from the activity. The variables can be considered also as maintaining the process state. The variables in BPEL are XML containers that hold XML structures inside them. Variables may be defined in terms of WSDL message types, XML Schema types, or XML Schema elements.

In BPEL, every variable is declared in a certain scope. A variable that has been declared in the highest scope (global) is visible through the entire process. Variables may be assigned to other variables in BPEL using the <assign> activity. XPATH may be further used to extract fragments of data from the contained XML structures.

In our example, we have declared four variables: getRateIn to store the invocation message of the client; getExchRateIn to construct the invocation message for the bank service; getExchRateOut to store its result, and getRateOut to construct the response message for the client of the process.

Step 8 – Write the actual process logic

Now, after we have built the foundation for our BPEL process it is time to develop the actual process logic. The process logic is built using the activities of BPEL. The activities represent the basic programming constructs of the BPEL language. Each BPEL definition consists of a bunch of different activities that act like the individual steps of the process. In Table 3 is shown a set of most commonly used BPEL activities.

In BPEL, the activities can be divided into three categories: to basic, structured, and special activities. Basic activities are conceptually simple behaviors like receiving a message, invoking a Web Service, and assigning values to variables. Structured activities are used to group basic activities, e.g., they allow to specify conditional statements and loops within BPEL processes. Special activities introduce variable scoping and handle abnormal activities such as process termination and compensation.

Figure 23 depicts the completed BPEL process definition of the Currency Converter example. One should compare it with the skeletal process of Figure 22. In the completed process we have used one structured activity (<sequence>) and four basic activities (<receive>, <invoke>, <assign>, and <reply>). Each of the activities have been further qualified with several attributes.

Basic activities	**Structured activities**	**Special activities**
<invoke> - invokes an operation of external Web services **<receive>** - waits for a message from external source **<reply>** - used together with receive to reply results to external source **<assign>** - the assignment of values to variables **<throw>** - throws faults to fault handlers in some scope **<catch>** - catches a fault by faultName in fault handler **<wait>** - waits until a given point in time or for a specified duration ... (see the BPEL specification for full list)	**<sequence>** - defines the sequence of execution of activities **<flow>** - parallel execution of activities **<pick>** - consists of a set of defined events that blocks the execution of the process until a message event or timed event has occurred. **<while>** - supports the repeated execution of activities **<switch>** - supports the conditional behavior	**<scope>** - defines a new scope for variables, fault handlers, and compensation handlers **<compensate>** - invokes compensation on an inner scope that has already completed normally **<terminate>** - immediately terminates a business process instance

Table 3: BPEL activities

Step 9 – Create a deployment archive and -descriptor

After we have completed the process definition we must package it to a so called deployment archive to be able to deploy it to an execution engine. Into a process archive is usually placed also the the other necessary deployment artifacts including WSDLs, XSDs, and XSLTs. Because most engines have been implemented in Java, the archive files are usually some sort of Java archives such as JARs, WARs, or similar.

In order to deploy the process, we need to also develop a so called deployment descriptor for the process archive. The deployment descriptor is a simple XML file that contains deployment-time instructions for the engine, and in which is cataloged the different deployment artifacts (involved files). For example, it may instruct the engine of artifact file locations, protocol bindings, default process inputs, etc. It is notable that deployment descriptors are not covered by the BPEL standard but are specific to BPEL engines.

Step 10 – Deploy and test

The process archive may be deployed to the engine usually from many places. Most engines support at least deploying from the server's command line and administration console. It is also often possible to deploy processes directly from the BPEL editors they have been designed in. Deploying directly from the editor is the easiest way because it may often done only with one click.

During deployment BPEL engine loads files from the deployment archive based on the properties specified in the archive's deployment descriptor. After reading the descriptor and files, BPEL engine is enabled to use namespace addresses (rather than file addresses) of the elements of the deployment artifacts.

After we have deployed our process successfully to the engine, it is time to test our process. Testing of BPEL processes is usually quite easy because the majority of today's BPEL platforms provide a bunch of sophisticated testing facilities. Such facilities include visual debugging of processes, process simulation, execution histories, complex testing frameworks, etc. The testing is most often performed in the BPEL editors the processes have been designed in.

```
<process
   targetNamespace="http://hiit.fi/Piispanen/CurrencyConvertor/BPEL"
   xmlns="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:ns1="http://hiit.fi/Piispanen/Bank/WSDL"
   xmlns:ns2="http://hiit.fi/Piispanen/CurrencyConverter/WSDL">

   <import location="Bank.wsdl"/>
   <import location="CurrencyConverter.wsdl"/>

   <partnerLinks>
      <partnerLink name="client" partnerLinkType="ns2:currencyConverter"
       myRole="provider"/>
      <partnerLink name="bank" partnerLinkType="ns1:bank" partnerRole="provider"/>
   </partnerLinks>

   <variables>
      <variable name="GetRateIn1" messageType="ns2:getRateRequest"/>
      <variable name="GetRateOut1" messageType="ns2:getRateReply"/>
      <variable name="GetExchRateIn1" messageType="ns1:getExchRateRequest"/>
      <variable name="GetExchRateOut1" messageType="ns1:getExchRateReply"/>
   </variables>

   <sequence>
      <receive partnerLink="client" portType="ns2:CurrencyConverterPortType"
           operation="getRate" variable="GetRateIn1"/>
       <assign name="Assign1">
          <copy>
              <from>'EUR'</from>
              <to part="currency1" variable="GetExchRateIn1"/>
          </copy>
          <copy>
              <from>'USD'</from>
              <to part="currency2" variable="GetExchRateIn1"/>
          </copy>
       </assign>
       <invoke partnerLink="bank" portType="ns1:bankPortType"
          operation="getExchRate"  inputVariable="GetExchRateIn1
         outputVariable="GetExchRateOut1"/>
       <assign name="Assign2">
          <copy>
              <from> ( $GetRateIn1.euros * $GetExchRateOut1.rate ) </from>
              <to part="dollars" variable="GetRateOut1"/>
          </copy>
       </assign>
       <reply partnerLink="Client" operation="getRate"
          portType="ns2:CurrencyConverterPortType" variable="GetRateOut1"/>
   </sequence>

</process>
```

Figure 23: Completed currency conversion BPEL process

5.5 BPEL Editors

A BPEL editor is just what the name says: an editor for BPEL code. The editor facilitates developer's work by providing a graphical environment for designing complex business processes. Writing BPEL code by hand is often considered as not only unintutive and error-prone but also very slow. The editors are usually aimed at both IT people and business people, or basically to all people who deal with business processes in today's enterprises. [64][73][74]

The editors are normally provided as plug-ins for common programming IDEs (Integrated Development Environment) such as Eclipse. They are only very rarely provided as stand-alone programs. The editors and IDEs are usually further bundled with the engines and other tools to large-scale software packages, known as BPM systems (BPMSs). Acquiring the editor in separate of the engine and other tools is not a common practice. In Figure 24 is shown a screenshot of a contemporary BPEL editor. [74]

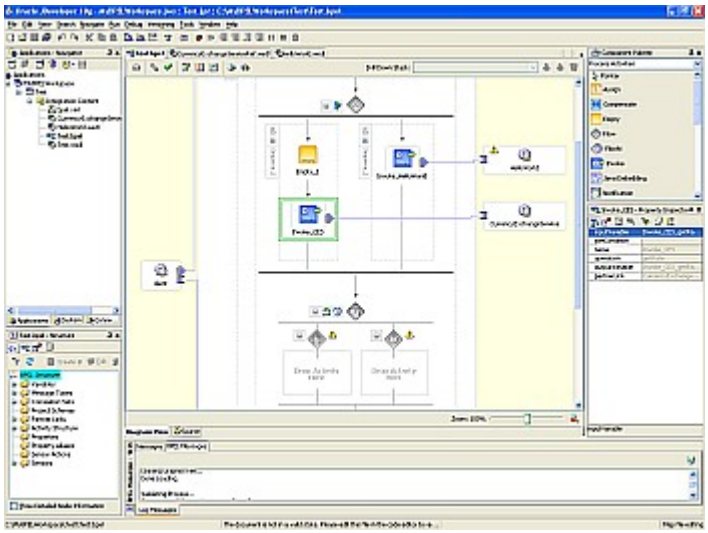


Figure 24: Oracle JDeveloper BPEL Designer

A process that has been developed in one vendor's editor may usually be deployed only to the same vendor's engine. Manually deploying processes to other engines is of course possible but is time-consuming and difficult in the long run. The coupling of editors and engines is mainly because the vendors get their biggest revenues from the execution engines.

The main component of the editor is the process canvas in which the process is built by drag-and-dropping BPEL activities from the activity palette. Some editors require the activities have have to be connected with connecting lines. The activity properties are usually specified in a separate property views or window. The graphical notation in which the process definitions are shown in editor canvases varies quite much. Many of the editors are relying on their respective notations, only some are supporting more common notations such as BPMN.

A completed BPEL process may be deployed from the editor to the engine usually by just one click. For this, the editor is required to automatically generate a deployment descriptor for the process which contains the deployment instructions for the engine.

The testing and debugging of deployed processes is also often possible in the editors. The debugging of BPEL processes is done in similar way as debugging of normal programming code. To support the debugging, the most editors utilize the built-in debugging capabilities of the IDEs they are built upon. In many editor the debugging may be done in a diagram view. This is often referred as visual debugging of BPEL processes. [73][74]

In Table 4 is shown a few BPEL editors that we have evaluated for the purpose of this thesis. The editors we have tested are either freeware/Open Source or such that may be evaluated for free. The vendors who are providing BPEL editors and tools include some large enterprise system vendors such as IBM, Microsoft, Oracle, Sun, and BEA. Today there are also some Open Source communities developing BPEL tools, e.g., Netbeans, ActiveBPEL and Apache ODE.

During our evaluation, we found out that BPEL editors are not always so similar to each other. There were some quite big differences in the feature sets of the editors. From the tested editors, the majority provided at least a process drawing canvas, support for drag-and-dropping activities, wizards for importing Web Services, a WSDL editor, and an expression editor for building XPath expressions. Lesser editors were providing facilities for visual debugging, process simulation, manual source code editing, or code validation. Only a few editors were supporting audit trails or Java code snippets within process definitions. We also discovered that many of the Open Source editors are actually very good, e.g., Netbeans was probably the best editor we tried.

BPEL Editors (as of 5/07)	Netbeans + Enterprise Pack	Oracle BPEL Designer For Eclipse	Oracle JDeveloper BPEL Designer	IBM BP4WSJ Editor	ActiveBPEL Designer	Eclipe BPEL Designer
Vendor	Sun + Netbeans Community	Oracle	Oracle	IBM	Active Endpoints	-
Type of software	Netbeans IDE plugin	Eclipse IDE plugin	JDeveloper IDE plugin	Eclipse IDE plugin	Eclipse IDE plugin	Eclipse IDE plugin
Current status	Discontinued	Discontinued	Under development	Discontinued	Under Development	Future unclear
Latest version	5.51 (2007)	2.2 (2004)	10.1.2.0.2 (2007)	2.1 (5/2004)	3.0 (2007)	0.01 (2006)
License	Freeware	Open Source	Freeware	Commercial	Freeware	Open Source
BPEL version supported	2.0 (partially)	1.1	1.1	1.1	2.0	1.1
Coupled with engine	Sun Java Application Server BPEL Engine, Glassfish	Oracle BPEL Process Manager	Oracle BPEL Process Manager	-	ActiveBPEL Engine, ActiveBPEL Enterprise	-
Debugging	Yes	No	No	No	Yes	No

Table 4: BPEL editors

5.6 Shortcomings

Although the latest version of BPEL (2.0) has been just released, the standard is still considered of having a few unsolved problems and lacking some features that would make it more useful in practice. Below we have listed some known issues (according to [75], [76], [77], [78]):

Data manipulation – Writing small code snippets, e.g., to manipulate data, calculate dates, construct loops, and prepare conditional statements, is hard to accomplish with BPEL. Despite BPEL has not been originally intended as a general-purpose programming language, it would be still very useful to be able to write at least some sort of small code snippets. Currently, in BPEL it is possible to write small code snippets only with XPath. XPath is yet a language for extracting data from XML structures rather than writing small code snippets. Because of the limitations of XPath, it is actually often required to outsource the code snippets as yet another Web Services. This means a new Web Service is developed for the code snippet which is then then invoked from the process like any other Web Service. This approach works but is not very elegant because the process logic gets scattered across a number of Web Services. It also causes performance overhead.

Human Tasks – In practice, the majority of business processes usually incorporate some sort of human involvement, e.g., processes may be dependent on the actions performed by humans or the information they provide. A typical human task in many business processes is that a human is required to approve of something. Currently, BPEL does not have a built-in support for human tasks which may be considered as a shortcoming. The only way to support human tasks in BPEL is to outsource them as Web Services.

Portability – BPEL promises to be portable in the sense that one should be able to port process definitions between different tools. Furthermore, most of the today's BPEL vendors also promise that their tools are standard-based and do not cause vendor lock-ins. The portability of BPEL processes is however not always so clear because of the proprietary extensions the many vendors have developed for the language. The vendors have developed their proprietary extensions mainly because of the reasons mentioned in this section.

Not backward-compatible – The latest BPEL version 2.0 is considerably different from the previous version 1.1 and unfortunately not backward-compatible. Although most of the changes are syntactic, there are yet a few bigger semantic changes, too. The semantic differences refer to how BPEL 2.0 deals with links, messaging, compensation handling, and data manipulation. Migrating from BPEL 1.1 to BPEL 2.0 is can be thus tricky.

Fault handling – The problem of BPEL fault handling is that it does not support system faults that are common in practice. A system fault occurs typically in a situation where one tries to invoke an unavailable Web service. BPEL currently supports catching of such faults only with the catchAll construct. However, because catchAll has no way of distinguishing between different faults, handling of system faults is often complicated and error-prone.

Subprocesses – According to many, any business process definition language should be such that supported modularization and re-use through subprocesses. However, currently BPEL language does not provide any facilities to support subprocesses. This is often considered as one of the main shortcomings of BPEL language. The only way to simulate subprocesses is to outsource them as Web Services. The problem with this approach is yet that such "subprocesses" are completely hidden from their parent processes. It is also difficult to coordinate the lifecycles of such processes together with their parent processes.

5.7 Extensions

In the previous section we discussed about the shortcomings of BPEL. In this section we introduce a few BPEL extensions that have been developed to address the above problems.

5.7.1 Data manipulation: BPELJ

As mentioned, writing small code snippets is often difficult with BPEL. To enable more sophisticated data manipulations, there has been developed an extension called BPELJ that allows programmers to use Java code snippets in BPEL processes. Java snippets are useful in BPEL processes with activities such as <assign>, <while>, <switch>, and <wait> . Java snippets are also useful with links and joins. However, it should be mentioned that using Java in BPEL processes violates the original idea of BPEL being a vendor-agnostic programming language. BPELJ can thus never become a part of the official BPEL specification. [79]

5.7.2 Human tasks: BPEL4People

BPEL4People (WS-BPEL Extension for People) extends BPEL to support also human tasks. The extension was announced in late 2005 by IBM and SAP. BPEL4People introduces the people activity as a new type of basic activity which enables the specification of human interactions in processes in a more direct way. The people activity allows to specify the humans tasks and to assign them to different roles. At run-time, when an execution engine encounters the people activity, it suspends the process until the person has completed the associated task. In order to resume the business process the user or the "task list agent" needs to notify the engine of the completed task. [80][81]

5.7.3 Subprocesses: BPEL-SPE

BPEL-SPE (BPEL 2.0 Extensions for Sub-Processes) is a proposal from IBM and SAP how to enable sub-processes in BPEL. The proposal calls for the explicit definition of subprocesses, allowing for the reuse of subprocesses within the same BPEL process or across multiple processes. The subprocesses defined with the aid of BPEL-SPE are also allowed to be bound to their parent processes and access the data in them. [82]

6 Business Process Run-Time Software and Tools

In this chapter discuss business process execution environments or information systems capable of executing business processes. We are going to present the Workflow Management Systems, Business Process Management Systems, Enterprise Service Buses, and BPEL engines. We are also going to discuss the concept of Business Activity Monitoring (BAM) and introduce and an Open Source BPM platform called Netbeans.

6.1 Workflow Management Systems (WfMS)

Workflow management systems (WfMS) are software systems aimed to support the management and automation of workflows. They are traditional enterprise systems that have been used already for over 20 years in many enterprises. WfMSs are often considered as predecessors of today's Business Process Management Systems (BPMS). [38][40]

According to Workflow Management Coalition (WfMC) [42], an organization that standardizes workflow automation, a WfMS is defined as "a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications".

Fundamentally, WfMSs aim at centralizing the workflows of an enterprise into a single system, i.e., in a similar way the database management systems centralized the management of data. This enables to more efficiently manage the workflows and to more easily implement changes to them when there is a change in the conditions. The centralized management of workflows appears often also as improved workflow efficiency, better productivity, reduced costs, increased security, and enhanced quality. [23][40][83]

Traditionally, WfMSs have been used in enterprises mainly to automate people-to-people workflows or to route information between employees. WfMSs have not been used to integrate applications because it has been traditionally thought that applications should be integrated with EAI systems instead. This is the main reason why applications have usually had only a supportive role in many of the WfMSs. However, as today the WfMSs are moving towards EAI systems, many WfMSs have started to adopt also more sophisticated EAI capabilities. [40]

The history of WfMSs dates back to the 1980s. The early workflow management systems were document management systems with a workflow support. They enabled to specify the routings of office documents between workers in offices. The early WfMSs can be thought of having started the era of modern business process automation. Since the early days, the WfMSs have evolved in a great way and become more generic systems to automate all kind of workflows. Today's BPM systems represent the latest step in this evolution. [37][38][83]

What differentiates the traditional WfMSs from the today's BPM systems is that they are fundamentally having a smaller scope. Where WfMSs have been usually focused on managing processes mainly at a workflow level, BPMSs are more focused on managing large-scale business processes that are enterprise-wide. Another difference is that most WfMSs are based on proprietary technologies, while the BPMSs are based on open standards such as XML or Web Services. The WfMSs neither provide a support for process lifecycles similarly as BPMSs. [37][84]

6.1.1 Components

The core component of each WfMS is a workflow engine that is responsible for executing the workflow definitions. The engine's role is to pass data from one process participant to another in a correct sequence, and ensure the participants fulfill the required contributions. The routing of tasks is performed according to inter-task dependencies. Some engines can proactively inform users of new tasks, warn them of late tasks, or re-route tasks from one user to another in abnormal situations. The execution speed is usually not a major issue with workflow engines. The execution may be often suspended while waiting for a process participant to complete a task. Furthermore, it is notable that the engine component actually executes workflow instances rather than "workflows". The idea is to enable the same workflow to be instantiated many times at the same time. [23][37][38][40]

Another important component of a WfMS is the workflow designer. It is a graphical tool that allows (business) users to design workflows in an intuitive way and to specify the associated tasks and routing rules. The designers usually represent the workflows as diagrams or directed graphs whose knots represent work tasks. The details of the tasks are encapsulated in the knots and are not of interest of WfMSs. They can be thought like "black boxes" within the definitions. [38][40][84]

WfMSs also consist of client programs that connect the employees to workflows. They are like workflow inboxes for employees to receive new tasks. They are either stand-alone e-mail type of programs or implemented as Web applications. [38]

According to Khan [38], many WfMSs also include a roles manager and facilities for monitoring and exception handling. Roles manager manages the roles to whom each workflow task is assigned. Usually, it is better to assign tasks to roles rather than individuals because the people's job functions tend to change often. Exception handling facilities are needed to handle exception occurring commonly during execution. An exception may occur, for example, when an employee is unable to complete a task on schedule, and the task needs to be reassigned to some other employee. The monitoring facilities allow managers to follow the progress of work and gain information about the timeliness and cost-effectiveness of workflows.

6.1.2 WfMSs today

Today WfMSs represent somewhat outdated technology because most of the enterprises are more interested to automate large-scale business processes (consisting of both application and human tasks) rather than only workflows. Although many of the WfMSs are providing also EAI capabilities as of now, they are still quite limited in this area and cannot really compete with EAI-based solutions, e.g., in regard to scalability or speed of message passing. This is why WfMSs have become superseded by newer technologies such as BPMSs and ESBs that aim at combining the best sides of workflow management and EAI. [40][84]

The WfMS products in the today's market place are often quite different from each other [37]. The differences can be explained by the fact that the required functionality vary so much depending on the enterprise. Many WfMSs have been actually aimed only at niche markets. The differences can be also explained by that the most solutions build entirely upon vendor-specific (proprietary) technologies. According to Aalst [37], some of the major WfMS vendors of today include Filenet, DST, Plexus, Staffware, Tibco, COSA, IBM and BEA.

6.2 Business Process Management Systems (BPMS)

Business Process Management Systems (BPMS) represent contemporary systems for managing business processes in enterprises and for enterprise application integration (EAI). BPMSs have arisen in the 00s as the next evolution of workflow management, EAI, and ERP systems. The main goal of BPMSs is to allow companies to realize increased efficiency and cost savings through reduced manual processes. In one definition BPMS is defined as "an electronic production line of information that flow information from one stage to another in order to enable decisions and actions for the delivery of goods, services, or knowledge to internal or external customers." [20][85]

Although BPMSs still represent rather new and maturing technology, they have already drawn some great interest in them from many enterprises. Some people are actually saying that BPMSs will become even more prominent enterprise applications than ERP, CRM and HR, all of which have become the mainstays of modern organizations. [85]

Although BPMSs promise pretty much the same as ERP systems (Enterprise Resource Planning) in the 90s - to realize the benefits of BPM in a single system, they are still yet very different kind of systems by nature. While ERPs were basically massive software systems that promised everything in a single package, BPMSs are in contrast minimum systems that allow organizations to pick best-of-breed components themselves. Unlike ERPs, BPMSs are also more open systems as they are based on open XML and Web Service standards. [85]

Compared to WfMSs, BPMSs have a larger scope. Where WfMSs were focused on managing processes primarily at the workflow level, BPMSs are focused on large-scale business processes that are enterprise-wide. Also, while WfMSs usually support only two phases of the business process lifecycle (modeling, automation), BPMSs support all of the four phases (also monitoring and analysis phases). Another difference is that BPMSs are based on open standards while WfMSs rely on proprietary technologies. BPMSs also add robust application integration (EAI), application development, and richer process simulation and modeling capabilities that are usually lacking from WfMSs. [85]

In the core of a BPMS is a SOA. The combination of a SOA and a BPMS represents currently the clear favorite technology how to address BPM in today's enterprises. In this combination, the SOA's role is to provide services and the BPMS's role to allow to build automated business processes from these services. The other core technologies of BPMSs include application server technologies such as J2EE and .NET and WS composition languages such as BPEL.

According to [86], today's major BPMS vendors include Tibco, Oracle, IBM, Microsoft, BEA Systems, Computer Associates, Vitria, Intalio, Fuego, Ultimus, Lombardi, Savvion, Bluespring and Pegasystems.

6.2.1 Components

At the heart of a BPMS is the process execution engine (usually a BPEL engine) that is responsible for executing the business process definitions. It is fundamentally a "black box" that controls the execution of business processes or ensures the processes continue to move towards completion. A process execution engine may be compared to a database management system (DBMS) because they both operate on schema definitions, the DBMSs operate on data schemas and BPMSs operate on process schemas. [20][85]

Business processes are designed in BPMSs using a graphical design tool called often as a process editor. The editor facilitates the developer's work by providing a graphical environment for designing business processes. The editor programs of the BPMSs have been usually built upon common programming IDEs such as Eclipse. The completed process designs may be usually deployed to the engine with only a single click. [20][85]

The deployed processes may be controlled in a BPMS via a Web-based management console which usually gives a dashboard-style view to the processes. The management can use the console to monitor the status of processes, analyze their performance, and possibly to make changes in real-time, or in other words, for Business Activity Monitoring (BAM). The consoles may also provide facilities for deploying and debugging of processes.

6.2.2 Advantages

Below is listed some of the advantages that a BPMS can bring for an enterprise. The list is built according to [9], [31], and [85].

- Increased efficiency, quality, reliability, and security of business processes and reduced costs.
- The ability to transform the distinct disciplines of workflow, EAI, and B2B from a complex high-end solution practiced by a few highly skilled people into a solution accessible to the masses of developers producing new types of agile, loosely coupled applications.
- The use of explicit process models provides a means for communication between the stakeholders in the company such as managers, business analysts and IT people.
- The fact that BPMSs are driven by replaceable process models rather than code allows for altering processes without recoding parts of enterprise systems.
- The vertical transparency in the company increases as the management is allowed to monitor and control the execution of business processes through the management console of the BPMS.
- Graphical designing of business processes with the bundled easy-to-use design tools.
- Provides easy integration of legacy- and custom-built applications through a set of built-in application adapters.
- The explicit representation of processes support redesign efforts by the management.

6.3 Enterprise Service Bus (ESB)

Enterprise Service Bus (ESB) is similar platform to a BPMS - an EAI/BPM platform to connect enterprise services and applications. The ESB represents one of the latest buzzwords of EAI and BPM. ESBs have emerged basically as a follow-up for Service-Oriented Architectures (SOAs) and Web Services. An ESB enhances a SOA by providing a universal integration backbone for it. ESBs aim at basically combining the application integration best practices of the last years. [87]

The differences between ESBs and BPMSs are rather small. They are both basically platforms that build upon a SOA and that enable to construct business processes through service orchestration. They differ probably most in that the ESBs are based upon a bus architecture and they provide somewhat more sophisticated EAI capabilities than BPMSs. [88]

The architectural difference between an ESB and a BPMS (and other common EAI platforms) is shown in Figure 25. Traditionally, EAI platforms have been dominated by the so called hub-and-spoke (or broker) architectures. However, the hub has recently become undesirable because it represents a single point of failure/congestion and because they are today very complex pieces of (proprietary) software that are difficult to manage. A bus instead is a highly distributed architecture that scales better. It is notable that despite its distributed nature, the ESB can still be centrally managed like traditional EAI systems. [12]

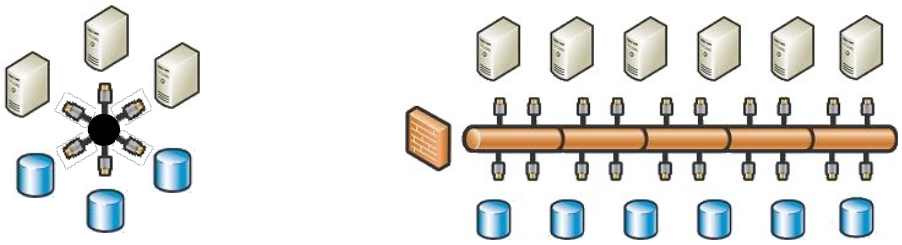


Figure 25: Hub-and-spoke and bus architecture

ESBs are very loosely coupled platforms by nature as the service consumers have been completely decoupled from the service providers. A service consumer in an ESB is enabled to invoke a service provider regardless of its physical location, protocol, or API. The consumer can connect to any of the bus endpoints in order to consume a service of a service provider. The consumer is often also enabled to choose the used protocol as many ESBs support connecting to the bus using several protocols, e.g., using SOAP, HTTP or JMS. In ESBs, the other differences in the connected applications are overcome through "mediation". The mediation capabilities enable to overcome differences in application message formats, communication semantics, and security models. [87]

It is notable that the bus is comprised, in addition to application services, also of so called infrastructure services. While the application services represent the business applications the bus offers for its clients, the infrastructure services are instead services the bus offers for itself. Infrastructure services are in the core of ESB and provide the bus with some critical capabilities such as content-based routing, orchestration, security, mediation, and logging. Infrastructure services are applied to bus messages en route between bus endpoints. Just like application services, infrastructure services may be physically located anywhere in the bus and they may be obtained from several vendors. [87]

6.3.1 Architecture and Routing

Figure 26 depicts the bus architecture more precisely. The highly distributed nature of the ESB is largely due to traits of the service container. Service containers serve essentially as the endpoints of the bus. Yet, each service container provides fundamentally the same base functionality as an EAI broker - connectivity, application adapters, routing of messages based on rules, and data transformation engine – but yet in a highly distributed fashion. Technically, the service containers are remote processes that host software components. They are managed by the surrounding invocation and management framework. Service containers are simple and lightweight and may be separately deployed. [89][90]

Routing is especially interesting in the ESB since it differs significantly from the traditional hub-and-spoke routing used in EAI systems. In ESBs, the default routing style is called as itinerary-based routing. In itinerary-based routing each message that is moving in the bus is specified with a list of forwarding addresses, or itineraries. An itinerary is basically a set of instructions in the message header telling the ESB invocation framework to which bus endpoints, or service containers, a specific message needs to be delivered as it travels across the bus. Itineraries contribute to the distributed nature of the ESB architecture by eliminating the dependency on a centralized routing engine, which could potentially be a single point of failure. An example routing itinerary is shown in figure 26. [89][90]

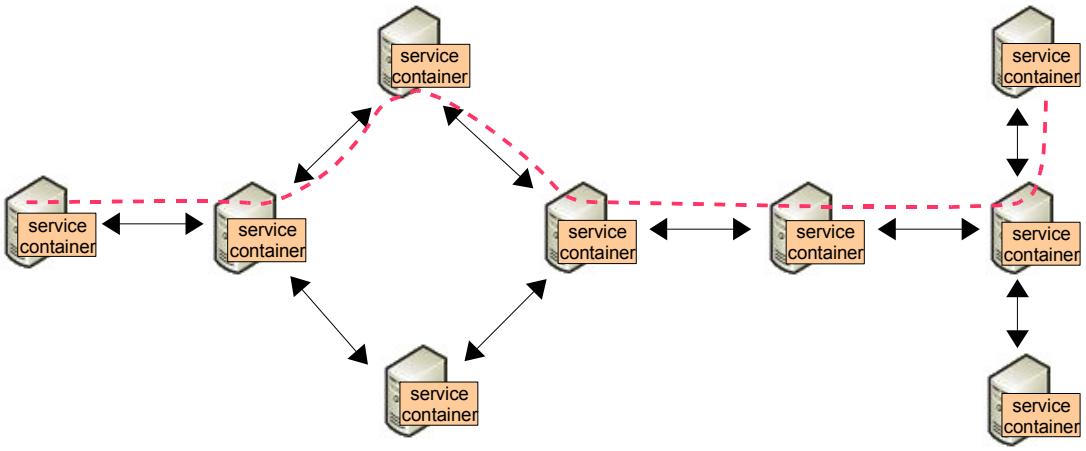


Figure 26: Architecture of ESB and itinerary-based routing

6.3.2 Features

Below we have listed some interesting features of the contemporary ESB products (according to [29], [87], [88], [89]). It is notable there is no common consensus of what exact features ESBs should provide.

Orchestration – Service orchestration is a critical capability of an ESB. Orchestration capabilities allow the company management to build automated business processes. Orchestration capabilities of ESBs are usually BPEL-based.

Distributed operation and management – As mentioned earlier the centralized hub-and-spoke architectures of traditional EAI solutions pose risks that can be avoided with the bus architecture. Because ESBs are based on distributed networks, there is no single point of failure or congestion. The distributed operation enables also to optimize network traffic, to provide alternate routes in case of failure, and to distribute processing loads. However, although the operation of an ESB is distributed, the system can be still managed centrally. For the management, the ESBs usually provide a Web-based management console.

Document-oriented processing model – ESBs are based on document-centric processing model rather than on traditional database-centric model. Each event, or step of a business transaction, is always triggered through a complete (business) document, which contains the necessary data items.

Multiple communication patterns – Many of the ESBs support several communication patterns including request-response, publish-subscribe, and one-way event-driven patterns.

Adapter mechanisms – In addition to standard SOAP and WSDL, many ESBs support also other interfaces and protocols such as SMTP, FTP, JMS, RMI, and JDBC through adapter mechanisms. The adapters enable to plug also the non-services (e.g. legacy applications) into the bus with only small efforts. Adapters also helps in bridging between already installed middleware platforms and between asynchronous (messaging) and synchronous (RPC) domains. Adapters may be developed within the enterprise, obtained from the bus provider, or from third party developers.

Mediation – The differences in the applications are overcome through "mediation" in ESBs. Mediation helps in bridging differences in message formats, security models, protocols, and communication semantics. It is basically much more simple to configure an ESB for mediation than what it is to make changes to the actual applications. The most often used form of mediation is message transformation that is used to convert application message formats in order that applications are able to understand messages being sent by the other applications. The data transformation rules can be usually written in a range of scripting or declarative languages, e.g., in XSLT. Many ESB transformation engines also support translating legacy formats (e.g., EDI, structured files, and relational data) into XML and vice versa.

Intelligent routing – The default routing mechanism of the ESB is known as itinerary-based routing (was discussed in detail earlier). In addition, many ESBs support also defining more intelligent type of routing, such as content-based, payload type (schema) based, workload sharing enabled, or failover routing. The content-based routing in ESBs is usually based on a combination of XPath and Javascript that can be used to identify key sections from the XML messages being transferred and to apply regular expressions in them. For example, XPath expression "/order/item[price>1000]" could be set to select all order items with a price greater than 1000 to be routed with a high priority. The workload sharing enabled routing refers to sharing heavy workloads evenly between the multiple instances of a service in the distributed bus. The failover routing is used to pass exception information between applications. Some of the most advanced ESBs allow also to change routing rules nimbly on-the-fly to rapidly respond to change.

Quality of Service – The critical quality of service features, including reliable delivery, transaction management, scalable performance, and security should be self-evidently supported by any real-world ESB solutions.

6.3.3 Advantages

Below is listed some of the key advantages of ESBs over some other EAI/BPM solutions (according to [87], [89]).

Based on open standards - The key advantage of ESB compared to former integration solutions is that its entirely based on open XML, Web Service, and WS-* standards. This has a major impact in the complexity, and especially in the cost of integration. Reduces also the risk of vendor lock-ins.

Flexible – The ESB's event-driven architecture is both responsive and adaptive. Change management becomes simpler and more powerful. Changes can be applied dynamically without disrupting systems in operation.

Modular and product-agnostic - ESBs promise to be extremely modular in the sense its parts can be replaced with new implementations and new functionality can be obtained from different vendors to address new integration requirements.

Developer- and management-friendly - Until recently there were basically only two routes to integration: "roll your own" or buy a packaged EAI product. Typically, developers would choose the first option for maximum flexibility, while the management preferred the second, for consistency and security. ESB has emerged as a "middle way" between these two approaches, providing a developer-friendly integration platform, open to all the latest Java, .NET, and other standards and components, without sacrificing the benefits of a packaged approach to integration.

Leverages existing applications - Because all applications are fundamentally services in ESBs, enterprises are enabled to assemble systems from best-of-breed pieces. No applications need to be replaced when an ESB is introduced.

Supports autonomous service development – In an ESB environment, different departments and business units of a company can host their own services and are allowed to autonomously develop and deploy new services. The business units are also allowed to separately scale, replicate, and replace their service.

Easy installation – The installation of an ESB product should be relatively easy in most cases. Furthermore, unlike traditional EAI platforms that often required a "big-bang installations", the today's ESBs may be introduced gradually, step by step.

6.4 Business Activity Monitoring Tools (BAM)

The term Business Activity Monitoring (BAM) was originally coined by analysts Gartner, Inc [35] and refers to the aggregation, analysis, and presentation of real time information about activities inside organizations and involving customers and partners. The term Business Activity may be considered as an automated business that is being executed through a BPM system. BAM is a solution primarily intended to provide a real-time summary of business activities to operations managers and upper management. Currently, BAM tools represent an emerging area of interest in BPM. [2]

The BAM benefits an enterprise in that it enables to make better informed business decisions and to more quickly address problem areas. The information provided by BAM enables to see the current trends in business, to take advantage of emerging opportunities, and to predict likely future events. BAM data is also useful also for managers whose job is to share workloads between employees, ensure the required process resources are available, handle exceptions that occur during execution, etc. [38]

In practice, business activity monitoring is about measuring a variety of different business process metrics such as volumes, durations, and costs. BAM tools may also support the measuring of individuals, applications, or even organizational entities such as departments. Many BAM tools provide also a knowledge base of historical data for answering questions of a type "what-when-who" at a later time. The business information gathered by BAM may be usually viewed through a dashboard-style user interface. The dashboard enables managers to view the process metrics in a visual view, e.g., in the form of diagrams. In Figure 27 is shown an example of IBM Websphere BAM Monitor's user interface. [38]

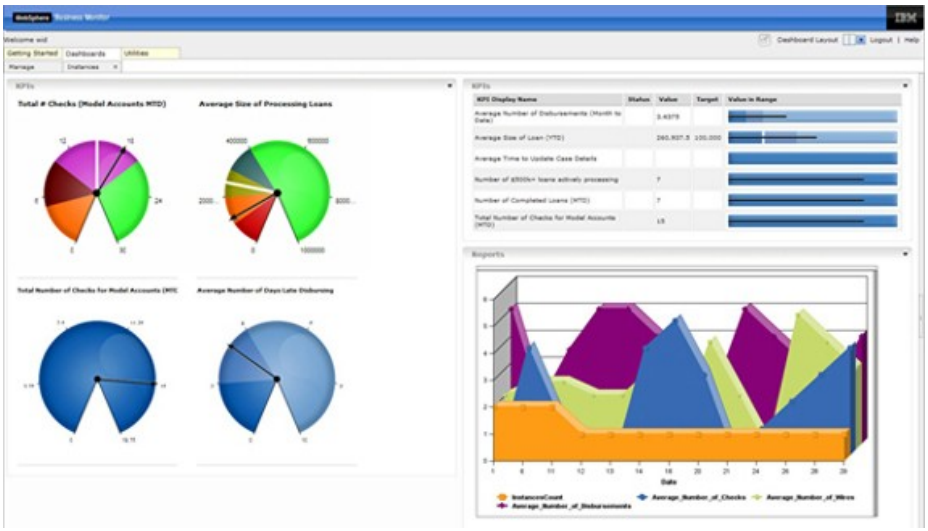


Figure 27: IBM Websphere Business Monitor

An essential part of any BAM tool is the reporting framework. The reporting framework provides business activity reports to business process owners. Two kind of reports are possible: metrics reports and proactive notifications. The metrics reports are generated on-demand by a user who is willing to inspect the process details. Proactive notifications are instead generated on a periodic basis or when certain performance thresholds are exceeded. Proactive notifications are used to notify users about conditions or certain events that may require manual intervention. For example, a simple e-mail alert may be sent to a process administrator when a certain threshold has been exceeded during the execution of process. [38]

6.5 BPEL Engines

A BPEL is fundamentally a server-side application that is capable of executing process definitions defined in BPEL language. A BPEL engine is often called also as a BPEL server, a process manager, or an orchestration server.

The main role of a BPEL engine is to execute BPEL processes that have been deployed to it. In addition to execution of processes, BPEL engines have also many other roles including exposing the deployed processes as Web Services, instantiating and terminating process instances, monitoring the processes being executed, catching exceptions, running exception handlers, executing transactions, and supporting process persistence. In addition, a BPEL engine is also required to interpret WSDL, XPath and XML Schema definitions that are associated with process definitions. And of course, it is also required to provide high performance and scalability. [64][74][91]

BPEL engines reside at the core of BPM systems. They are thus in most cases application server applications like the BPM systems. Basically, none of the BPEL engines have been implemented as stand-alone applications. The engines are usually coupled with the other applications that come with BPM systems (including BPEL editor, application server, database, etc.). The engines cannot usually be acquired separately from BPM systems. [64] [91]

BPEL process definitions may be usually deployed directly to the BPEL engines from the editors they have been developed in. The definitions may be yet deployed usually also through the management console of an engine or directly from the command line. Before a process definition can be deployed, it must be first packaged into a so called process archive into which is placed the other necessary deployment artifacts including WSDLs, XSDs, XSLTs, and others. Into a process archive must be included also a so called deployment descriptor. A deployment descriptor is a simple XML file that acts as a catalog of the deployment artifacts and contains deployment-time instructions for the engine. Deployment descriptors are not covered by the BPEL standard but are specific to BPEL engines. The process archives are most often some sort of Java packages such as JARs, WARs, or similar.

Administration and monitoring of running processes is performed through the Web-based management console of the engine. The console is used to administer, monitor, deploy, undeploy, and debug processes. The console can provide some really useful real-time metrics for the management for analyzing the performance of business processes being executed. In Figure 28 is shown a screenshot of the management console of the Oracle Business Process Manager BPEL engine.

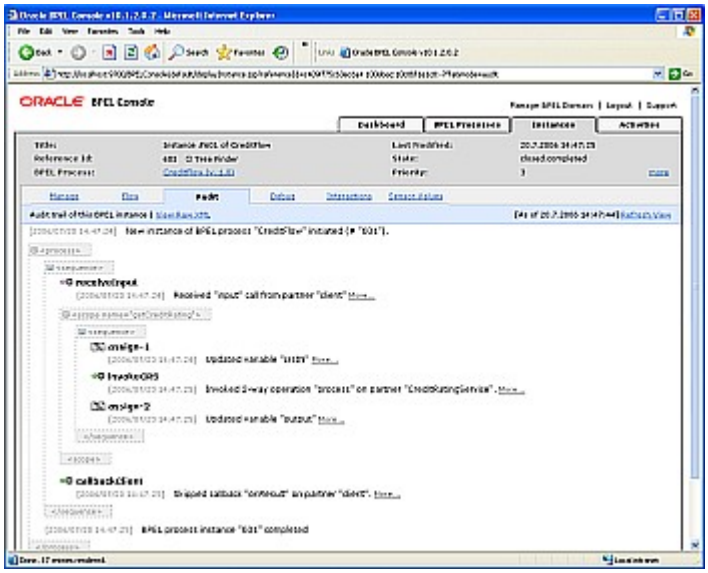


Figure 28: Oracle Business Process Manager administration console

Process persistence is an often heard term in the context of BPEL engines. If a BPEL engine is running in persistence mode, it means that the process state information is being stored in a database. This prevents processes from failing in the case of a server failure (e.g. caused by a power cut). The other very similar concept is "process hydration" (and dehydration) that refers to suspending inactive processes to a database from consuming the working memory of a BPEL engine.

In Table 5 is shown a few BPEL engines that we have evaluated for the purpose of this thesis. Most of the today's BPEL engines are some very expensive commercial systems provided by some major system vendors including Microsoft, IBM, SAP, BEA and Oracle. There are yet some engines that may be acquired also for free. The free alternatives include such as ActiveBPEL, Apache Ode, ObjectWeb orchestra, OpenESB, JBoss jBPM, and Virtuoso. Some of them are Open Source and some are freeware. Some of the tested BPEL engines are no longer actively developed. They include Twister, Bexee, MidOffice, Apache Agila, and PXE. For a complete list of the current BPEL engines, the reader is advised to view [92].

BPEL Engines (May 07)	OpenESB+ BPEL engine component	ActiveBPEL Engine	Apache Ode	Oracle BPEL Process Manager	IBM Websphere Business Integration Server Foundation	ActiveBPEL Enterprise	Microsoft Biztalk 2006	IBM BPWS4J Engine
Provider	Java community	Active Endpoints	Apache	Oracle	IBM	Active Endpoints	Microsoft	IBM
Platform	J2EE	J2EE	J2EE	J2EE	J2EE	J2EE	.NET	J2EE
Current status	Under development	Under development	Under developme nt	Under development	Under development	Under development	Under development	Discontin ued
Latest version	2.0 (2007)	2.0	1.1	10.1 (2006)	4.4 (4/2005)	2.0	2006	2.1 (2004)
License	Open Source	Open Source	Open Source	Commercial	Commercial	Commercial	Commercial	Commerc ial
Supported BPEL version	1.1	2.0	2.0 (partially)	2.0	2.0	2.0	?	1.1
Debugging	No	No	Yes	Yes	No	No	?	No

Table 5: BPEL engines

6.6 Open Source BPM Platform: Netbeans 5.5

Netbeans is an Open Source programming IDE developed originally by Sun Microsystems. It is an IDE intended in the first place for Java programming. It is a very similar IDE to another common IDE – Eclipse. In addition to Java (SE/EE) programming, Netbeans can be used today also for many other purposes. This is because the IDE can be extended with a variety of extensions. There have been developed extensions, e.g., for SOA/BPEL programming, business process modeling, Web Service development, mobile programming, Web programming, C/C++ programming, etc. [93][94]

Netbeans becomes a complete BPM platform when one installs the Enterprise Pack (extensions known as packs in Netbeans) for it. The Enterprise Pack is an extension geared toward enterprise application development. The pack provides the integrated tooling and runtime components (servers) needed to build, deploy, and manage business applications and processes. The tooling includes, in addition to a complete J2EE development environment, a BPEL editor and debugger, WSDL and XML Schema editors, and a project system for creating Web Services and composite Web Services. Netbeans with the Enterprise Pack provides basically all the required tooling for BPM within a single IDE.

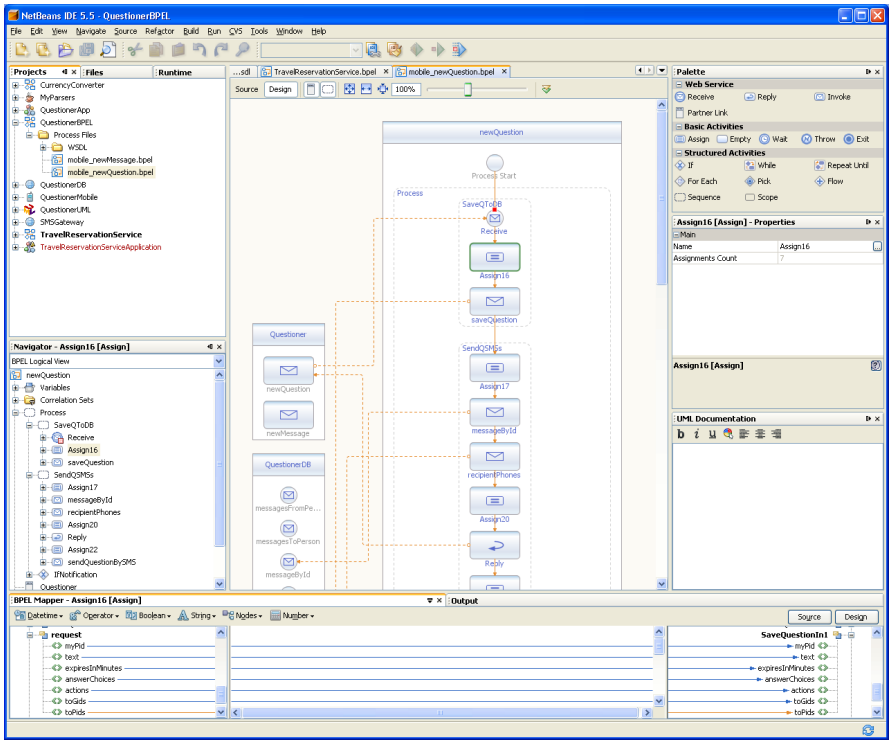


Figure 29: Developing BPEL processes with Netbeans

The development of Web Services is especially easy with Netbeans. Namely, with the aid of JAX-WS Web Service annotations of J2EE5, it is possible to wrap any Java application into a Web Service with minimal efforts. With JAX-WS one can annotate the methods of a Java application's main class (one writes @Webmethod on top of a Java class) and when the application is deployed, the application server exposes the application automatically as a Web Service and generates a WSDL for it.

To support the development of Web Services, Netbeans (enhanced with the Enterprise Pack) also provides a WSDL editor and a WSDL-to-Java wizard for developing Java applications from the already existing WSDLs. One can also define Web Services with security features such as authentication and authorization. SAML and the Liberty token profile are among the standards that are supported. Testing of Web services is also well-supported by Netbeans.

To support the development of business processes, the Enterprise Pack includes a BPEL editor and a BPEL execution engine. The development of BPEL processes is very straightforward with the bundled BPEL editor. The editor enables to orchestrate Web Services in a visual view and use a drag-and-drop technique. The editor shows the process diagrams in BPMN notation. The editor provides also a visual BPEL debugger that enables to step through processes and track the data being passed from one activity to another. Furthermore, in Netbeans it is also possible to nest and group BPEL processes into larger logical units by combining several IDE projects.

The BPEL execution environment builds on a Sun Java System Application Server that is bundled with the Enterprise Pack. More precisely, the application server includes an Open ESB module upon upon which the BPEL engine has been implemented. Both the application server and the BPEL engine have been seamlessly integrated with the IDE. It is possible to also integrate other application servers to the IDE including at least Glassfish, Weblogic, and JBoss.

7 Advanced Topics

This chapter discusses some practical issues of BPM and about some new ideas that we have found from miscellaneous papers and articles we have read for the purpose of this thesis.

7.1 BPMN-to-BPEL Mapping

A common problem in BPM is how to transfer the coarse-grained process designs of BPMN diagrams into executable BPEL processes. As today the business processes are most often modeled with BPMN and executed with BPEL, performing such mappings is often required.

In the case of simple BPMN diagrams, the mapping is relatively easy to do. This can be explained by the fact that the basic constructs of the languages are somewhat similar. However, in the case of any more complex diagrams, the mapping is not always a trivial exercise anymore because there are some structural disparities in the mathematical foundations of the languages. This is why it is often required to restructure the flows of BPMN diagrams to be BPEL isomorphic before they can be actually mapped. It should be noted also that BPEL language has nor ever been designed with the process design interchange in mind. [95]

Round-trip engineering or generating a BPMN diagram from a BPEL process definition, is even more difficult. It is often even impossible because of the previously mentioned structural disparities. One can also consider that the mapping from BPEL to BPMN is always more difficult because BPEL represents fundamentally a subset of the BPMN language in what comes to expressiveness of the language. The round-trip engineering problem appears in practice in that it is difficult to keep the original BPMN diagram and the generated BPEL code in sync, in the sense any modification to either of the models always becomes propagated to the another. [95]

It is notable that the BPMN-to-BPEL mapping is possible to do using a dedicated tool for this purpose. This kind of tools have been sometimes bundled with BPMN editors. The tool enables to map a BPMN diagram to a BPEL process similarly as one can map a UML diagram to Java code in some UML/Java development environments. The generated code represents a skeletal BPEL process that consists of invoke/receive activities, partner links, and scopes. Further coding is required to specify the application bindings, messaging patterns, data transformations, an exception handlers.

7.2 B2B Automation: Introducing Choreographies

To take the process automation one step further from automating internal business processes to automating also B2B processes, it is required to utilize choreographies because orchestration is not sufficient for this purpose anymore. By B2B processes is meant essentially the processes of strategic enterprise networks (e.g., a supply chain process).

As we remember from Chapter 2, the choreographies represent an alternative way for WS composition. Choreographies describe the global, publicly observable view of a multi-participant process exchange without the need for a central coordinator. Choreographies are based on stateful services and they may be specified with choreography languages such as WS-CDL.

From the business perspective, choreographies represent contracts between companies of how to conduct business using Web Services. Each contract specifies unambiguously the rules of engagement between the services of the companies.

Orchestration is not sufficient for B2B situations basically because it is required to use a centralized coordinator (orchestration server). The coordinator does not really fit in B2B because it is difficult to find a place for the centralized server. Namely, none of the partners who are involved are probably willing to give away the control of the process to some other partner. A choreography is in this sense a solution that is more fair to the involved parties. [96]

One might think that why do not we specify a choreography just as a diagram, e.g., as an UML or a BPMN diagram. The reason is quite simple - the graphical notations are not enough precise. In order to unambiguously specify the rules of a choreography, the only way is to use a dedicated choreography language. [97]

Developing a B2B choreography is a collaborative effort by all of the involved partners. The partners have to jointly agree upon the design of the choreography process. When there is an agreement on the (top-down) design, the next step is that each of the partners implement the required services. The behavior of the implemented services must match the choreography. It is notable that a choreography does not specify how each of the parties should implement the services, it only specifies what services each party is required to provide. [96][97]

In order to provide outputs for the required services, the partners may have to implement new private business processes. The new private business processes may be implemented using orchestration. Namely, each partner of a choreography is enabled to utilize the choreography definition to generate skeletal private orchestrations for themselves (see next section for more). [97]

7.3 Combining Choreographies and Orchestration

It is interesting to see that the two WS composition paradigms, choreography and orchestration, can be actually combined. According to Havey [97], the paradigms should not be considered as exclusive but rather complementary to each other.

As mentioned in the previous section, choreographies are used to specify public (B2B) processes between companies, and orchestration to specify private processes within companies. Also, to support the choreographies, the involved companies may be required to develop new services and private business processes.

What is interesting here is that to develop the services and business processes required by a choreography, the involved companies can actually utilize the choreography specification to generate them. This is enabled by the fact that it is always possible to extract each involved participant's individual behavior and interface from any choreography specification. Such an approach can reduce significantly the time needed in each company to develop the services required by a choreography. [96][97]

Combining a choreography with orchestration refers in practice to mapping a WS-CDL choreography to BPEL. This is because these languages represent currently the de facto choreography and orchestration languages, respectively. To perform the mapping between the languages, there have been developed some dedicated tools for this purpose. A WS-CDL-to-BPEL mapping tool can take a WS-CDL as input and output a BPEL process (and its WSDL). The generated BPEL process is a skeletal definition consisting of only the interaction points where the process is required to interact with the outer world. The actual process logic, or how the process handles the choreography requests internally, must be further specified between these interaction points. [96][97]

Figure 30 depicts a company network in which four companies collaborate in a choreography process. To support the choreography, each company has deployed an orchestration server to provide the expected services for the choreography. The services that are provided for the choreography are composite services comprised of a process.

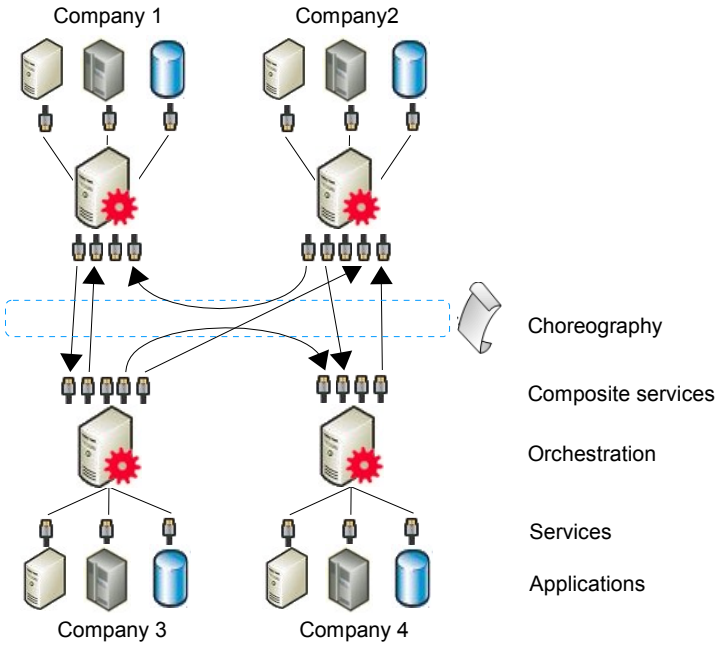


Figure 30: Collaborating in a choreography

According to Havey [97], mapping a WS-CDL choreography to BPEL is not the only option, namely, one could also map to BPMN. The motivation is that BPMN is in most cases the main design language of business processes in companies. It should be mentioned that both the WS-CDL-to-BPEL and the WS-CDL-to-BPMN mappings are still quite difficult in practice because of many structural disparities between the languages.

8 Experimental Part: Questioner Application

This chapter serves as the experimental part of this thesis. For the purpose of the experimental part, we have developed a client-server application called Questioner.

8.1 Overview

Questioner is a client-server application consisting of a J2EE/BPEL server and a J2ME mobile client program. The Questioner application is fundamentally a specialized SMS gateway through which it is possible to send question-formatted special type of SMSs to mobile phones. It has been originally intended essentially as a reusable component for automated business processes to enable simple human interactions with them.

"Questions" are similar to ordinary SMSs but they differ in that they have predefined answer choices. This means a question recipient can reply the question only with those answer choices that have been defined by the sender at the the moment of sending. Some other differences to SMSs include that questions may be specified with an expiry time, they may be sent to groups of people, and that the replies are collected to the server rather sent directly to the sender.

In Figure 31 is shown the server and client components of the Questioner application. The server is fundamentally a message broker which role is to deliver questions to the mobiles of the users. The second main role of it is to act as a centralized storage of all data. The server has a Web Service interface through which it is possible to send questions and access various operations of the system. The mobile clients connect to the server also through the Web Service interface. The questions are sent from the server to the clients over SMS connection. The server utilizes a normal GSM phone to send the SMSs in practice (see picture).

The mobile client program's role is to listen for incoming questions sent from the server. The mobile program allows also to view the received questions and to answer them. It also allows to send new questions. The client program is a basically simple Java application that can be installed to almost any of today's phones. The program communicates with the server over HTTP/SOAP.

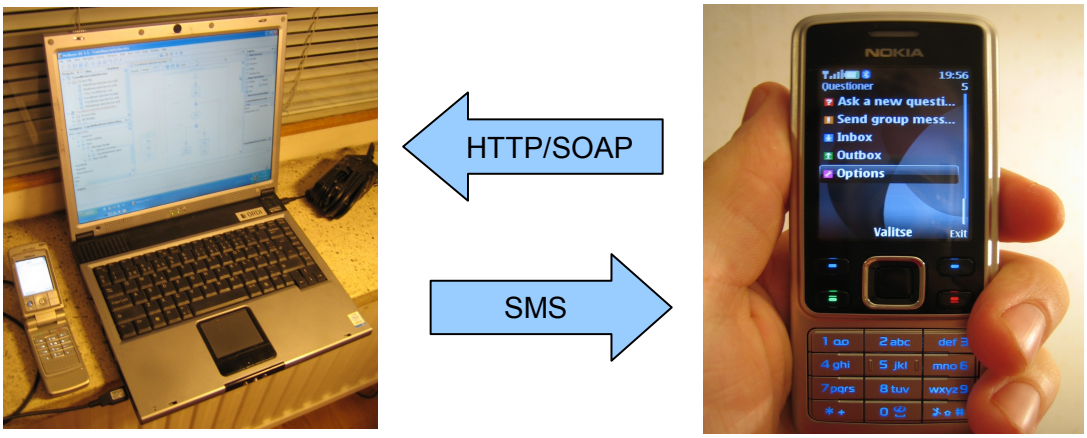


Figure 31: Questioner components

8.2 Original Intended Use

Although Questioner can be used now for actually quite many purposes, it was originally intended as a reusable human interaction component for automated business processes. The idea was to enable executable business processes to query business process owners for (processing) instructions, decisions, and other run-time guidance. By business process owners is meant the executives, managers, process administrators, or basically anybody who are responsible for a business process.

With the help of Questioner, the business process owners can control their business processes independently of time and place. This can lead to an improved efficiency as the process is enabled to receive important processing instructions and decisions more quickly.

In Figure 32 we can see how a business process may be controlled in practice with Questioner. The figure represents a simple order-delivery process of a manufacturing company. The goal of the process is to manufacture the ordered goods and then deliver them to the customer. In the figure is shown only a few steps of the entire process. The Questioner is used for two purposes in the process: to query a manager for continue instructions when enough goods have been manufactured, and to send maintenance requests if production equipment requires service.

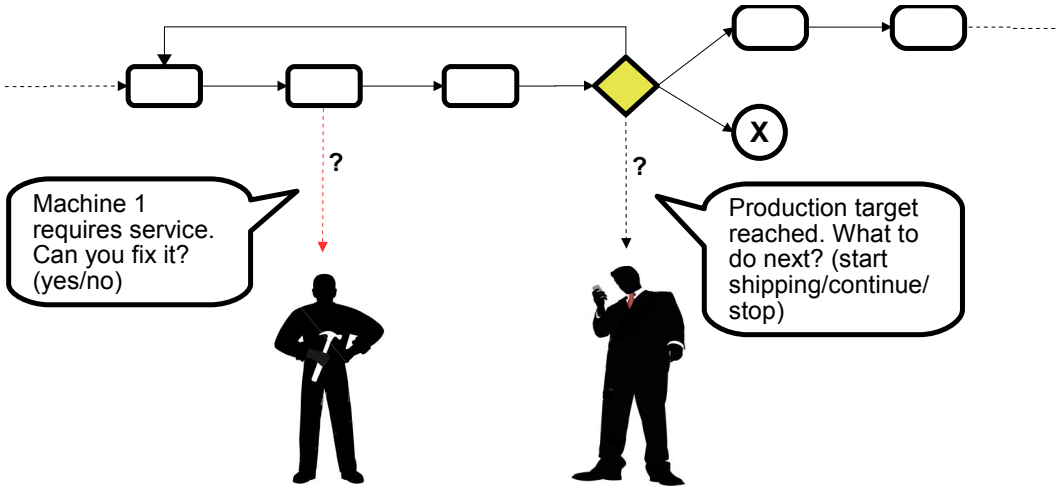


Figure 32: Questioner with an automated business process

8.3 Other Uses

Because Questioner is fundamentally a very generic application, it is actually quite easy to think many other use scenarios for it. In the following we present a few examples.

8.3.1 Organize Events with Friends

One use case of Questioner would be to use it in organizing events within communities. By communities is meant social networks such as friend groups, sports club members, work place teams, etc. The idea is that community members are allowed to easily invite people to events they are going to organize. The Questioner enables a community member to send an event invitation to the other members as a simple question in which he/she asks whether the other members want to participate the event or not. Examples of typical community events include parties, meetings, and sports events such as football matches that require a number of people.

Figure 33 represents an example scenario in which a person (on the left) tries to organize a football match with his football friends. In order to be able to organize the match, the person needs at least 10 players to join him. To get so many players, the organizer is required to send the question to at least 20 of his friends.

Because the person wants to play the match already today, he specifies to the question that his friends have exactly two hours time answer to the question. After two hours of sending the question, when the invitation has expired, the match organizer receives a summary message (from the server) of all the answers from his friends. He can then decide whether there are enough players for a match or not and then send either a confirmation message or a cancellation message to the people who have answered "yes". If he decides to send a confirmation message then he can send also more details (e.g. about the match location) in the same message.

In this kind of scenario, the main benefit of using Questioner instead of SMS is that the invitation message can be sent simultaneously to a multitude of people and that the sender will eventually receive only one reply message (a summary of the answers) instead of many. It is quite much easier for the organizer to receive only a one summary message as a reply rather than 20 SMSs. The other main benefit over SMS is that the invitation can be specified with an expiry time.

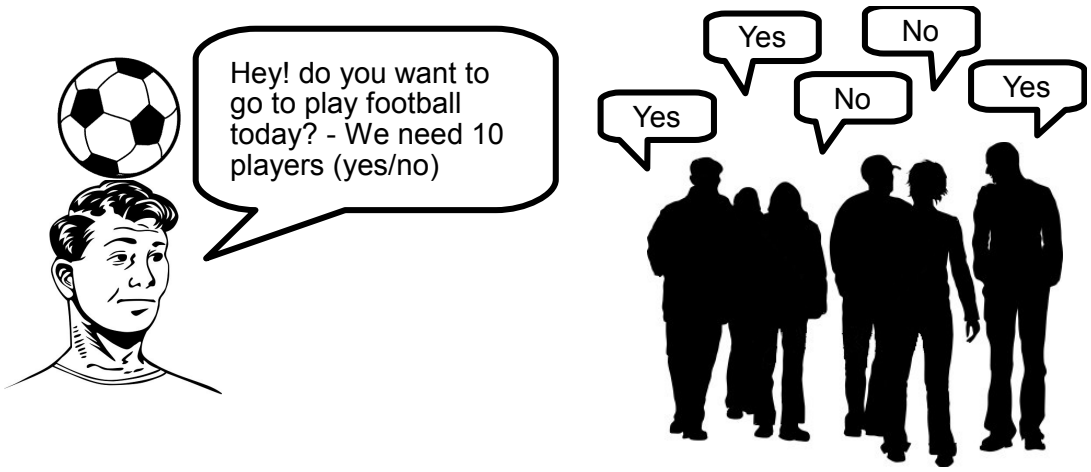


Figure 33: Organizing events with Questioner

8.3.2 Quick Polls

Questioner can be also used to arrange many kind of opinion polls or votes within user communities. In Figure 34 is shown an example scenario in which a person asks his friends which restaurant they should go for a dinner later on the same day. This scenario represents only one of the numerous similar scenarios in which the Questioner may be used to ask an opinion of people about something.

Because Questioner supports sending questions also through the Web Service API, it would be completely possible to organize polls from Internet and, more importantly, from other applications. For example, some survey software could utilize Questioner relatively easily as an application component to execute mobile surveys.

Figure 34: Opinion polling with Questioner

8.4 Features

In the following we have listed the most important features of Questioner.

- **Human and Web Service interface –** The most important feature of Questioner is probably that it provides two interfaces: a human interface (mobile) and a system interface (Web Service API). The use of Questioner is not only limited to the mobile client program but one can also use the Web service interface to send and receive questions, view answers, view user information, join and unjoin user groups, etc. All of the functionalities of Questioner are accessible through both interfaces.

- **Questions may be sent both to individuals and groups** – The sender can select either an individual, a group of individuals, or a predefined community group as a recipient for his/her question.

- **Answer time of questions –** The sender can specify an expiry time for a question or by when the recipients should have sent their answers.

- **Result messages -** After the question has expired, the server (automatically) generates a summary message of the answers of the recipients and sends it to the sender. The result message feature may be only used if the question has been sent from a phone. The feature may be also disabled. In the summary message is listed how many times each answer choice was answered and by whom.

- **Group messages –** With Questioner it is possible to also send ordinary SMS-style messages, both to individuals and groups. This is useful, e.g., when sending confirmation messages of events or providing question recipients with further details about the sent question or the event invitation.

- **View question status –** The question sender is not necessarily required to wait for the full question expiry time in order to be able to view answers from people to his/her question. The status of the answers may be viewed at any time, both by the sender and recipients of the question. Furthermore, the status may be viewed either through the human (mobile) or the Web Service interface.

- **Create/join/unjoin groups –** The users can create new user groups in which the others users can further join. The user groups facilitates the sending of questions to large number of users.

- **Inboxes/outboxes –** Each user has his/her own inbox and outbox of questions in which is stored all the questions the user has sometimes received or sent.

- **Sending questions is free –** It is notable that the use of Questioner is actually free. This is because there are no SMS costs involved. The SMS channel is only used by the server (for invoking the client program). The HTTP channel is used instead whenever the client program initiates the connection.

8.5 Server-side Implementation

The server-side of the Questioner builds upon a J2EE application server. Upon the server is implemented three Web Services. The most important service is the Questioner Web Service through which all the questions are sent and which acts as the system API of Questioner. The Questioner Web Service is a composite service, a BPEL process, that orchestrates the two other services that are involved. The two other services include the Database Web Service and the SMS Gateway Web Service.

8.5.1 Components

Below is explained the three Web Service components that build the server-side of the Questioner application. In Figure 35 is represented further the architecture of the server-side.

Questioner Web Service

As mentioned, the main component of the server-side is the Questioner Web Service. Its role is to act as the system API of Questioner. All new questions are sent through this interface. It is also possible to query the interface for a bunch of information about questions, answers, users, groups, etc. It is notable that also the mobile client programs connect to this interface.

The Questioner Web Service is fundamentally a composite Web Service, or a BPEL process, that orchestrates the two other services that are involved. It orchestrates the Database Web Service to store and fetch data to and from the database, and the SMS Gateway Web Service in order to send questions to mobiles (over SMS). The enclosed BPEL process is executed by the BPEL engine of the application server.

The reason why BPEL technology has been used is that it enables to easily build long-running transactions. Sending a question in Questioner is fundamentally a long-running transaction. This can be explained by the fact that the process of sending a new question does not stop there when the question has been sent by the server but continues until the server has received the answers and then sent the result message to the sender. The process can take hours or even days to complete. If the Questioner Web Service would have been implemented only with Java, the implementation would have been considerably more complex.

In Appendix A is represented the process diagram, the WSDL, and the BPEL process definition of the Questioner Web Service.

SMS Gateway Web Service

This Web Service is a supportive resource for the Questioner Web Service. Its role is to allow to send questions to phones over SMS, or put another way, to act as an SMS gateway. The SMS Gateway is fundamentally a Java application that has been exposed as a Web Service. The gateway has been built using a Java SMS library called SMSlib [98]. It has been exposed as a Web Service through JAX-WS annotations of J2EE5.

To establish an SMS connection, the gateway utilizes a GSM phone that is connected to the server via a USB cable (see Figure 31). The built-in modem of the phone sends the SMSs in practice. The modem is controlled by means of AT commands. The AT commands are sent to the modem through a COM port. Any GSM phone can be basically utilized to send SMSs programmatically. The only requirement from the phone is that it supports the extended set of AT commands (almost all phones support).

The WSDL of the SMS Gateway can be found in Appendix B.

Database Web Service

This Web Service is also a supportive resource for the Questioner Web Service. The Database Web Service is only a Web Service interface for the database used with Questioner. As we mentioned earlier, all data in Questioner is centrally stored in a database. The Questioner Web Service utilizes the Database Web Service to store and fetch data to and from the database. To the database is stored all questions, group messages, answers to questions, users, user groups, etc.

The Database Web Service is in fact only a Java wrapper class that has been exposed as a Web Service. The wrapper class consists of a bunch of methods that perform the actual database operations. The methods include operations to save a question, fetch lists of questions, save an answer to question, etc. The actual SQL queries have been encoded inside the methods. JAX-WS annotations (of J2EE5) have been used to expose the wrapper class as a Web Service.

The WSDL of the Database Web Service and the database schema can be found in Appendix C.

8.5.2 Architecture

Figure 35 shows the architecture of Questioner. The figure shows also the "process" of sending a new question (from a mobile). In step 1, a new question is sent to the server from a mobile. In step 2, the server creates a new BPEL process instance and starts it. In step 3, the question is saved to the Questioner's database. In step 4, the question is sent to the recipients' phones through the SMS Gateway Web Service. In step 5, question recipients send their answers to the server. Answers are stored to the database through the Database Web Service. In step 6, while the question recipients are sending their answers, the BPEL process is in a loop mode polling the database every 1 minutes to check if everybody has already answered. In step 7, if all recipients have answered or if the answer time has expired, the BPEL process builds a summary message of all answers and sends it to the original sender as an SMS.

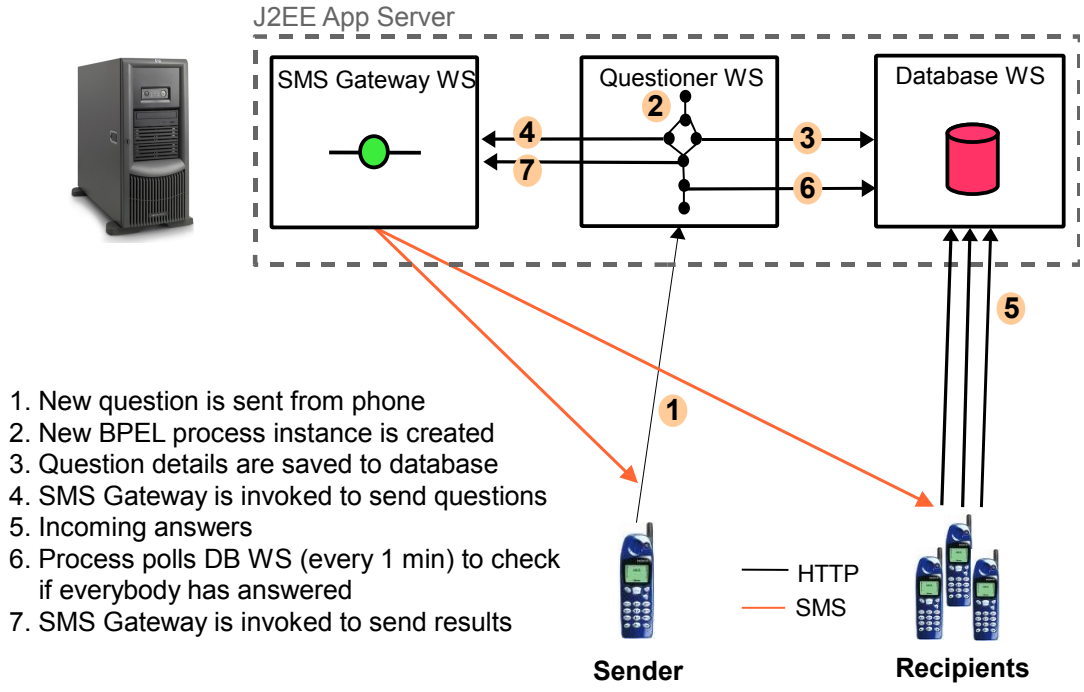


Figure 35: Questioner architecture

8.5.3 Development Tools

Netbeans IDE has played a centric role in the development of the Questioner application. Netbeans is an Open Source Java IDE similar to Eclipse. Although it is in the first place a Java IDE, it provides also many powerful tools for the development of Web Services and BPEL processes. All of the Web Services and BPEL processes of Questioner have been implemented using the integrated tooling of Netbeans. In addition to sophisticated tooling, Netbeans also provides the required run-time components. The Netbeans IDE was discussed earlier in Section 6.6. The figure 29 of Section 6.6 represents the Questioner BPEL process design-time view in Netbeans.

8.5.4 Used Technologies

Below is summarized what technologies and software have been used in the implementation of Questioner server-side.

- **Development environment –** Netbeans IDE 5.51
- **J2EE server** – Sun Java System Application Server 9.0
- **BPEL engine –** Netbeans Enterprise Pack OpenESB BPEL Engine
- **BPEL editor** – Netbeans Enterprise Pack BPEL Designer
- **SMS Gateway** – Implemented using SMSlib Java SMS library
- **Database –** Apache Derby 10.1.3.1 (was bundled with the application server)
- **Web Service wrappers** – Generated from Java classes using JAX-WS annotations (of J2EE5)

8.6 Client-side Implementation

The client-side of the Questioner has been implemented as a J2ME Midlet (Java 2 Mobile Edition). The main role of the program is to listen for incoming questions from the server. The client program also allows the user to view received questions, answer to questions, and send new questions. In addition, the user can also send group messages, view his/her question inbox and outbox, view other users' answers to questions, and to join/unjoin user groups.

In Figure 36 is shown two screenshots of the client program. On the left is shown the welcome screen of the program and on the right the "received question" screen which is shown to the user when he/she has received a new question, i.e., when the program has been automatically started by the phone. The same screen is also shown to the user when he/she has selected a question in the question inbox or outbox. More screenshots and a screen map of the client program can be found in Appendix D.

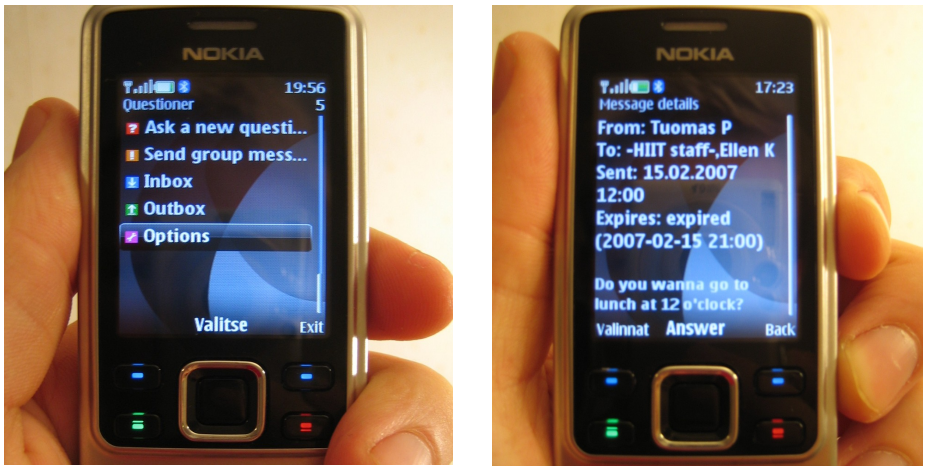


Figure 36: Questioner client program

Because the client program has been implemented as a Java program, it can be installed basically to any of today's mobile phones. The minimum requirements include the phone should support Java with WMA (Wireless Messaging API) and be enabled to connect to Internet. To be able to install the client program to as many phones as possible was actually one of the original design goals of the Questioner. This is why the client program was not implemented, e.g., in Symbian or Python.

What is interesting in the client program is that it uses two communication channels: SMS and HTTP. SMS channel is used to receive questions from the server and HTTP for the rest of the communications. The reason why the server uses the SMS channel is that this is the only way how the server can proactively invoke the client program ("sleeping client"). HTTP channel is instead used always when the client initiates the connection with the server, e.g., when it sends a question or queries the server for information. It is notable that when the client communicates over HTTP, the HTTP requests always carry SOAP messages. The SOAP messages are sent to the Questioner Web Service.

8.6.1 Features

XML/SOAP messaging – All messages that are being sent or received by the client program are in XML (or SOAP) format (for details, see the next section).

Push messages – The "Question SMSs" that are sent by the server to the client program are special-formatted SMSs. A Question SMS differs from an ordinary SMS in that its has a slightly different message header. An incoming Question SMS is recognized by its special header by the phone from the other incoming SMSs. The recognition is performed in practice by the J2ME push registry. A push registry is a special facility in Java phones that solves the "push problem" or allows to listen for incoming SMSs and proactively start applications in the phone.

Dumb client – The mobile program is a "dumb client" in the sense it stores no data locally but receives everything from the server over the air. The client program's role is to only view the data that has been sent to it (from the server) but not to store it. In Questioner, the server's database acts as a centralized data storage of the system.

Includes an XML parser – The client program bundles an XML parser which role is to parse data from the incoming XML and SOAP messages. The parser parses data from the messages so that it can be shown on the screen of the mobile. NanoXML parsing library has been used to implement the parser.

8.6.2 Messaging

As mentioned, all messages that are being sent or received by the client program are in XML format. The outgoing messages are in SOAP/XML format because they are sent to the Questioner Web Service. The incoming questions that come from the server over SMS are instead in plain XML format. The XML structures inside the SMSs have been kept very minimal because of the maximum length of an SMS (160 bytes).

In Figure 37 is represented the message format of an outgoing and incoming question. The message on the left represents the SOAP message that is sent by the client program when the user asks a new question. The message on the right represents the SMS that is further sent to the question recipients by the server.

```
<soapenv:Envelope>
  <soapenv:Body>
    <newQuestion>
      <myPid>1</myPid>
      <text>Do you like bananas?</text>
      <expiresInMinutes>20</expiresInMinutes>
      <answerChoices>yes,no</answerChoices>
      <actions></actions>
      <toGids></toGids>
      <toPids>2,3,4</toPids>
    </newQuestion>
  </soapenv:Body>
</soapenv:Envelope>
```

Outgoing question

```
<q>
  <id>32</id>
  <sdr>Thomas</sdr>
  <txt>Do you like bananas?</txt>
  <ts>23.09.2007 21:51</ts>
  <rcps>John,Paul,Andrea</rcps>
  <exp>0 h 20 min</exp>
  <ac>yes,no</ac>
</q>
```

Incoming question

Figure 37: Question message formats

8.6.3 Development Tools

The client program has been developed also using the Netbeans IDE. The IDE was enhanced with a plug-in called Mobility Pack to enable the development of mobile applications. The Mobility Pack offer some very advanced integrated tooling for the development of J2ME applications. For example, the pack allows to design the screens of a mobile application in a visual map style of view. Figure 38 represents the screen map of the client program in the mobile application screen design view of Netbeans.

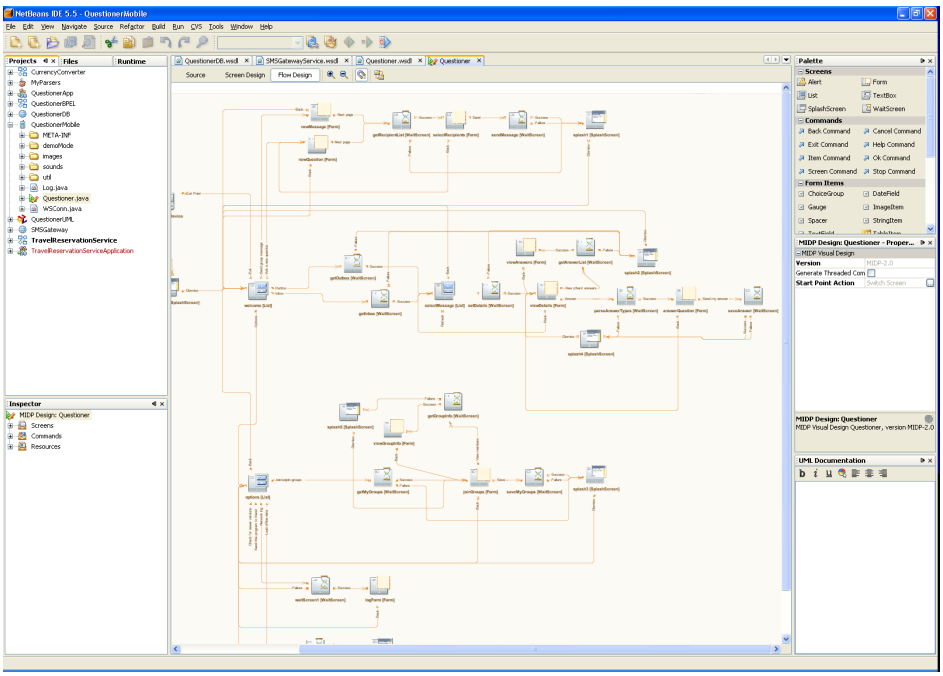


Figure 38: Developing the Questioner mobile client program

8.7 Discussion

The development of the Questioner application was rather a demanding task. It took almost five months to build the application. In the beginning, it took a long time before we actually realized what kind of application we want to develop. All we knew was that we want to develop some sort of application that utilizes BPEL technology and mobile phones.

In the beginning, we did not either have any proper project plan. Now, afterwards, we realize that it would have been a good idea to make a detailed plan of the application and its development. We should have planned better especially the client-server interaction patterns and the Web Service interfaces, i.e., the WSDLs.

The biggest problems we had during the development were related to the SMS Gateway and the application server. We just could not get them to work. Many of the problems were caused by software bugs we encountered in the application server and BPEL engine software, and in the SMSlib Java library. We also had some problems in testing the application components. Especially the testing of the mobile client program was difficult because the program can be properly tested only in the phone. It took a lot of time to install the program to the phone over and over again.

The further development plans for the Questioner are still open. The application could use some further development. Although the application works fine in demonstrations, it is difficult to say about how it would work in practice. More work is needed at least to enable the SMS Gateway to operate under heavy message loads. Furthermore, the client program should be tested more properly and with phones of many brands. We have only tested the client program in Nokia phones. A nice feature that could be developed would be to allow users to send the client program to their friend over Bluetooth.

The future plans for the Questioner include that we are going to publish all code to the Web as Open Source. We are currently developing a Web site for the Questioner. In addition, we are probably going to add a demo video of the application to Youtube in the near future. The reader is advised to search the Web for “Tuomas Piispanen” or “Questioner” to find the Web site of the Questioner and the video. At a later time, we might also try to figure out some commercial use for the application.

9 Conclusions

In this chapter we summarize the thesis and discuss the results and the outlook of BPM. We also provide some ideas of further research that could be performed.

9.1 Summary

This thesis has given an overview of the state-of-the-art technologies and methodologies of Business Process Management. The goal of this thesis has been to see a big picture of what BPM is today, and particularly from the IT perspective. We have examined several different technologies that may be used to apply BPM in practice. For example, we have presented technologies (and concepts) such as EAI, Web Services, Web Service composition, business process modeling, business process execution languages, and business process run-time systems.

In the experimental part of this thesis we built a client-server application called Questioner. The application is intended in the first place as a reusable Web Service component for automated business processes to enable human interactions. The idea is to enable business process owners to interact with business processes, at run-time, through a mobile phone. The application is also suitable for many other purposes.

9.2 Observations

The main objective of this thesis, to get the big picture of what is currently going on in the arena of BPM has probably been reached quite well. The secondary objectives (that were listed in Section 1.2) have been also covered by this thesis. For example, we have discussed the various buzzwords of BPM, standardization, relationship of BPM and Web Services, historical aspects, and many other topics. We have also shown how to automate a business process in practice.

Below we have summarized some of our main observations about BPM:

- **BPM gaining momentum** – BPM has gained a lot of new momentum in recent years as new kind of technical solutions have emerged. Particularly, the emergence of Web Service standards have had a significant effect on the development of BPM technologies in this decade.
- **Technologies still maturing** – BPM technologies are still maturing and there is quite much insecurity related to them. However, it has been already shown that the early adopters can score some impressive gains from the technologies.
- **A combination of software tools** – The term BPM is today more often considered as a term that is related to IT. Many say BPM is today more or less a collection of software tools. The tools aim at facilitating the design, execution, monitoring, analysis, and optimization of business processes in an enterprise.
- **Web Services play a centric role –** The majority of today's BPM technologies build upon Web Service technologies. The main benefit of Web Services is that they offer a standard-based and Internet-enabled application integration platform. Upon the platform it is possible to build business processes through WS composition and -languages.
- **Standardization problems –** The standardization of BPM is somewhat problematic. The fragmented BPM standardization and the lack of cooperation has lead to many misunderstood, misapplied, and incompatible standards. Only a few of the many BPM standards have become widely adopted.

- **Convergence of EAI and Workflow Management Systems** – Today's BPM systems is a result of the convergence of EAI and Workflow management systems. The BPM systems aim at combining the best sides of them. The modern BPM systems enable to integrate applications but also to build workflows and business processes.
- **Towards open technologies and Open Source** – Today's BPM solutions are characterized by that they build upon open technologies such as XML, Web Services, and BPEL. Still in the last decade most of the solutions (EAI and workflow systems) were based merely on proprietary technologies. Furthermore, many Open Source BPM solutions and technologies have emerged in the last years.
- **Core standards –** The most relevant BPM standards of today include BPEL, BPMN, and WS-CDL. They represent the de facto languages for defining executable business processes, modeling business processes, and specifying B2B processes (choreographies), respectively.

9.3 Outlook on BPM

Below we have listed a few of our thoughts about the outlook of BPM:

- **SOA & Web Services** – It is likely that more and more enterprises will adopt SOA in the near future. The role of Web Services as the main application integration technology is thus going increase.
- **Process-orientation –** Process-orientation is probably going to become a much bigger trend in enterprise computing as it is today. Some people say the process-orientation may become a trend in computing similar to object-orientation.
- **Rise of BPM systems** – The market of BPM systems is growing rapidly. Some people have predicted that BPM systems may become some day as important systems in enterprises as what Database Management Systems (DBMS) are today.
- **Merger of BPM standards –** It is likely that there is going to be mergers of BPM standards in the next few years because of the abundance of standards. It is interesting to see how the core standards of BPM (BPMN, BPEL, WS-CDL) are going to be fitted together or if they are going to merge. The standards are still partially incompatible with each other. According to [99], the next versions of BPMN are likely to render BPEL unnecessary.
- **Next decade** – It is likely that in the next decade BPM will be impacted by the Semantic Web, a movement towards a more intelligent Web. Model-driven architectures and dynamic discovery of Web Services may also have become common by then.

9.4 Further Research

Below we have listed some emerging technologies and concepts that may have future impacts on BPM and could be further researched:

- Model-driven Architectures (MDA)
- Dynamic discovery and invocation of Web Services
- Enterprise Semantic Web
- Semantic Web Enabled Web Services (SWWS) and BPEL
- Business rules and rule languages such as RuleML, BRML, SWRL
- Rapid development and auto-generation of Web Services
- Zero-Latency Enterprises (ZLE)

- Process-oriented application development
- Java Business Integration (JBI)
- Service Component Architectures (SCA)
- Business Process Networks (BPN)
- e-Business frameworks such as RosettaNet, UBL, ebXML, and UN/CEFACT
- Enterprise portals
- Business Intelligence (BI)
- Dynamic business process optimization

References

(Links accessed Jan 10, 2007)

[1] H. Smith, P. Fingar, Business Process Management: The Third Wave, Meghan-Kiffer Press 2003

[2] T. Crusson, Business Process Management Essentials, Glintech, 2006, http://www.glintech.com/downloads/BPM%20Essentials%20with%20Open%20Source.pdf

[3] S. Lusk, S. Paley, A. Spanyi, The Evolution of Business Process Management as a Professional Discipline, BPTrends.com, 2005, http://www.bptrends.com/publicationfiles/06%2D05%20WP%20ABPMP%20Activities%20%2D%20Lusk%20et%20al2%2Epdf

[4] T. Brown, The Road to BPM, Business Integration Journal, Oct 2004, http://www.bijonline.com/index.cfm?section=article&aid=325#

[5] M. Ould, Business Process Management: A Rigorous Approach, Meghan-Kiffer Press 2005,

[6] J. Chang, Business Process Management Systems: Strategy and Implementation, Auerbach 2005

[7] M. Havey, Essential Business Process Modeling, O'Reilly Media 2006

[8] R. Marks, E. Bell, Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology, Wiley 2006

[9] M. Dumas, W. van der Aalst, A. H. ter Hofstede, Process-Aware Information Systems: Bridging People and Software Through Process Technology, Wiley 2005

[10] W. Ulrich, Enterprise Application Integration, Cutter IT Journal, Vol.14:8, 2001, http://www.chaosity.com/documents/citj0801.pdf

[11] U. Endlich, An Evaluation of Enterprise Application Integration Solutions and the Role of Web Services, Master's Thesis, http://www.infosys.tuwien.ac.at/Staff/sd/DA/UlrichEndlich.pdf

[12] L. Bakker, Goodbye Hub-and-Spoke, Hello ESB?, .NET Developer's Journal, Sep. 2005, http://dndj.sys-con.com/read/121831.htm

[13] D.Oberle, Semantic Management of Middleware, Springer 2006, http://www.springer.com/cda/content/document/cda_downloaddocument/9780387276304-c2.pdf?SGWID=0-0-45-321328-p77146519

[14] S. Strüver, The Impact of Web Services in the context of Enterprise Application Integration in the Financial Services Industry, Master's Thesis, 2002 , http://www.competence-club.com/eaisysteme.nsf/AAB969E5C9BE672FC1256CA8005EEF46/$File/webservices_in_fis---struever.pdf

[15] World Wide Web Consortium, http://www.w3.org/

[16] T. Bellwood, Understanding UDDI, IBM, 2002, http://www.ibm.com/developerworks/library/ws-featuddi/

[17] Cover Pages, Universal Description, Discovery, and Integration (UDDI), Technology Report, 2005, http://xml.coverpages.org/uddi.html

[18] C. Lee, Business Process Management of Web Services, SOAWorld Magazine, Apr 2002, http://webservices.sys-con.com/read/39430.htm

[19] J. Allemann, Web Service Integration and Composition for Enabling Automatic Adaption of Heterogeneous WSDL Descriptions, Master's Thesis, University Zurich, 2006, https://www.ifi.unizh.ch/fileadmin/site/teaching/Diplomarbeiten/Abgeschlossene_Diplomarbeiten/Jahrgang_2007/Allemann_Jonas.pdf

[20] T. Baeyens, P. Fricke, Bringing BPM to the Mass Market, Business Integration Journal, Mar/Apr 2006, http://www.bijonline.com/index.cfm?section=article&aid=246

[21] M. Juric, A Hands-on Introduction to BPEL, Oracle Technology Network, http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html

[22] SearchSOA.com, Dancing with Web Services: W3C chair talks choreography, interview of Steve Ross-Talbot, 2005, http://searchsoa.techtarget.com/news/interview/0,289202,sid26_gci1066118,00.html?bucket=NEWS

[23] S. Leutenmayr, Selected Languages for Web Services Composition: Survey, Challenges, Outlook, Master's Thesis, Ludwig-Maximilians-Universität Muenchen, 2007, http://www.pms.ifi.lmu.de/publikationen/diplomarbeiten/Stephan.Leutenmayr/Diplomarbeit%20Stephan%20Leutenmayr.pdf

[23] S. Vangala, Light Weight Workflow Management System Based on Web Services, Master's Thesis, Saarland University, 2004, http://domino.mpi-inf.mpg.de/intranet/ag5/ag5publ.nsf/79510285d663a8d9c12565ee003f05dd/bf121cf08f367ff1c1256f9000371e5a/$FILE/Master_Vangala.pdf

[24] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org

[25] S. Guener, Architectural Approaches, Concepts and Methodologies of Service Oriented Architecture, Master's Thesis, 2005, Technische Universität Hamburg-Harburg

[26] S. Tähtinen, Järjestelmäintegraatio: tarve, vaihtoehdot, toteutus, Talentum, 2005,

[27] A. Arsanjani, How to Identify, Specify, and Realize Services for Your SOA, WebSphere Journal, Mar 2005, http://wsdj.sys-con.com/read/48879.htm

[28] B. Borges, K. Holley, A. Arsanjani, Delving into Service-Oriented Architecture, Developer.com, http://www.developer.com/java/ent/article.php/3409221

[29] C. Hérault, Mediation and Enterprise Service Bus, ITEA, Services for All, 2007

[30] S. Kemsley, A short history of BPM, A series of articles in Sandy Kemsley's blog, 2006, http://www.ebizq.net/blogs/column2/archives/bpmhistory/

[31] G. Samtani, Business Process Management & Web Services, SOAWorld Magazine, May 2002, http://soa.sys-con.com/read/39475.htm

[32] G. Behara, BPM and SOA: A Strategic Alliance, BPTrends.com, 2006, http://www.bptrends.com/publicationfiles/05%2D06%2DWP%2DBPM%2DSOA%2DBehara%2Epdf

[33] T. Tanrikorur, Business Process Management 101: The Basics of BPM and How to Choose the Right Suite, Intelligententerprise.com, 2007, http://www.intelligententerprise.com/print_article.jhtml?articleID=199204260

[34] Y. Hang, A. Kauranen, E. Kristola, J. Merinen, Human Interaction Management – Adding Human Factors into Business Processes Management, 2006, http://www.soberit.hut.fi/T-86/T-86.5161/2006/HIM_Humanedj_final.pdf

[35] Gartner Inc., world's leading information technology research and advisory company, http://www.gartner.com

[36] M. Rosen, BPM and SOA - Where Does One End and Other Begin?, BPTrends.com column, Jan 2006, http://www.bptrends.com/publicationfiles/01-06%20COL%20SOA%20-Where%20Does%20One%20End%20-%20Rosen.pdf

[37] W. Aalst, Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management, 2004, http://is.tm.tue.nl/staff/wvdaalst/publications/p226.pdf

[38] R. Khan, Business Process Management: A Practical Guide, Meghan-Kiffer Press 2004,

[39] A. Bog, A Visual Environment for the Simulation of Business Processes based on the Pi-Calculus, Master's Thesis, University Potsdam, 2006, http://bpt.hpi.uni-potsdam.de/pub/Public/FrankPuhlmann/AnjaBogThesisFinal.pdf

[40] M. zur Muehlen, Workflow-based Process Controlling, Logos Verlag Berlin, 2002, http://www.workflow-research.de/Publications/Book/Michael_zur_Muehlen_-_Workflow-based_Process_Controlling_(Web).pdf

[41] A. Garnemark, Workflow and Knowledge Management, Master's Thesis, Göteborg University, 2002, http://www.handels.gu.se/epc/archive/00001220/01/Nr23_AG.pdf

[42] Workflow Management Coalition, http://www.wfmc.org

[43] D. McGoveran, Taking the Pulse of BPM, BPM.com, 2005, http://www.bpm.com/FeaturePrint.asp?FeatureId=155

[44] D. Cameron, BPM to the Rescue, Web Services Journal, 2004, http://www2.sys-con.com/ITSG/virtualcd/services/archives/0307/cameron/index.html

[45] J. Mendling, Business Process Execution Language for Web Services, Vienna University of Economics and Business Administration, 2006, http://wi.wu-wien.ac.at/home/mendling/publications/emisaforum06.pdf

[47] M. Bartonitz, Wachsen die BPM- und Workflow-Lager zusammen?, BPM-guide.de, 2006, http://www.bpm-guide.de/articles/66

[48] A. Lonjon, Business Process Modeling and Standardization, MEGA, BPTrends.com, 2004, http://www.businessprocesstrends.com/publicationfiles/12%2D04%20WP%20BPM%20and%20Standardization%20%2D%20Lonjon%2Epdf

[49] B. List, B. Korherr, An Evaluation of Conceptual Business Process Modelling Languages, Vienna University of Technology, 2006, http://www.wit.at/people/korherr/publications/acmsac2006.pdf

[50] J. Recker, M. Rosemann, M. Indulska, P. Green, Business Process Modeling: A Maturing Discipline?, Eindhoven University Of Technology, 2006, http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2006/BPM-06-20.pdf

[51] K. Jonah, Model Programs, Government Computer News – GCN.com, 2002, http://www.gcn.com/print/21_27/19878-1.html

[52] Instream.co.uk, RAD – Role Activity Diagram, http://www.instream.co.uk/instream/rad.jsp

[53] Zimbio.com, Advantages For Business Process Modeling, 2007, http://www.zimbio.com/Business+Processes/articles/3/Advantages+Business+Process+Modeling

[54] Business Process Modeling Notation specification, OMG, http://www.bpmn.org

[55] E. Durocher, Business Process Management Notation: Java Graphic Implementation for Today's BPM World, Align Journal, Oct 2005, http://www.alignjournal.com/index.cfm?section=article&aid=144

[56] M. Fowler, UML Distilled, Third Edition, Addison-Wesley, 2006

[57] L. Eloranta, E. Kallio, I. Terho, A Notation Evaluation of BPMN and UML Activity Diagrams, Helsinki University of Technology, 2006, http://www.soberit.hut.fi/T-86/T-86.5161/2006/BPMN_vs_UML_final.pdf

[58] M. Owen, J. Raj, BPMN and Business Process Management, BPMN.org / Popkin Software, 2003, http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf

[59] S. White, Process Modeling Notations and Workflow Patterns, IBM, http://www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf

[60] Perspectives in Process, Phil Gilbert's blog, http://blog.lombardicto.com/

[61] BPMS-Watch, Bruce Silver's blog, http://www.brsilver.com/wordpress/

[62] Go Flow, Keith Swenson's blog, http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel/

[63] P. Stevens, XMI and MOF: a mini-tutorial, presentation slides, University of Edinburgh, http://homepages.inf.ed.ac.uk/perdita/XMI/tutslides2up.pdf

[64] J.C. Ibanez, Evaluation of the BPEL Specification for Business Process Composition, Licentiate Thesis, University of Gent, 2007, http://faramir.ugent.be/thesis/Clua.pdf

[65] B. Lublinsky, An Introduction to Business Process Execution Language (BPEL), Business Integration Journal, Oct 2004, http://www.alignjournal.com/index.cfm?section=article&aid=77

[66] D. Townsend, Using BPEL, Web Services Journal, 2004, http://www2.sys-con.com/ITSG/virtualcd/WebServices/archives/0307/cairns/index.html

[67] C.. Stahl, A Petri Net Semantics for BPEL, Humboldt-Universität zu Berlin, http://www2.informatik.hu-berlin.de/Institut/struktur/systemanalyse/preprint/stahl188.pdf

[68] Y. Goland, BPEL4WS – The Promise of Portable Business Processes, Web Services Journal, 2004, http://www2.sys-con.com/ITSG/virtualcd/WebServices/archives/0212/goland/index.html

[69] Devarticles.com, Orchestrating Web Services, 2006, http://www.devarticles.com/c/a/Web-Services/Orchestrating-Web-Services/

[70] B. May, P. Savur, Business Process Execution Language, Sun Developer Network, 2006, http://developers.sun.com/jsenterprise/nb_enterprise_pack/reference/techart/bpel.html

[71] D. Sherman, BPEL: Make Your Services Flow, Web Services Journal, 2004, http://www2.sys-con.com/ITSG/virtualcd/WebServices/archives/0307/sherman/index.html

[72] JBoss.com, JBoss jBPM, WS-BPEL Runtime User Guide, http://docs.jboss.com/jbpm/bpel/v1.1/userguide/index.html

[73] T. Piispanen, BPEL & SOA, 2006, http://users.tkk.fi/~tpiispan/BPEL_SOA.pdf

[74] F. Perkonigg, Application of Business Process Management in Web-Based Education, Master's Thesis, Graz University of Technology, 2006, http://www.iicm.tu-graz.ac.at/thesis/fperkonigg.pdf

[75] M. Koskimies, Composing Services in SOA: Workflow Design, Usage and Patterns, Research Seminar, University of Helsinki, http://www.cs.helsinki.fi/u/thruokol/opetus/2006/sose/papers/koskimies_workflows.pdf

[76] O. Ezenwoye, S. Sadjadi, Composing Aggregate Web Services in BPEL, Florida International University, 2006, http://www.cs.fiu.edu/~sadjadi/Publications/AggregateWS-ACMSE-2006.pdf

[77] D. Linthicum, WS-BPEL 2.0: Not Backward Compatible?, SOAWorld Magazine, 2006, http://webservices.sys-con.com/read/291050.htm

[78] E. Pieczkowski, Is BPEL Positioned for the Future?, Networkcomputing.com, 2007, http://www.networkcomputing.com/article/printFullArticle.jhtml?articleID=198000132

[79] M. Havey, BPEL SOA and Web Services for Java, 2006, http://wldj.sys-con.com/read/169346.htm

[80] Cover Pages, BPEL4People Specifications Integrate Human Interactions Into Business Process, 2007, http://xml.coverpages.org/ni2007-06-25-a.html

[81] BEA.com, BPEL4People and WS-HumanTask, specification, 2007, http://dev2dev.bea.com/pub/a/2007/06/bpel4people.html

[82] IBM, WS-BPEL 2.0 Extensions for Sub-Processes, specification, 2005, http://www.ibm.com/developerworks/library/specification/ws-bpelsubproc/

[83] Z. Li, Using Collaboration Task In Orbwork Enactment System For Meteor Workflow Management System, 1996, http://lsdis.cs.uga.edu/lib/download/thesis/ZqLi-Thesis.pdf

[84] K. Stoilova, T. Stoilov, Evolution of the Workflow Management Systems, 2006, http://www.visp-project.org/docs/publications/ICEST06_12.pdf

[85] H. Smith, P. Fingar, Workflow is just a Pi process, Fairdene, 2003, http://www.fairdene.com/picalculus/workflow-is-just-a-pi-process.pdf

[86] Networkcomputing.com, A Review of Business Process Management Systems, 2007, http://www.networkcomputing.com/article/printFullArticle.jhtml?articleID=165600046

[87] Neudesic, Neuron ESB: An Enterprise Service Bus for the Microsoft Platform, 2006, http://www.neudesic.com/media/Neudesic%20-%20Neuron%20ESB%20White%20Paper%20(P1).pdf

[88] N. Thomas, The Enterprise Service Bus, SOAWorld Magazine, Sep 2003, http://webservices.sys-con.com/read/39873.htm

[89] K. Kobielus, Enterprise Service Bus: Web Services Meet Message-Oriented Middleware, Red Orbit, 2006, http://www.redorbit.com/news/technology/478926/enterprise_service_bus_web_services_meet_messageoriented_middleware/

[90] D. Chappell, ESB Integration Patterns, SOAWorld Magazine, Aug 2004, http://webservices.sys-con.com/read/46170.htm

[91] A. Charfi, Aspect-Oriented Workflow Languages: AO4BPEL and Applications, Doctoral Thesis, Darmstadt University of Technology, 2007, http://www.st.informatik.tu-darmstadt.de/pages/projects/AO4BPEL/diss.pdf

[92] User:Jayvdb, Wikipedia, Comparison of BPEL engines, http://en.wikipedia.org/wiki/User:Jayvdb/Comparison_of_BPEL_engines

[93] Netbeans Open Source programming IDE, http://www.netbeans.org

[94] H. Raj, How to Deliver Composite Applications with Java, WS-BPEL & SOA, Java Developer's Journal, Apr 2007, http://jdj.sys-con.com/read/358049_1.htm

[95] Y. Gao, BPMN-BPEL Transformation and Round Trip Engineering, eClarus, 2006, http://www.eclarus.com/pdf/BPMN_BPEL_Mapping.pdf

[96] J. Mendling, M. Hafner, From WS-CDL Choreography to BPEL Process Orchestration, Vienna University of Economics and Business Administration, 2006,http://wi.wu-wien.ac.at/home/mendling/publications/TR06-CDL.pdf

[97] M. Havey, What Is Business Process Modeling, OnJava.com (O'Reilly), 2005 http://www.onjava.com/pub/a/onjava/2005/07/20/businessprocessmodeling.html

[98] SMSLib, Java library for sending and receiving SMS message via a GSM modem, http://www.smslib.org

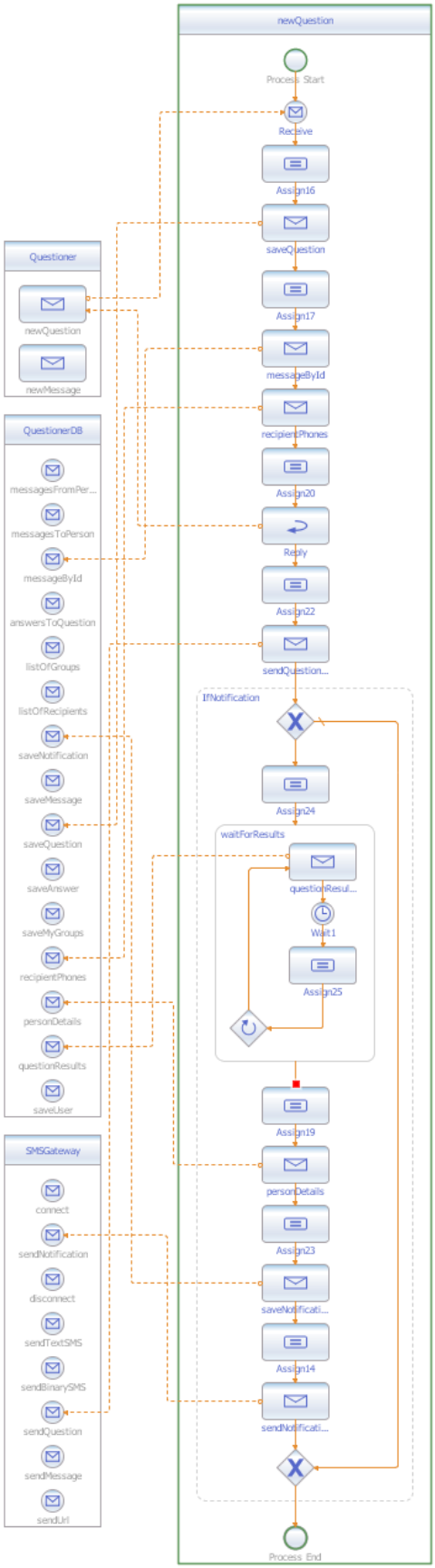
[99] J. de Jong, Could BPMN 2.0 Make BPEL a Moot Point, SD Times, Jul 2006, http://www.sdtimes.com/article/story-20060701-02.html

[100] M. Juric, BPEL and Java, Theserverside.com, Apr 2005, http://www.theserverside.com/tt/articles/article.tss?l=BPELJava

[101] IT Redux, Ismail Ghalimi's blog, http://itredux.com/blog/2007/04/04/get-your-bpmn-schema-today/

Appendix A

In this appendix is presented the process diagram, the WSDL, and the BPEL process definition of the Questioner Web Service.



---------- WSDL ----------

```
<definitions
   targetNamespace="http://hiit.fi/S4All/Questioner"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   name="Questioner"
   xmlns:plink="http://schemas.xmlsoap.org/ws/2004/03/partner-link/"
                                          xmlns:tns="http://hiit.fi/S4All/Questioner"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2004/03/partner-link/">
   <types/>
   <message name="question">
       <part name="myPid" type="xsd:int"/>
       <part name="text" type="xsd:string"/>
       <part name="expiresInMinutes" type="xsd:int"/>
       <part name="answerChoices" type="xsd:string"/>
       <part name="actions" type="xsd:string"/>
       <part name="toGids" type="xsd:string"/>
       <part name="toPids" type="xsd:string"/>
   </message>
   <message name="return">
       <part name="newMid" type="xsd:string"/>
       <part name="smsTo" type="xsd:string"/>
   </message>
   <message name="message">
       <part name="myPid" type="xsd:int"/>
       <part name="text" type="xsd:string"/>
       <part name="toGids" type="xsd:string"/>
       <part name="toPids" type="xsd:string"/>
   </message>
   <portType name="PrivateOperations">
       <operation name="newQuestion">
           <input name="input1" message="tns:question"/>
           <output name="output1" message="tns:return"/>
       </operation>
       <operation name="newMessage">
           <input name="input2" message="tns:message"/>
           <output name="output2" message="tns:return"/>
       </operation>
   </portType>
   <binding name="QuestionerBinding" type="tns:PrivateOperations">
       <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="newQuestion">
           <soap:operation style="rpc"/>
           <input name="input1">
               <soap:body use="literal" namespace="http://hiit.fi/S4All/Questioner"/>
           </input>
           <output name="output1">
               <soap:body use="literal" namespace="http://hiit.fi/S4All/Questioner"/>
           </output>
       </operation>
       <operation name="newMessage">
           <soap:operation style="rpc"/>
           <input name="input2">
               <soap:body use="literal" namespace="http://hiit.fi/S4All/Questioner"/>
           </input>
           <output name="output2">
               <soap:body use="literal" namespace="http://hiit.fi/S4All/Questioner"/>
           </output>
       </operation>
   </binding>
   <service name="QuestionerService">
       <port name="QuestionerPort" binding="tns:QuestionerBinding">
           <soap:address location="http://localhost:8087/bpel/QuestionerService"/>
       </port>
   </service>
   <plink:partnerLinkType name="QuestionerPartner">
       <plink:role name="Provider" portType="tns:PrivateOperations"/>
   </plink:partnerLinkType>
   <plnk:partnerLinkType name="Questioner_PLT">
       <plnk:role name="Provider" portType="tns:PrivateOperations"/>
   </plnk:partnerLinkType>
</definitions>
```

---------- BPEL ----------

```
<process
   name="newQuestion"
   targetNamespace="http://hiit.fi/S4All/Questioner/BPEL"
   xmlns="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
   xmlns:ns7="http://hiit.fi/S4All/QuestionerDB"
   xmlns:ns8="http://hiit.fi/S4All/SMSGateway"
   xmlns:ns10="http://hiit.fi/S4All/Questioner"
   xmlns:ns1="http://www.w3.org/2001/XMLSchema">
                    <import       namespace="http://hiit.fi/S4All/SMSGateway"
     location="WSDL/SMSGateway.wsdl"
     importType="http://schemas.xmlsoap.org/wsdl/"/>
                   <import       namespace="http://hiit.fi/S4All/QuestionerDB"
     location="WSDL/QuestionerDB.wsdl"
     importType="http://schemas.xmlsoap.org/wsdl/"/>
                    <import       namespace="http://hiit.fi/S4All/Questioner"
     location="WSDL/Questioner.wsdl"
     importType="http://schemas.xmlsoap.org/wsdl/"/>
   <partnerLinks>
       <partnerLink name="Questioner" partnerLinkType="ns10:Questioner_PLT"
          myRole="Provider"/>
       <partnerLink name="QuestionerDB" partnerLinkType="ns7:QuestionerDB_PLT"
          partnerRole="Provider"/>
       <partnerLink name="SMSGateway" partnerLinkType="ns8:SMSGatewayPLT"
          partnerRole="Provider"/>
   </partnerLinks>
   <variables>
      <variable name="QuestionResultsOut" messageType="ns7:questionResultsResponse"/>
      <variable name="QuestionResultsIn" messageType="ns7:questionResults"/>
      <variable name="minsLeft" type="ns1:int"/>
      <variable name="AnswersToQuestionOut"
       messageType="ns7:answersToQuestionResponse"/>
      <variable name="AnswersToQuestionIn" messageType="ns7:answersToQuestion"/>
                                    <variable      name="SaveNotificationOut"
       messageType="ns7:saveNotificationResponse"/>
      <variable name="SaveNotificationIn" messageType="ns7:saveNotification"/>
      <variable name="MessageByIdOut" messageType="ns7:messageByIdResponse"/>
      <variable name="MessageByIdIn" messageType="ns7:messageById"/>
      <variable name="reply" messageType="ns10:return"/>
      <variable name="PersonDetailsOut" messageType="ns7:personDetailsResponse"/>
      <variable name="PersonDetailsIn" messageType="ns7:personDetails"/>
                                    <variable      name="SendNotificationOut"
       messageType="ns8:sendNotificationResponse"/>
      <variable name="SendNotificationIn" messageType="ns8:sendNotification"/>
      <variable name="SendQuestionIn" messageType="ns8:sendQuestion"/>
      <variable name="RecipientPhonesOut" messageType="ns7:recipientPhonesResponse"/>
      <variable name="RecipientPhonesIn" messageType="ns7:recipientPhones"/>
      <variable name="SaveQuestionOut1" messageType="ns7:saveQuestionResponse"/>
      <variable name="SaveQuestionIn1" messageType="ns7:saveQuestion"/>
      <variable name="request" messageType="ns10:question"/>
      <variable name="SendQuestionOut" messageType="ns8:sendQuestionResponse"/>
   </variables>
   <sequence name="Process">
       <sequence name="SaveQToDB">
                   <receive name="Receive" createInstance="yes"
                      partnerLink="Questioner" operation="newQuestion"
                      portType="ns10:PrivateOperations" variable="request"/>
                   <assign name="Assign16">
                       <copy>
                           <from variable="request" part="myPid"/>
                           <to variable="SaveQuestionIn1" part="myPid"/>
                       </copy>
                       <copy>
                           <from variable="request" part="text"/>
                           <to variable="SaveQuestionIn1" part="text"/>
                       </copy>
                       <copy>
                           <from variable="request" part="expiresInMinutes"/>
                           <to variable="SaveQuestionIn1" part="expiresInMinutes"/>
                       </copy>
                       <copy>
                           <from variable="request" part="toGids"/>
                           <to variable="SaveQuestionIn1" part="toGids"/>
                       </copy>
                       <copy>
                           <from variable="request" part="toPids"/>
                           <to variable="SaveQuestionIn1" part="toPids"/>
                       </copy>
                       <copy>
                           <from variable="request" part="answerChoices"/>
```

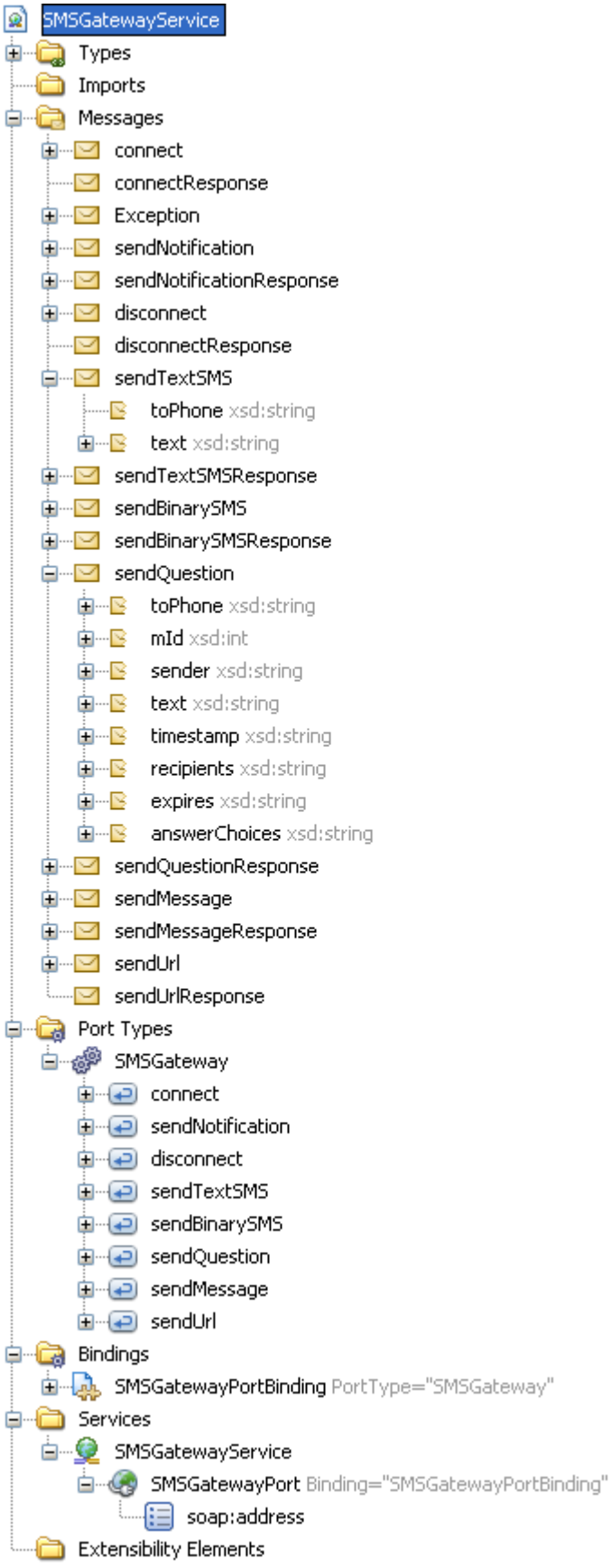
```
                <to variable="SaveQuestionIn1" part="answerChoices"/>
            </copy>
            <copy>
                <from variable="request" part="actions"/>
                <to variable="SaveQuestionIn1" part="actions"/>
            </copy>
        </assign>
        <invoke name="saveQuestion" partnerLink="QuestionerDB"
           operation="saveQuestion" portType="ns7:QuestionerDB"
           inputVariable="SaveQuestionIn1"
           outputVariable="SaveQuestionOut1"/>
    </sequence>
    <sequence name="SendQSMSs">
    <assign name="Assign17">
        <copy>
            <from variable="SaveQuestionOut1" part="return"/>
            <to variable="RecipientPhonesIn" part="mId"/>
        </copy>
        <copy>
            <from variable="SaveQuestionOut1" part="return"/>
            <to variable="MessageByIdIn" part="mId"/>
        </copy>
    </assign>
        <invoke name="messageById" partnerLink="QuestionerDB"
           operation="messageById" portType="ns7:QuestionerDB"
           inputVariable="MessageByIdIn" outputVariable="MessageByIdOut"/>
        <invoke name="recipientPhones" partnerLink="QuestionerDB"
           operation="recipientPhones" portType="ns7:QuestionerDB"
           inputVariable="RecipientPhonesIn"
           outputVariable="RecipientPhonesOut"/>
            <assign name="Assign20">
        <copy>
            <from variable="SaveQuestionOut1" part="return"/>
            <to variable="reply" part="newMid"/>
        </copy>
        <copy>
            <from variable="RecipientPhonesOut" part="return"/>
            <to variable="reply" part="smsTo"/>
        </copy>
    </assign>
            <reply name="Reply" partnerLink="Questioner"
                operation="newQuestion" portType="ns10:PrivateOperations"
               variable="reply"/>
            <assign name="Assign22">
        <copy>
            <from variable="RecipientPhonesOut" part="return"/>
            <to variable="SendQuestionIn" part="toPhone"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/sender</from>
            <to variable="SendQuestionIn" part="sender"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/text</from>
            <to variable="SendQuestionIn" part="text"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/timestamp</from>
            <to variable="SendQuestionIn" part="timestamp"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/expires</from>
            <to variable="SendQuestionIn" part="expires"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/recipients</from>
            <to variable="SendQuestionIn" part="recipients"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/answerChoices</from>
            <to variable="SendQuestionIn" part="answerChoices"/>
        </copy>
        <copy>
            <from>$MessageByIdOut.return/message/id</from>
            <to variable="SendQuestionIn" part="mId"/>
        </copy>
    </assign>
    <invoke name="sendQuestionBySMS" partnerLink="SMSGateway"
        operation="sendQuestion" portType="ns8:SMSGateway"
        inputVariable="SendQuestionIn" outputVariable="SendQuestionOut"/>
```

```
        </sequence>
    <if name="IfNotification">
        <condition>$request.actions != ''</condition>
        <sequence name="sendQuestionResults">
            <assign name="Assign24">
                <copy>
                    <from variable="request" part="expiresInMinutes"/>
                    <to variable="minsLeft"/>
                </copy>
                <copy>
                    <from variable="SaveQuestionOut1" part="return"/>
                    <to variable="QuestionResultsIn" part="mId"/>
                </copy>
            </assign>
            <repeatUntil name="waitForResults">
                    <sequence name="seq1">
                         <invoke name="questionResults" partnerLink="QuestionerDB"
                            operation="questionResults" portType="ns7:QuestionerDB"
                            inputVariable="QuestionResultsIn"
                            outputVariable="QuestionResultsOut"/>
                        <wait name="Wait1">
                            <for>'P0Y0M0DT0H0M2S'</for>
                        </wait>
                        <assign name="Assign25">
                            <copy>
                                <from>( $minsLeft - 1 )</from>
                                 <to variable="minsLeft"/>
                            </copy>
                        </assign>
                    </sequence>
                <condition>( $QuestionResultsOut.return != 'wait' )</condition>
            </repeatUntil>
                <sequence name="SaveNotificationAndSend">
            <assign name="Assign19">
                <copy>
                    <from variable="request" part="myPid"/>
                    <to variable="PersonDetailsIn" part="pId"/>
                </copy>
            </assign>
            <invoke name="personDetails" partnerLink="QuestionerDB"
              operation="personDetails" portType="ns7:QuestionerDB"
              inputVariable="PersonDetailsIn"outputVariable="PersonDetailsOut"/>
                <assign name="Assign23">
                    <copy>
                        <from variable="request" part="myPid"/>
                        <to variable="SaveNotificationIn" part="toPids"/>
                    </copy>
                    <copy>
                        <from variable="QuestionResultsOut" part="return"/>
                        <to variable="SaveNotificationIn" part="text"/>
                    </copy>
                </assign>
                <invoke name="saveNotification" partnerLink="QuestionerDB"
                   operation="saveNotification" portType="ns7:QuestionerDB"
                   inputVariable="SaveNotificationIn"
                   outputVariable="SaveNotificationOut"/>
                <assign name="Assign14">
                <copy>
                    <from><literal>Server</literal></from>
                    <to part="sender" variable="SendNotificationIn"/>
                </copy>
                <copy>
                    <from>$PersonDetailsOut.return/name</from>
                    <to variable="SendNotificationIn" part="recipients"/>
                </copy>
                <copy>
                    <from>$PersonDetailsOut.return/phone</from>
                    <to variable="SendNotificationIn" part="toPhone"/>
                </copy>
                <copy>
                    <from>concat(substring(current-date(), 0, 11), ' ',
                       substring(current-time(), 0, 6))</from>
                    <to part="timestamp" variable="SendNotificationIn"/>
                </copy>
                    <copy>
                        <from variable="SaveNotificationOut" part="return"/>
                        <to variable="SendNotificationIn" part="mId"/>
                    </copy>
                    <copy>
                        <from variable="QuestionResultsOut" part="return"/>
```

```
                        <to variable="SendNotificationIn" part="text"/>
                    </copy>
                </assign>
                <invoke name="sendNotificationBySMS" partnerLink="SMSGateway"
                   operation="sendNotification" portType="ns8:SMSGateway"
                   inputVariable="SendNotificationIn"
                   outputVariable="SendNotificationOut"/>
          </sequence>
        </sequence>
      </if>
   </sequence>
</process>
```

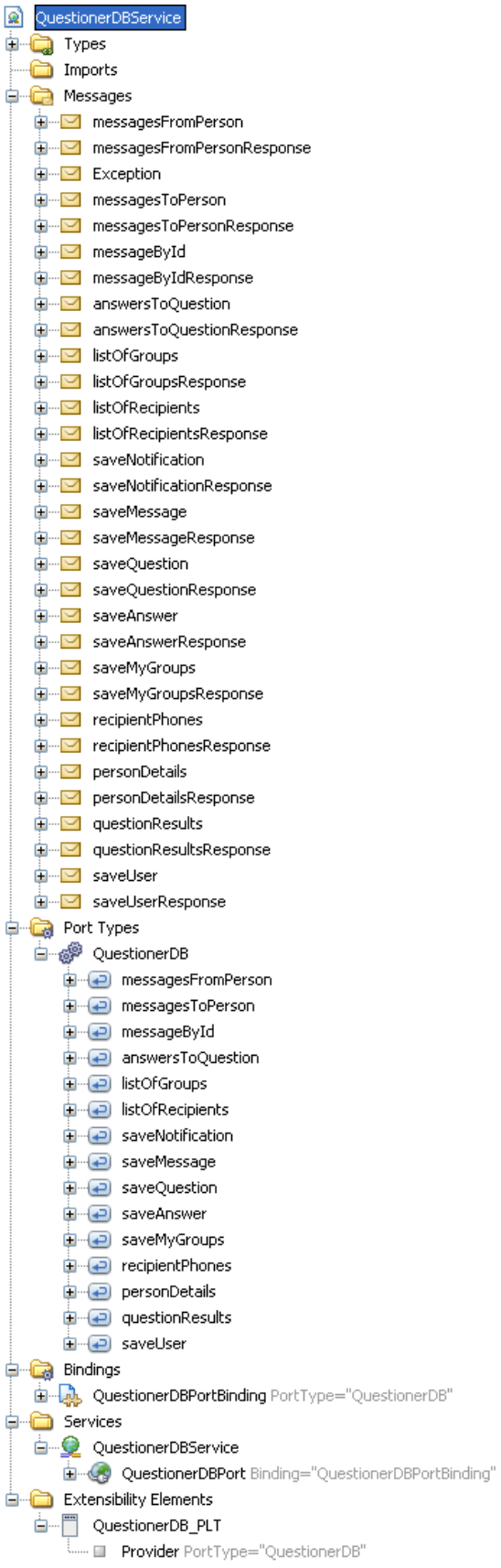
Appendix B

In this appendix is presented the WSDL of the SMS Gateway Web Service component of the Questioner application.



Appendix C

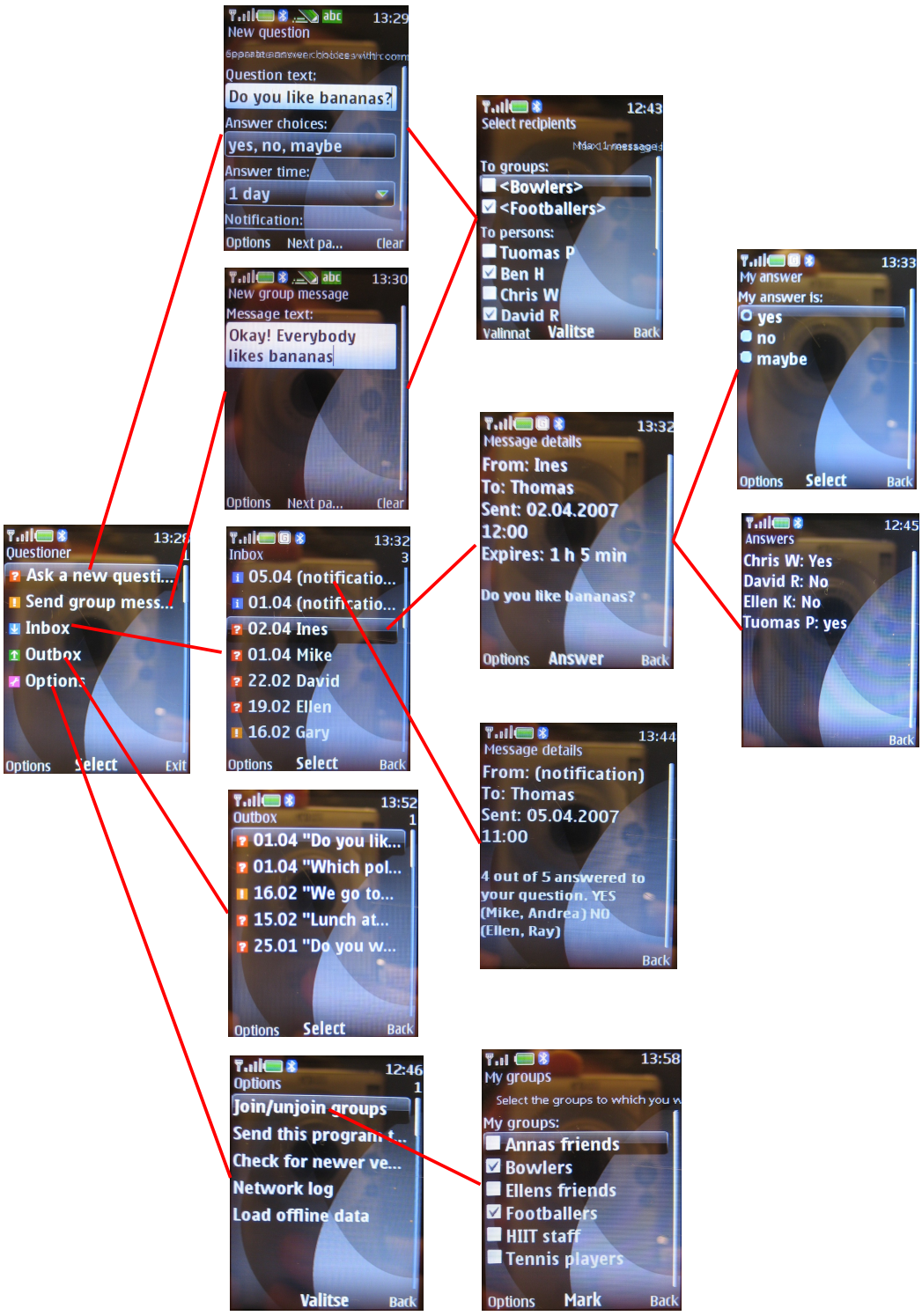
In this appendix is presented the WSDL of the Database Web Service component of the Questioner application and. The database schema of the database can be found further below.

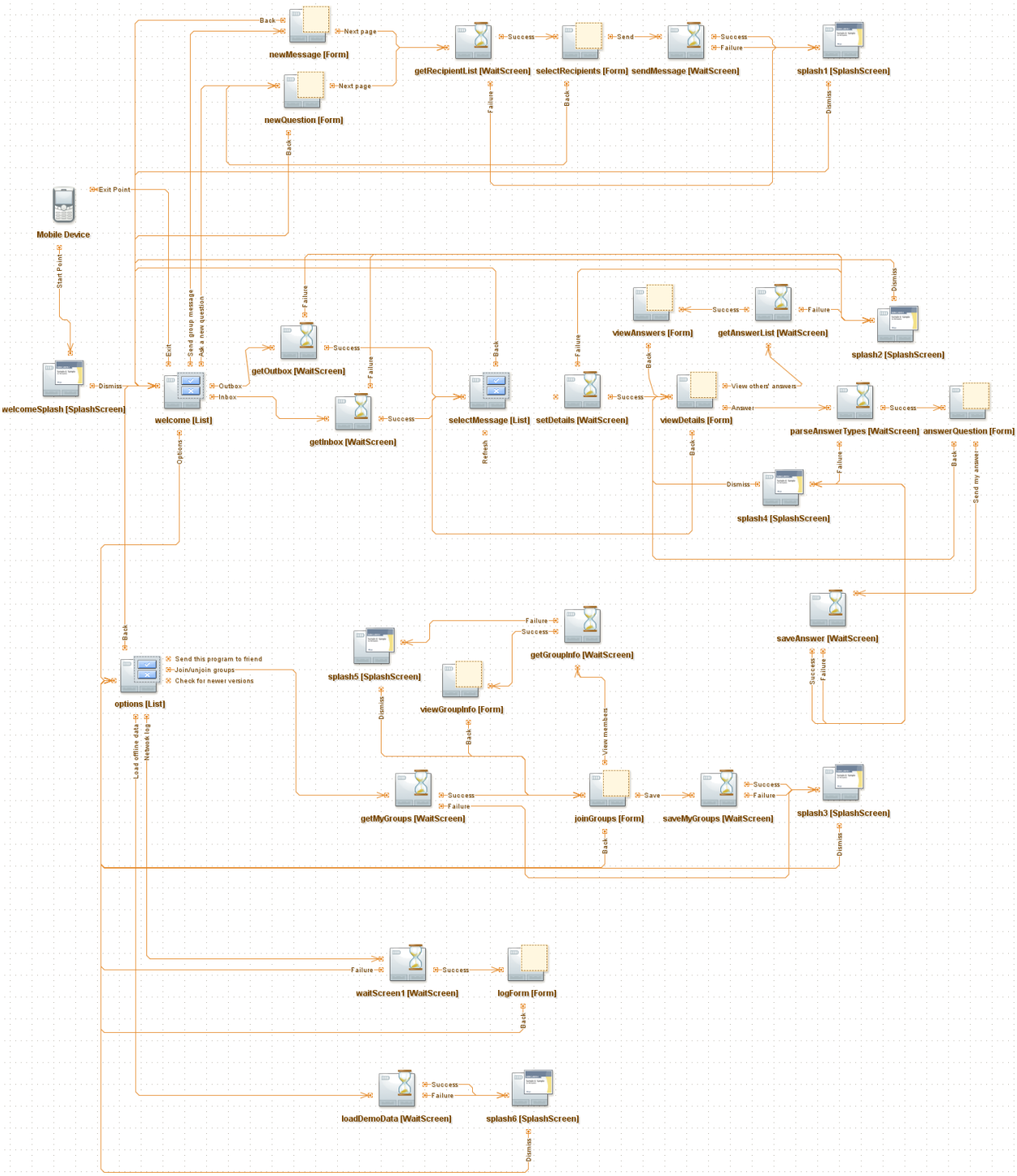


```
create table grouphaspersons
    (gp_id smallint generated always as identity primary key,
     gp_gid smallint,
     gp_pid smallint);

create table groups
    (g_id smallint generated by default as identity primary key,
     g_name varchar(30),
     g_description varchar(50));

create table persons
    (p_id smallint generated by default as identity primary key,
     p_name varchar(15),
     p_phone varchar(30),
     p_email varchar(50),
     p_fullname varchar(50));

create table messages
    (m_id smallint generated by default as identity primary key,
     m_pid smallint,
     m_type varchar(15),
     m_actions varchar(30),
     m_answerchoices varchar(100),
     m_text varchar(250),
     m_ts timestamp,
     m_expires_ts timestamp);

create table answers
    (a_id smallint generated always as identity primary key,
     a_mid smallint,
     a_pid smallint,
     a_answer varchar(50));

create table messagestopersons
    (mp_id smallint generated always as identity primary key,
     mp_mid smallint,
     mp_pid smallint);

create table messagestogroups
    (mg_id smallint generated always as identity primary key,
     mg_mid smallint,
     mg_gid smallint);
```

Appendix D

In this appendix is presented some screenshots of the Questioner mobile client program and its screen map.




Back
Next page
newMessage [Form]
getRecipientList [WaitScreen]
Success
selectRecipients [Form]
Send
sendMessage [WaitScreen]
Success
Failure
splash1 [SplashScreen]
Next page
newQuestion [Form]
Failure
Back
Dismiss
Back
Exit Point
Mobile Device
Start Point
Exit
Send group message
Ask a new question
Failure
getOutbox [WaitScreen]
Success
Failure
Back
Failure
Back
viewAnswers [Form]
Success
getAnswerList [WaitScreen]
Failure
Dismiss
splash2 [SplashScreen]
Dismiss
welcomeSplash [SplashScreen]
welcome [List]
Outbox
Inbox
getInbox [WaitScreen]
Success
selectMessage [List]
setDetails [WaitScreen]
Success
Back
viewDetails [Form]
View others' answers
Answer
parseAnswerTypes [WaitScreen]
Success
answerQuestion [Form]
Options
Refresh
Back
Failure
Back
Dismiss
splash4 [SplashScreen]
Send my answer
saveAnswer [WaitScreen]
Back
Send this program to friend
Join/unjoin groups
Check for newer versions
options [List]
Load offline data
Network log
splash5 [SplashScreen]
Failure
Success
getGroupInfo [WaitScreen]
viewGroupInfo [Form]
Success
Failure
Dismiss
Back
View members
getMyGroups [WaitScreen]
Success
Failure
joinGroups [Form]
Save
saveMyGroups [WaitScreen]
Success
Failure
splash3 [SplashScreen]
Back
Dismiss
Failure
Success
waitScreen1 [WaitScreen]
logForm [Form]
Back
Success
Failure
loadDemoData [WaitScreen]
splash6 [SplashScreen]
Dismiss