Master's Thesis

Application of Business Process Management in Web-Based Education

Fidelis Perkonigg

Institute for Information Systems and Computer Media (IICM)
Graz University of Technology
Head: O. Univ.-Prof. Dr. phil. Hermann Maurer

Supervisor: Dipl.-Ing. Dr. techn. Denis Helic

Graz, September 2006

Abstract

This thesis deals with Business Process Management and its possible application in the field of Web-based Education and learning processes. To investigate these possibilities in practice, a tool that supports learning processes through Business Process Management has been developed. The thesis presents the software development process of that tool in detail. Thus, the thesis is divided into three main parts.

The first part discusses the current state of Business Process Management in detail. It introduces concepts, technologies and tools to facilitate the Business Process Management by supporting the whole business process life cycle in a seamless way. For each phase of the process life cycle, that are, model, implement, execute, analysis, and optimize, emerging standards are presented which help manage quick changing business processes.

The second part argues that Web-based Education deals with similar issues as Business Process Management. Moreover, learning processes have the same life cycle as business processes and can be treated like them. Thus, concepts and technologies that support Business Process Management can be applied. In addition, this part of the thesis also evaluates a Learning Process Management System to show the usefulness of this idea. Finally, it identifies missing tools in that system that can improve management of the learning process lifecycle.

The last part of the thesis presents a complete software development process of one of such missing tools for the Learning Process Management System. This tool improves the modeling phase of the learning process life cycle.

Contents

List of Figures

Listings

1. Introduction and Motivation

"To improve is to change;
to be perfect is to change often." – Winston Churchill

"In many cases, e-learning may be the only way forward. The future demand for business education is such that it is unlikely that traditional or campus-based education will be able to cope. A solution needs to be found rapidly so that growth rates in the world economy can be sustained". Did you read this comment from Ms Lange at U21 Global, an online graduate school, in the Financial Times on Monday the 20th of March 2006? Further, did you know that online enrolment for e-learning courses in the US increased from 1.98 million in 2003 to 2.35 million a year later? This is a growth rate more that 10 times. Not only the US, Europe is also investing hundreds of thousands of Euros to establish e-learning programmes. There is also already a huge market potential in Asia, especially in less developed coutries where there is less access to places of higher education.

"Technology leads the way as online learning comes of age", was the title of the special report in the Financial Times where all these facts and much more was written.

Everyone seems to be investing in e-learning, seems to be speaking about it, and many are already using it. Thus, is it not worth looking at the technical background of e-learning, also called distance learning or web-based education? Now you will probably think that there are already a lot of books, articles on the internet or in journals out there that cover this very hot topic. That is very true but it is also the reason why we want to look at it from a different view.

This paper argues that e-learning tasks can be treated like business processes because of their very similar nature. Consequently, well established technologies to support business processes can also be used for e-learning managing learning processes. In addition, a tool has been developed which contributes to an e-learning system based on technology for Business Process Management.

This thesis is divided into three main parts to give you a clear picture of Business Process Management. First, it is given a brief overview about business processes. It covers the definition of business processes, what they are good for and the arising problems to manage and implement them in an organisation. This is followed by a look at the state of the art of concepts, technologies and tools to tackle and overcome the existing problems.

Then, e-learning is introduced. After motivating why it is the future of education, this paper takes a look at some e-learning systems, especially at the e-learning system running at the Institute for Information Systems and Computer Media (IICM) at Graz

University of Technology. It is shown that e-learning tasks can be treated as business processes and technology used for Business Process Management can be applied. Despite this remarkable concept, the system still lacks the ease of operation for the users. A novel has been developed to solve this problem.

The third and last part of this paper describes the development of the program. It does not start with the implementation immediately. It rather motivates the need for a structured software development process. Then, all stages of it are described in detail until the program is ready for use.

Finally, the paper proviedes concluding remarks. These three parts of the paper offer a good insight about technical concepts, technologies and tools to manage business processes and learning processes. In addition, the whole software development process is presented.

1.1. The Business Process

We ought to start off with the definition of a business process. Sometimes the term business processes is used to talk about high-level descriptions of organisation's activities which are also used to describe market-oriented aspects like the satisfaction of customer needs [ZR]. Also [Wes99] uses a pretty similar definition. "The business process comprises all activities carried out in an enterprise, including e.g. staffing, financing, production, marketing, etc."

Figure 1.1 on page 3 shows a rather general model of business processes. In general, a business process comprises activities that produce an output of value to the customer. A business process can be thought of as a box turning a certain input into an output of greater value [Ham97]. Usually it is the desired output of the customer but it can also add value to the company itself.

The business process of product development, for instance, has an idea or concept as an input and gets a design or prototype as an output. In order to get the desired output, many kind of people have to participate, such as, research and development (contributing their technical expertise), marketing (offer knowledge of customers' needs), manufactoring experts and finance people.

Looking back at the box, you can find the process activities which are the means to transform the input into the desired output. At any time the activities have a specific state and the process flow is determined by business rules [Roh].

In other words, a business process can be thought of as a cookbook for running a business. It is a recipe to achieve a commercial result.

1.1.1. Importance of Business Processes

History showed us the importance of business processes. In the 1980s the inefficiencies and inaccuracies of company performances were beginning to matter. Before, customers had not had much choice where to buy specific products, but in the 1980s customers

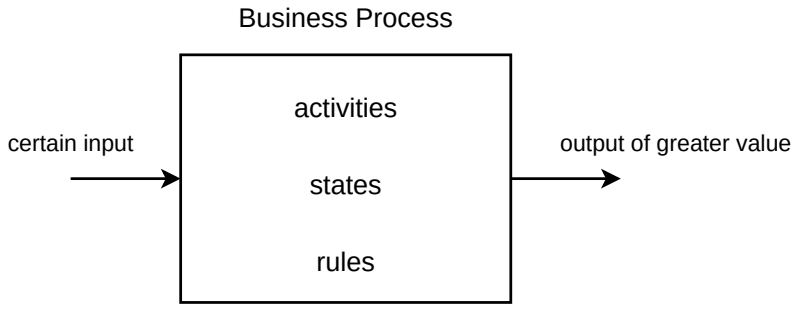


Figure 1.1.: Shows the general model of business processes

started deserting to other companies when they were not satisfied because they had simply the choice due to increasing competition.

At that time many corporations had to improve their business to keep their customers. After attempts to change their situation, they found out the main problem. It was the fact that they were applying task solutions to process problems. A task is a unit of work usually performed by one person. In contrast, a process comprises of tasks and activities to create a result of value to the customer [Ham97].

That was the time when organisations got aware of the importance of business processes. It is vital for the survival to let them be at the heart of the company.

In other words, business processes are the key to a successful business. This is probably the case as they focus on creating value for customers. They alone are not the solution, though. We need also Business Process Management to contribute to this concept.

1.1.2. Business Process Management

Business Process Management (BPM) is needed to keep pace with the changing business environment and its core discipline is the continuous improvement of business processes.

Today the redesign of business processes is becoming more and more difficult. This is due to the fact that it changes not only how work is done but also changes where the work is done and by whom. Specific work is moved to companies where you can get best performance for the lowest price. This trend is called outsourcing and offshoring [Cha06].

Not only does the redesign become more difficult, the intervals of changes get shorter and shorter. Companies want to improve the productivity as quickly as possible to stay ahead of their competitors. Thus, they change and improve their business processes more often. [Mie06] says that four to five business optimization cycles a year are usual and good practice.

All these examples show that, in fact, a finished business process does not exist. It takes multiple iterations to produce an effective solution and it is even more likely to change an existing process again due to changes in the business environment. It seems

that to manage a business successfully, we have to manage the steady changing business processes.

However, in general, enterprises make agile course corrections in terms of persuing strategic initiatives with confidence, including mergers, consolidation, alliances, acquisitions, outsourcing and global expansion, and process management is the only way to manage and achieve these objectives. It is the aim of Business Process Management to strengthen the ability to achieve the company's goals in an environment with growing complexity [BSW].

Thus, we can say that Business Process Management manages the life cycle of processes with respect to improvement and optimization to strengthen the ability to achieve the company's goals [SF03].

Usually this management activity is aided by software tools or also called BPM suites offered by BPM systems. Let us look at the process life cycle (refer to Figure 1.2 on page 4) to understand what these tools must be able to cope with.

The desire to respond to changing business environments by means of adaptive processes, which can be changed and optimized, has resulted in a closed loop. To put it simple, the development of a business process keeps going without approaching a final state.

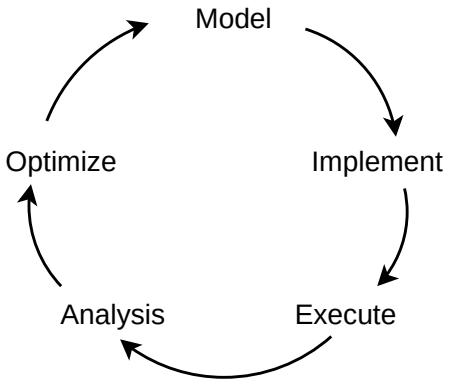


Figure 1.2.: Shows the business process life cycle

Looking at the business process life cycle, the following activities can be identified:

Model: During this activity business process owners or analysts design new, modify or capture existing processes. They create a high level design of tasks to be performed usually supported by graphical editors.

Implement: The aim of this activity is to convert the high level business process from the high level definition to an executable process model. In other words, it is implemented into the real-world environment of the company including all kind of employees and resources.

Execute: During this activity the implementation is executed and is supposed to create desired output for the customer or to add value to the company in general.

Analysis: This activity monitors and analysises business processes that are being executed to improve them in the next step.

Optimize: After the running business process has been monitored, the collected data is used to optimize and to redesign the process to improve performance.

Business Process Management Systems must include many features to be able to cope with the whole process life cycle. In general, it must include process modeling support, business rules environments, integration facilities, sophisticated user interfaces and powerful analytic tools. Furthermore, the management system should be capable of giving direct access to all participants of the change process in terms of events, business rules, user interfaces, process flows, analysis etc. [Mie06].

It is not enough to consider the features of the Business Process Management System only. The business process itself and all third party applications have to support Business Process Management as well. For example, if an application does not offer any monitoring data about its activity, the Business Process Management cannot monitor the performance of this task and it is very difficult to optimize this process.

Until now we have talked about how the business process life cycle can be controlled and managed from a rather business perspective. As we live in a virtual world dominated by computerized infrastructure, we should focus on executing the business processes and models through information technology (IT). This leads us to *Workflow Systems*.

1.1.3. Workflow Systems

Workflow systems bridge the gap between the real world and the virtual world. In a company, the real world consists of physical goods, employees and other organisations. In contrast, the virtual world is made up of computerized infrastrucure, applications and databases. In order to bridge the gap between these two worlds, workflow systems offer a visible way to model the organization's process design and a possibility to interprete and execute the design [Jon06]. In other words, a workflow system enables the company to execute a designed process immediately without making any efforts to implement new or change existing technical infrastrucure, applications or databases.

It sounds awesome, doesn't it? In fact, such workflow systems exsist for very specific domains but the reality looks a bit different. A top-down evolution of a process change happens as follows. First, executives, business managers or business analysts decide to change some goals and operations. They are supported by computer based management systems offering a wide range of tools to assist the business life cycle at a high level of abstraction. Then, if the company has a well designed workflow system, the workflow model has to be changed to match the new goals and operations to execute the new model. In most cases a workflow system cannot embrace the whole IT infrastrucure of a company. This means that usually you have to set up developer teams to implement

the new process to be able to execute it. These teams develop new necessary applications, change existing implementations, redefine or define new interfaces, data formats, database schemes etc..

Apparently the step from the business process model to the implementation and execution takes a lot of time. Time is the ingredient organisations do not have, though. In order to stay in the competition, organisations have to improve their performance by means of more effective business processes. As a consequence, and due to the growing complexity of the business environment, business processes have to change in shorter intervals. Sometimes processes change again even before the last change is fully implemented.

This paper and especially the next chapter focuses on reducing the time, resources and costs to put through changes of business processes. Now we must have enough insight to understand the problems and to be able to move on to present concepts, technologies and tools to tackle this issue.

2. Concepts, Technologies and Tools

This chapter introduces emerging concepts, technologies and tools promising to support and facilitate the business process life cycle and Business Process Management better than ever before.

This chapter is not meant to compare existing concepts and technologies. A top-down approach is used to present a specific concept that has become a hot topic during the recent years. It starts with the highest level of abstraction of Business Process Management, that is, the modeling of business processes and steps down until the execution level, representing the lowest level of abstraction, is reached. A solution is presented to implement and execute steady changing business processes as quickly as possible.

First, an emerging standard is introduced enabling managers, process owners or process designers in general, to model and change their business processes in a consistent and easy way. Then, specifications and technologies are discussed to define and describe a business process in such a way that it can be immediately deployed and executed on an execution engine. Finally, such execution engines and their features are presented.

The main purpose of this chapter is to discover whether all these concepts, technologies and tools are able to form a chain capable of facilitating Business Process Management or not and to which extent. For example, a dream of every manager is to change a business process and to click on an OK button executing the new process. At the end of this chapter the question, how close the emerging concepts and technologies can bring the managers to this dream will be asked.

Let us start off now with Business Process Modeling Notation. But it is important to keep in mind at any time throughout this chapter that despite talking about business processes, the same concepts and technologies are utilized to manage learning processes in the next part.

2.1. Business Process Modeling Notation

2.1.1. What it is About

Business Process Modeling Notation (BPMN) is an emerging standard for business process modeling. It was developed and put forth by the Business Process Management Initiative[1] (BPMI) and provides a graphical notation that is easy to use and to understand to all business users. Business analysts creating initial process drafts as well as technical developers implementing the technology that will perform these processes,

[1] http://www.bpmi.org/

and managers managing and monitoring the running processes, are able to read the diagrams provided by BPMN. It bridges the gap between technical and non-technical people. Despite of being easy to use, it has the ability to model very complex processes.

BPMN specifies a single diagram, called the Business Process Diagram (BPD). This diagram was developed with web services and Business Process Execution Languages in mind. Thus, BPDs map directly to any major execution language such as, for example, Business Process Execution Language for Web Services (BPEL4WS) and Business Process Modeling Language (BPML). We will describe these execution languages in more detail later. Besides directly mapping to execution languages, BPD visually expresses different execution languages with a common notation.

While reading the last two paragraphs, some readers might have thought of the Unified Modeling Language (UML). At first sight BPMN and UML could look similar but they are not.

2.1.2. Advantages over Unified Modeling Language

According to [OR] BPMN has the following advantages over UML because it offers a process-centric approach rather than an objective-oriented approach offered by UML:

- The process flow model of BPMN is more natural and intuitive for the business analyst to use.
- BPMN is based on a solid mathematical foundation and therefore can be mapped to Business Process Execution Languages in contrast to UML. PI-Calculus is the name of the mathematical background that is a "formal method of computation that forms the foundation for dynamic and mobile processes". "It is analogous to the functionality of relational data models and the generation of SQL statements" [OR].
- BPMN and UML can play together as BPMN can also be mapped to UML.

Despite of some advantages of BPMN, UML has not become obsolete. There is still a need for systems development and UML is a language helping developers visualize, define, specify and document models of software systems.

Now let us take a look at the Business Process Diagram and its elements.

2.1.3. Business Process Diagram

The Business Process Diagram (BPD) is utilized to offer a simple mechanism for creating business process models. On the one hand, it is easy to use and to understand business analysts, managers and technical people. On the other hand, it is also possible to design complex processes and map them to a Business Process Execution Language.

The BPMN Specification [Whi] contributes to this concept and divides the specification of BPD into three parts. Firstly, the core elements giving the basic look and

feel of BPMN. Secondly, an entire list of all elements needed to model complex processes. Thirdly, non-graphical attributes providing information to map to an execution language.

We will not cover the whole BPMN Specification but instead we will give an overview of the main objects being a good starting point to understand Business Process Diagrams.

The four basic categories of elements are:

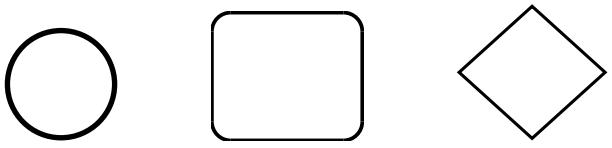
- Flow Objects
- Connecting Objects
- Swimlanes
- Artifacts

There are three flow objects defining the behaviour of a business process [Whi]:

Events: An event is something that happens during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of events, based on when they affect the flow: Start, Intermediate, and End. For the notation look at Figure 2.1(a) on page 9.

Activities: An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a process model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are rounded rectangles. Processes are either unbound or contained within a Pool. For the notation look at Figure 2.1(b) on page 9.

Gateways: A Gateway is used to control the divergence and convergence of Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths. For the notation look at Figure 2.1(c) on page 9.



(a) Event (b) Activity (c) Gateway

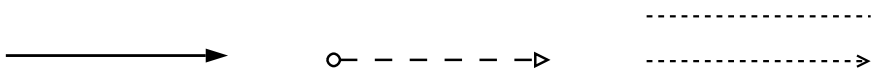
Figure 2.1.: Shows BPMN Flow Objects

Further, there are three ways of connecting Flow Objects to each other called Connecting Objects [Whi]:

Sequence Flow: A Sequence Flow is used to show the order that activities will be performed in a process. You can find the notation in Figure 2.2(a) on page 10.

Message Flow: A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in the diagram will represent the two participants (e.g., business entities or business roles). You can find the notation in Figure 2.2(b) on page 10.

Association: An Association is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects. You can find the notation in Figure 2.2(c) on page 10.



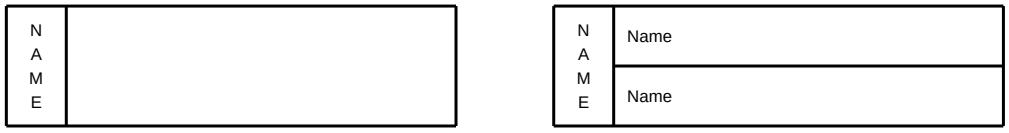
(a) Sequence Flow (b) Message Flow (c) Association

Figure 2.2.: Shows BPMN Connecting Objects

BPMN specifies two ways of grouping modeling elements (e.g. progresses, events and gateways) through so called Swimlanes:

Pool: A Pool (see Figure 2.3(a) on page 10) is used to separate participants in a process or different business entities. Typically, a Pool represents an organisation.

Lane: In contrast, Lanes (see Figure 2.3(b) on page 10) are used to organize, separate and categorize activities within a pool. Both, Pools and Lanes, specify who does what, where events occur, where decisions are made, and who makes them.



(a) Pool (b) Lane

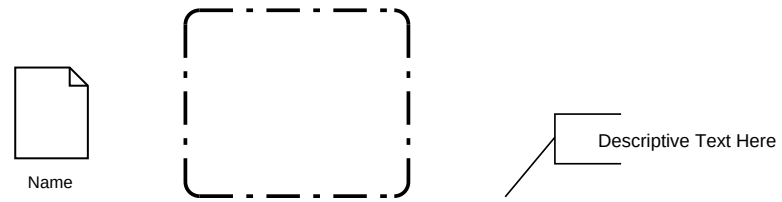
Figure 2.3.: Shows BPMN Grouping Objects

The last category is about Artifacts. It is used to provide additional information about the process. There are three Artifacts specified in BPMN but modeling tools are allowed to add as many new Artifacts as they need [Whi]:

Data Object: Data Objects (see Figure 2.4(a) on page 11) are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the process, but they do provide information about what activities require to be performed and/or what they produce.

Group: Grouping (see Figure 2.4(b) on page 11) of activities does not affect the Sequence Flow. The grouping can be used for documentation or analysis purposes.

Text Annotation: Text Annotations (see Figure 2.4(c) on page 11) are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.



(a) Data Object (b) Group (c) Text Annotation

Figure 2.4.: Shows BPMN Artifacts

This was a brief overview of the core elements of the Business Process Diagram specified by BPMN. If you want to know more about it, read the BPMN Specification itself [Whi] or *BPMN and Business Process Management* [OR] for a very good introduction.

2.1.4. Business Process Diagram Example

After this short introduction of the Business Process Diagram and its elements and objects, we must be able to understand the example depicted in Figure 2.5 on page 12. This BPD describes an auction. As you can see, it consists of two pools specifying who does what. In this particular case we are talking about a bidder and a seller. If the bidder finds an item of interest, the whole process starts.

First, the bidder reads the item description. If the provided information about the item is sufficient, he or she can proceed to make a choice whether to deal with the seller. If the information is not satisfying, the bidder contacts the seller for further information. Also, he or she can review seller's credentials. However, all actions end up with the final question whether to make a deal with the seller or not. If the bidder does not want to, the process is over and no bid is made. In case the bidder wants to buy the item, a message is sent to the seller letting him or her know that a specific item is the object of interest. Then, the seller verifies bidder's credentials and closes the auction. The process ends with the end of the auction and a sold item that is subject to payment.

Of course it is a very simplified auction. Butit gives a gist how easy it is to read a Business Process Diagram and how easy it must be to draw it.

2.1.5. Software Implementations

There is no point in using a standard if there are not enough software applications implementing it. It is obvious that BPMN would be useless if no vendor implemented it.

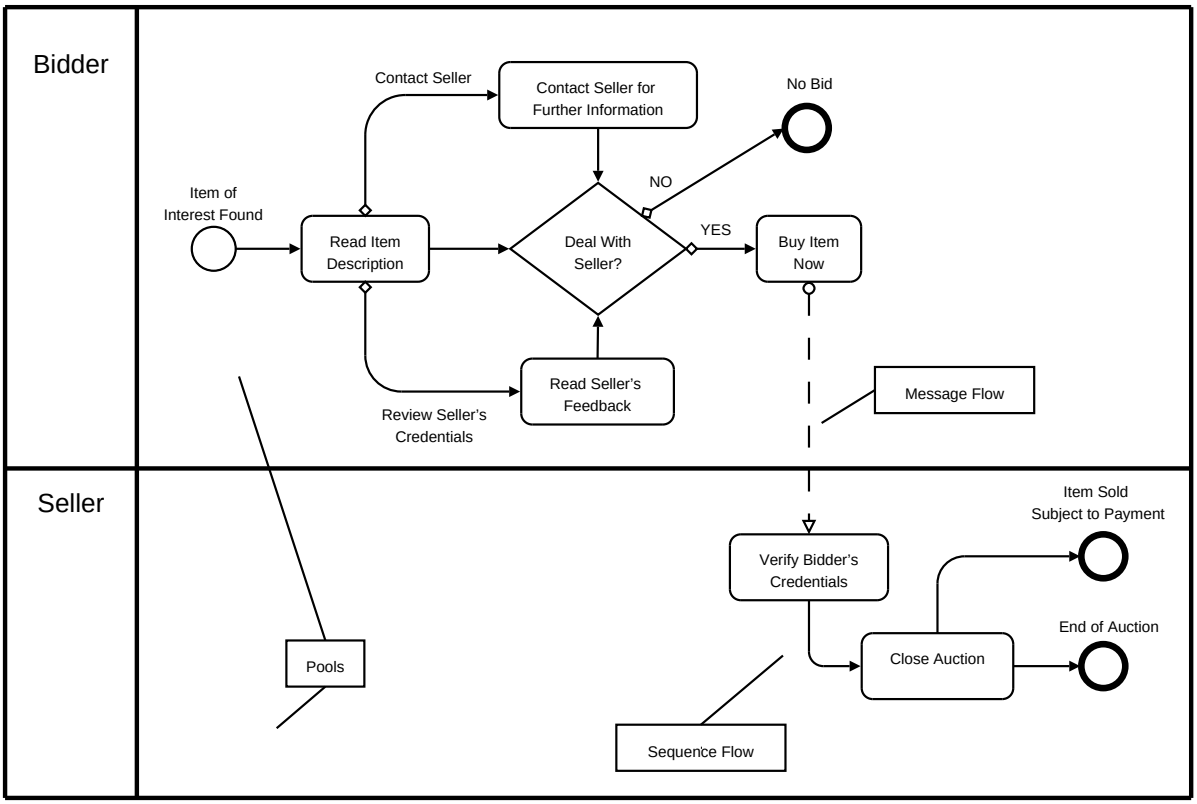


Figure 2.5.: Shows a Business Process Diagram with pools

For the time being the Object Management Group (OMG)[2], which is an international, open membership, not-for-profit computer industry consortium, maintains a web page with BPMN information[3].

At the moment 37 BPMN implementations are listed and four are planned. In the list[4] there are several well-known vendors with their products such as BEA with Fuego 5, Fujitsu with Interstage Business Process Manager 7.1, Italio with n3 Designer, Corel with iGrafx, and Sun Microsystems with Studio Enterprise Edition, just to mention some.

As we can see, there is no need to worry about software being available but at the time of writing only proprietary products and no decent free or open source applications were found.

[2]http://www.omg.org/
[3]http://www.bpmn.org/
[4]http://www.bpmn.org/BPMN_Supporters.htm

2.1.6. Next Steps

What are the next steps to get from the Business Process Diagram (BPD) to a running process? Let us assume a business analyst has designed a new process using BPMN and BPD by means of a software product. According to [OR] it is a four-stage process to make web services work. Web services are one of the concepts to facilitate the process management and process life cycle, and will be introduced later in this paper. For now, all we must know about web services is that, when they work, the business process is implemented and running.

The four-stage process is as follows [OR]:

1. Design the processes using BPMN. This step has already been done.
2. Simulate the processes and modify them for efficiency. Usually BPMN suites and Business Process Management Systems (BPMSs) support simulating and analyzing the process before their realization. During this simulation assumptions such as time requirements for each activity and resource requirements for each task have to be made. A simulation helps identify throughput, transaction times, bottlenecks, and resources required to fulfil specific tasks in decent time to optimize the process[Nai04].
3. Make the services (e.g. our modeled processes) available by publishing them using a Business Process Execution Language.
4. Orchestrate the web services by assembling them and coordinating their behaviour by means of Business Process Management Systems.

The next section will be covering point two and three. In other words, we will see what has to be done to implement and execute the process drawn by means of BPD. Before beginning the next section, we will sum up.

2.1.7. Conclusion

We have introduced Business Process Modeling Notation (BPMN) that is a standard for business process modeling. It specifies a graphical notation which is easy to use and to understand for all business users including business analysts, managers, and technical staff. It is also possible to model very complex processes although its aim is to be easy and clear to grasp. Moreover, it is capable of mapping the graphical presentation to business execution languages.

Some advantages of BPMN over UML have been discussed. Apparently BPMN suits non-technical staff better than UML but still can be used by technical staff too. This is due to the process-centric approach offered by BPMN.

We have described the basic elements of the Business Process Diagram (BPD) and it has been shown that a BPD is easy to read and use, and there is no need for a technical expert.

However, it seems to be easy to model and change processes through BPMN and BPDs but how is it possible to implement and execute these processes easily. As mentioned in the preceding section, we will take a look at the next steps in this management chain and move on to Web services and the Business Process Execution Language to find out how to implement and execute our modeled processes eventually.

2.2. Web Services and Technologies

Before moving on to implementing business processes designed by means of BPMN, some time has to be invested introducing particular concepts and technologies related to Business Process Execution Languages. This is necessary to fully understand the subsequent sections.

First, this section introduces the concepts of Service Oriented Architecture (SOA) and Web Services. As the web service approach is based on several standards, they are briefly described. Thus, Simple Object Access Protocol (SOAP) being a protocol for exchanging information, Web Service Description Language (WSDL) describing web services, and security related technologies will end this section.

2.2.1. Service Oriented Architecture and Web Services

Service Oriented Architecture (SOA) is an Information Technology (IT) approach that makes use of services on a network[Ort05]. Today the used network is very often the World Wide Web or the Local Area Network (LANs) in companies. The approach involves the usage of services and creation of new services.

In general a service provides a specific function. It can be distinguished between a single discrete function and a set of functions[Ort05]. The former performs a simple operation like converting currency whereas the latter offers a greater service such as, for example, booking a flight.

The SOA approach is to connect such single services which might be exposed again as another more complex service. To put it differently, this approach can be thought of as sharing functions between applications or even whole applications in a very flexible way. For example, somebody could write a function converting currencies and expose it through a network. You could write an application calculating the costs traveling the world and because you would probably need a function to convert currencies, the already existing function could be used. In the same way someone else could exploit your application if you make it available. This is exactly the idea of an SOA.

This approach is agile in responding to changing business needs. It speeds up the application development process and systems become more adaptable. Furthermore, the client, which communicates with the services, is independent of the services itself. It does not have to know about the platform, the services run on, or the programming language they are written in. The client simply does not need to care about. For example, very often the client sends a document to the service and gets a result back

based on the content. This service is called document-oriented service as it performs an operation on a document.

The client must communicate through well-defined interfaces. This is the point where web-based SOA comes in.

Web-based SOA is SOA by means of web services. Web services communicate through a set of standard protocols and technologies supported by major software vendors. This ensures a consistent communication between clients and services over many different platforms.

Now these protocols and technologies will be described in the following sections with the focus on web services.

2.2.2. Extensible Markup Language

Extensible Markup Language (XML) is a simple and very flexible text format derived from SGML[W3Ca]. Over the years it has become a de facto standard for describing data to be exchanged on the Web and elsewhere. It was generally adopted to be the data exchange format for web services.

Tags describe the content of the document that has a well-formed structure. According to the XML Specification[BPSM+04], a well-formed XML document, for example, has ending tags and needs to be completely nested not like the HyperText Markup Language(HTML).

Listing 2.1 on page 15 shows a very simple XML document describing the content of a bookshelf.

Listing 2.1: A very simple XML document describing the content of a bookshelf

```
1  <bookshelf>
2   <book>
3    <title>1984</title>
4    <author>George Orwell</author>
5   </book>
6   <book>
7    <title>Brave New World</title>
8    <author>Aldous Huxley</author>
9   </book>
10  <book>
11   <title>The Hitchhiker's Guide to the Galaxy</title>
12   <author>Douglas Adams</author>
13  </book>
14 </bookshelf>
```

However, it is up to the reader to do further readings about XML if he or she is not familiar with this Markup Language. This paper is not meant to give an introduction to XML basically due to the fact that it would double the size of this paper and that it is very likely that you already know enough about XML.

2.2.3. Simple Object Access Protocol

Simple Object Access Protocol (SOAP)[W3Cb] is an XML-based protocol for exchanging data. As already mentioned, XML is used for web services as the data exchange format. It is not enough to define how the data is structured. The receiver has to understand how it is transmitted and what the main part of the message is.

The concept of SOAP is that any two computers can communicate through a high level protocol such as HTTP, SMTP (Simple Main Transfer Protocol) or POP3 (Post Office Protocol 3) to transmit information by means of XML. HTTP is by far the most commonly used protocol not only because it can pass through firewalls. Both, XML and HTTP are open standards, wide spread and many freely tools are available to work with them.

The SOAP communication is based on SOAP messages that consist of three parts, the SOAP envelope (mandatory), SOAP header (optional) and the payload (mandatory). The envelope defines the XML Namespace[5] avoiding name clashes and the encoding style of data types. The optional header can extend the SOAP message in a modular way and is typically used to transmit security related information. Last but not least, the payload is the actual message to be sent.

Listing 2.2 on page 16 is an example of how a SOAP message might look like when being sent to a warehouse web service. The client requests details about the product with ID 12345. The SOAP message could be sent through HTTP to the server where the specific web service runs.

Listing 2.2: Example of a SOAP message requesting product details

```
1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
    envelope/">
2  <soap:Body>
3   <getProductDetails xmlns="http://warehouse.example.com/ws
      ">
4    <productID>12345</productID>
5   </getProductDetails>
6  </soap:Body>
7 </soap:Envelope>
```

The message shown in Listing 2.3 on page 16 could be replied by the warehouse's web service. It provides the product name, ID, description, price and the information whether it is in stock or not.

Listing 2.3: Example of a SOAP message replying details about a product

```
1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
    envelope/">
2  <soap:Body>
```

[5] http://www.w3.org/TR/REC-xml-names/

```
3    <getProductDetailsResponse xmlns="http://warehouse.example
        .com/ws">
4     <getProductDetailsResult>
5      <productName>RED SATIN PROM DRESS</productName>
6      <productID>12345</productID>
7      <description>Bright Red Satin Dress</description>
8      <price>75</price>
9      <inStock>true</inStock>
10    </getProductDetailsResult>
11   </getProductDetailsResponse>
12  </soap:Body>
13 </soap:Envelope>
```

Now we know how to communicate and transfer data between clients and web services but how does a client know what functions a web service provides? The Web Service Description Language is the answer.

2.2.4. Web Service Description Language

Web Service Description Language (WSDL)[W3Cc] is an XML format for describing web services or more generally speaking network services. It describes in an abstract way where to find the services, what operation it exposes and how its messages look like. To put it differently, a WSDL definition of a service can be thought of as an interface telling the client how to use it. Usually when a new service is exposed, a WSDL file is written and a pointer to this file is registered in a public service directory. This enables users to find the WSDL file and use the new service.

It is probably the best way to start with an example to get an insight into WSDL. Listing 2.4 on page 18 shows a simplified but suitable WSDL file for our warehouse example. It almost consists of all major WSDL elements. Let us describe these elements with respect to our example:

message: It defines the data elements of an operation. As it can be seen, two message elements are needed describing the message format for the request of the client and for the response of the web service. When the client contacts the warehouse's web service to ask for details about a particular product, it only has to send the product ID that is an integer. In contrast, the response message of the web service consists of more information, that is, the product's name, ID, description, price and whether it is available.

portType: It describes a web service in terms of the messages that are involved and the valid operations that can be performed. In our example we have only one operation defined, *getProductDetails* (line 14). It needs an input and output message to operate which are defined in line 15 and 16 and refer to the messages previously defined. The operation type here, which takes an input and output message,

is called Request-response type but there are also other types like One-way or Notification for example. However, these elements describe the web service in a rather abstract way.

binding: This element defines the message format and protocol details for the web service and therefore has to leave the abstract way of description. In our case the binding is for the specific web service called *productDetails* (line 20). Furthermore, the style attribute states that complete documents (typically XML documents) will be exchanged. In addition, the transport attribute (line 22) makes sure that HTTP as a transfer protocol will be used. Next, in line 24 and 25 the Uniform Resource Locator (URL) points to the location of the web service. Finally, the remaining lines specify how the input and output are decoded. In this case it is *literal*.

Now we know exactly where to find the web service and which protocol has to be used.

Listing 2.4: Example of a WSDL file describing the warehouse web service

```
1  <definitions>
2   <message name="getProductDetailsRequest">
3    <part name="id" type="xs:integer"/>
4   </message>
5   <message name="getProductDetailsResponse">
6    <part name="name" type="xs:string"/>
7    <part name="id" type="xs:integer"/>
8    <part name="description" type="xs:string"/>
9    <part name="price" type="xs:float"/>
10   <part name="stock" type="xs:boolean"/>
11  </message>
12
13  <portType name="productDetails">
14   <operation name="getProductDetails">
15    <input message="getProductDetailsRequest"/>
16    <output message="getProductDetailsResponse"/>
17   </operation>
18  </portType>
19
20  <binding type="productDetails" name="b1">
21   <soap:binding style="document"
22    transport="http://schemas.xmlsoap.org/soap/http" />
23   <operation>
24    <soap:operation
25     soapAction="http://warehouse.example.com/
          getProductDetails"/>
```

```
26      <input>
27       <soap:body use="literal"/>
28      </input>
29      <output>
30       <soap:body use="literal"/>
31      </output>
32     </operation>
33    </binding>
34  </definitions>
```

After this example you must be able to read and understand almost all WSDL files. Do not bother yourself too much with WSDL, though, as there are many tools helping to deal with it in terms of generating and validating WSDL files.

So much for WSDL, but let us talk now a bit about security for web services.

2.2.5. Security

There are two emerging standards of special interest. We are taking a very brief look at Web Service Security (WSS) and Security Markup Assertion Language (SAML).

Web Service Security

Web Service Security (WSS) is a standard released by OASIS[6] describing enhancements to SOAP messaging. Basically it provides protection by means of message integrity and confidentiality[OAS]. Integrity means that a SOAP message is not tampered while being on the way to its final destination. Confidentiality means that a SOAP message is only seen by entitled recipients.

WSS is very flexible and can be used with several security models and cryptographic technologies. It uses security tokens and digital signatures to apply security to web services. It is quite flexible because no specific type of security token is specified by WSS and a variety of security models such as Public-Key Infrastructure (PKI), Kerberos[7], Secure Socket Layer (SSL), and Transport Layer Security (TSL) can be used.

However, WSS describes enhancements to SOAP messaging and embeds additional security information into the SOAP message. As it was mentioned before, the SOAP header is used to carry this kind of information. If you want to see how an example SOAP message with WSS looks like, check out Listing A.1 on page 100. It was taken over from the WSS Specification[OAS04] and everything that is related to WSS is between line 10 and line 37.

[6] http://www.oasis-open.org/
[7] http://web.mit.edu/Kerberos/

Security Assertion Markup Language

Security Assertion Markup Language (SAML) was developed by the OASIS Security Services Technical Committee in March 2004. It is "an XML-based framework for exchanging security information. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain" [Cov]. In other words, SAML is an XML standard for exchanging authentication and authorization data between identity providers and service providers.

So called Assertions are transfered between providers. They contain statements that are used by service providers to make decisions on whether an entity is permitted to access a particular service. This concept also enables single sign-on (SSO), which makes it possible to sign in to multiple services (e.g. typically web services) that are scattered in a distributed environment.

SAML can be bound to different communication and transport protocols and the request and response message format is XML-based. For the moment, SAML defines one binding, that is, to SOAP over HTTP[Cov]. Hence it follows that it can be used with the concepts and technologies we are describing in this paper.

2.2.6. Conclusion

We have seen what Service Oriented Architecture is all about and that it is agile in responding to changing businesses. Especially web-based services sharing functions through a network are the concepts we can apply to tackle the issues of steady changing business processes.

Furthermore, several specifications have been introduced with focus on web services. First, we discussed XML utilized to structure and to describe data to be exchanged. Then, SOAP was introduced which specifies the communication of information (e.g. XML document) through a high level protocol that is typically HTTP. Next, WSDL was presented which is used to describe web services in terms of what operations they can perform, how messages look like, and where to find web services. Finally, two security-related emerging standards were shown.

However, some of you might still be asking how web services can be utilized to implement and execute designed business processes through BPMN. When we look back at the process example of an auction (look at Figure 2.5 on page 12), it is pretty obvious that that Process Diagram alone cannot just be executed. Implementations have to be done somewhere.

The approach of web-based SOA could be used to speed up the development process and to respond to changing business needs quickly. Taking our auction as an example, every activity could be implemented by means of a web service. For instance, the first activity, called *Read Item Description*, could be implemented as a web service. It might get an item ID from the client and then would respond with the item's description. All other activities could be implemented as web services likewise.

The whole process (e.g. the auction) would benefit from this architecture. As every activity is represented by one web service, you do not have to change the services itself if you change the sequence of the process or if you add new activities. In addition, if one activity has to be changed, all other web service implementations can remain untouched. Also, all web services are independent and nobody has to care about which programming language they are written in or which platform they are running on. All these points come in very handy when we have to respond to a process change quickly.

Now there is only one question left. How can we coordinate web services to reflect the designed business process? This is exactly the job of Business Process Execution Languages that are described next.

2.3. Business Process Execution Language

Business Process Execution Languages (BPELs) are XML-based meta-languages used for modeling business processes in terms of orchestrating, running and controlling web services. Most of them are built on the top of WSDL and add a good orchestration layer to SOA. BPELs are the languages for process management, just as XML is the language for structuring and exchanging data, and HTML is the language for hyperlinked Web pages.

The most important characteristic of BPELs is that it can be directly executed on an IT infrastructure. BPEL code is the code that is executed by a process virtual machine, called Business Process Execution Engine. It can be thought of as the way a Java program is executed by the Java Virtual Machine[SF03]. This means that you can take the same business process modeled in BPEL and execute it on any operating system where an execution engine exists. BPELs define only what is required to describe and model a business process. No details about the environment, and system, the processes run on, are defined. BPELs describe and manage business processes in a rather abstract way.

In other words, we can say that BPMLs provide a vocabulary for interchanging business process definitions across heterogeneous systems and modeling tools[SF03]. This is important to companies as they want to use the best-of-breed components and tools to manage their processes. Nobody wants to stick to a home grown solution when it comes to such a complex and vital topic, as Business Process Management is.

The most BPEL standards provide specifications for[OR]:

- Dataflow
- Messages
- Business Rules
- Events
- Exceptions

- Transactions

However, in our context BPELs describe business processes by means of orchestrating web services that can be deployed and executed on any system providing an execution engine. Actually, this is the goal of our journey. After all we want to execute the business process modeled through BPMN and expose the whole process as an own web service if needed.

In the following, the history of BPELs will be looked at as there is not only one BPEL specification as you might image. Then, we will give an overview of the Web Service Business Process Execution Language (WSBPEL) and its main elements of the grammar. Finally, some BPEL Designers and Execution Engines will be mentioned.

Let us move on to the history of BPEL now.

2.3.1. History and Development

Figure 2.6 on page 23 gives an overview of competing BPML standards and their development. Business Process Execution Language for Web Service (BPEL4WS) grew out of language features of IBM's Web Service Flow Language (WSFL), which is an XML language for the description of Web Services compositions [Ley], and Microsoft's XLANG, which is a notation for the specification of message exchange behavior among participating web services[Tha]. Both WSFL and XLANG are superseded by the BPEL4WS specification.

IBM[8], Microsoft, BEA Systems[9], SAP[10], and others had formed a joint venture to push BPEL4WS (or very often just called BPEL) forward. The biggest rival was Business Process Management Initiative's (BPMIs) Business Process Modeling Language (BPML) that is just one standard among two other BPMI's specifications. Business Process Modeling Notation (BPMN) and Business Process Query Language (BPQL) are the other two. As mentioned earlier in this paper, BPMN builds on a mathematical foundation and it is the same for the other two specifications.

However, in April 2003 BPEL4WS 1.1 and BPML were submitted to the OASIS Technical Committee[11] to create one standard. OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards.[12] The outcome of the committee was a draft for a new standard called Web Service Business Process Execution Language 2.0 (WSBPEL)[AAA+]. The name WSBPEL and the version starting with 2.0 is meant to make clear that it is a new standard and not BPEL4WS anymore. At the time of writing the standardisation process is still underway to elevate the existing draft to the final specification.

[8] http://www.ibm.com/
[9] http://www.bea.com/
[10] http://www.sap.com/
[11] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office
[12] http://www.oasis-open.org/who/

There is already software implementing WSBPEL but since it is not a final specification yet many tools still support BPEL4WS 1.1 and also BPML.

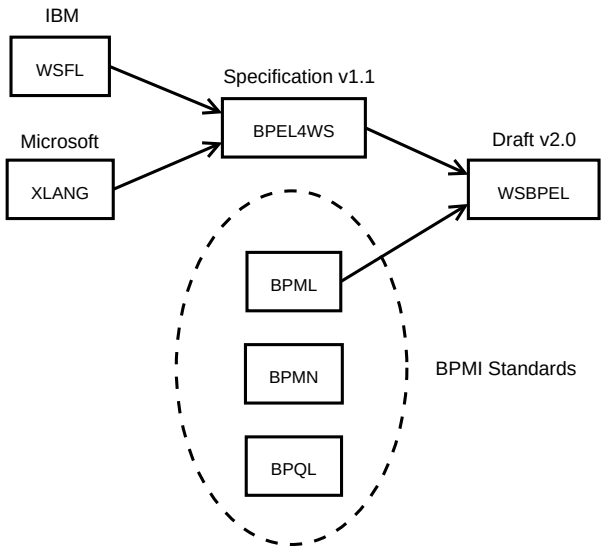


Figure 2.6.: Shows the history and development of competing BPML standards

Now let us focus on the WSBPEL specification.

2.3.2. Web Service Business Process Execution Language

As most BPELs, WSBPEL is used to define and execute business processes. Especially WSBPEL enables the realization of SOA through composition, orchestration and coordination of web services. Basically it offers elements to invoke web services in a specific order, either in a sequence or parallel, and to express conditional behaviour.

Web services can be combined in two ways:

- Orchestration
- Choreography

Figure 2.7 on page 24 and Figure 2.8 on page 24 depict the difference between orchestration and choreography[Jur]. Using orchestration, a typical scenario could look as follows. A coordinator receives a request from a client web service (i.e. web service A). Usually such a request asks to do a specific task, for instance, to book a flight, ask for a loan, etc.. In our example the coordinator invokes web service B, C and D respectively in order to fulfil the tasks. Finally, the client gets a reply when the tasks are accomplished. All involved web services are controlled by a central process (coordinator) and they do not have to be aware of that they are taking part in a higher-level business process.

In contrast, using choreography, the conduction of the same tasks would look different. A web service (in our example web service B) receives a request from a client web service (i.e. web service A) asking for a task that has to be done. This web service contributes with its part of the job and knows which web service to invoke next to get the remaining parts of the whole task done. Thus, in our example web service B invokes service C. Service C does it likewise and finally invokes service D which replies to the client web service after finishing its part and with it the whole task. The main difference to the approach of orchestration is that all involved web services must be aware that they are taking part in a higher-level business process.

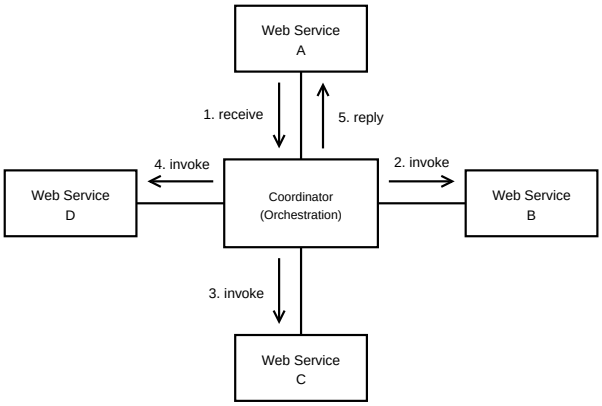


Figure 2.7.: Shows the composition of web services through orchestration

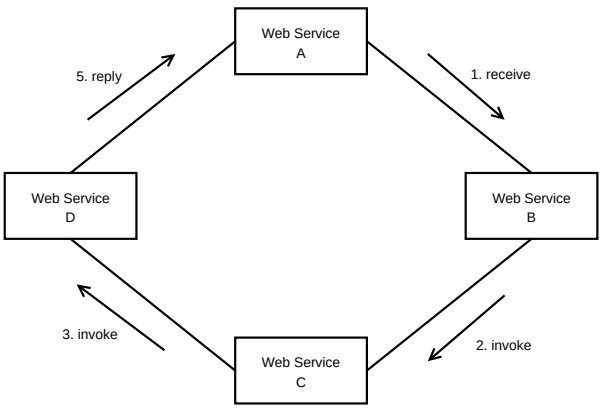


Figure 2.8.: Shows the composition of web services through choreography

The main advantages of orchestration over choreography are that it is centrally managed by a known coordinator, that web services do not have to be aware that they are part of something, and last but very essential, that in case of an error during the in-

vocation chain, the coordinator can do appropriate error handling or even try to find another solution.

Despite some disadvantages of choreography, both, orchestration and choreography are supported by WSBPEL. This is necessary because real life processes can be divided into private and public processes.

Private processes (also called executable processes) are processes within an organisation. They are coordinated in a centralized way as it is exactly known what these processes are supposed to do and how they actually do it. Thus, they can be executed within the organisation. This kind of processes are combined through orchestration.

On the other hand there are public processes. Those are services crossing several organisations. Each organisation knows about its own job of the whole processes but has no clue how the other parties do their part of the job. The only things they know from each other are the incoming and outgoing messages and what the goals are but not how the other parties do it. As of this nature, these kind of processes are combined through choreography.

This situation can also be compared with an abstract class. You know its methods, their parameters (incoming messages) and what they return (outgoing messages) but you do not know how they are implemented. That is why these processes are also called abstract processes.

However, the support of orchestration and choreography makes sure that all real-life business processes can be managed through WSBPEL. After these considerations let us take a look at what elements a WSBPEL file consists of.

WSBPEL Elements

A WSBPEL file consists of the basic structure shown in Listing 2.5 on page 25. *Process* is the root element and besides the process name (e.g. FlightBooking) it holds the elements *partnerLinks*, *variables* and *sequence*. Now we will describe all three elements in more detail.

Listing 2.5: Basic structure of a WSBPEL file

```
1  <process name="FlightBooking" ... >
2     <partnerLinks>
3        <!-- The declaration of partner links -->
4     </partnerLinks>
5
6     <variables>
7        <!-- The declaration of variables -->
8     </variables>
9
10    <sequence>
11       <!-- The definition of the BPEL business process -->
12    </sequence>
13 </process>
```

The *sequence* elements define the actual business process. All parts in this element are called activities. There are basic activities, which do something, and structured activities that organize the basic activities without actually doing anything.

Listing 2.6: An example of some basic activities

```
1  <sequence>
2   <receive createInstance="yes" name="startProcess_receive"
       operation="start" partnerLink="PManagerStarter" portType
       ="ProcessManagerService" variable="start" />
3   <invoke inputVariable="start" name="startProcess_invoke"
       operation="start" outputVariable="started" partnerLink="
       PManagerInvoker" portType="ProcessManagerService" />
4   <reply name="startProcess_reply" operation="start"
       partnerLink="PManagerStarter" portType="
       ProcessManagerService" variable="started">
5  ...
6  <!-- assign activity example -->
7  <assign>
8   <copy>
9   <from variable="receive" />
10  <to variable="result" />
11  </copy>
12 </assign>
```

Listing 2.6 on page 26 is a typical example for the beginning of a business process using basic activities. It creates a new instance of the process when a client contacts it, invokes a new web service (called ProcessManagerService), and finally sends a message to it. You will encounter these lines again when you proceed to the second big chapter of this paper. The following are basic activities:

receive: This activity waits until a client invokes the business process by sending a message. It is also typically used to create an instance of the business process when a client invokes it for the first time. In this case the createInstance attribute is set to yes (see Listing 2.6 on page 26 line 2). The attribute partnerLink, which refers to the partnerLinks element in the WSBPEL file, operation, and portType are WSDL related. It was discussed earlier that these elements in WSDL define which operation the web service should perform and what protocols the communication is bound to.

There is also the activity's name and a variable defined where the received data is stored.

invoke: Invoke invokes another web service. Besides the WSDL related statements,

there are two variables defined that contain the input and output data of the invoked web service.

reply: This activity sends a synchronous message stored in a variable.

assign: It is used to manipulate data variables. For instance you can copy the value of one variable to another variable (see Listing 2.6 on page 26 line 7).

throw: This activity indicates exceptions.

wait: It waits a given time before the process resumes.

terminate: This activity is used to terminate the whole process.

Structured activities can be compared with control flow statements in programming languages such as Java and C++ for instance. The following are the most important structured activities in WSBPEL:

- switch
- while
- pick - to select one of several possible paths
- flow - to define a set of activities to be invoked in parallel
- sequence - to define a set of activities to be invoked in an ordered sequence

The elements left are *partnerLinks* and variables. The former defines all different parties interacting with the business process. Basically it characterizes partner links and makes the link to the WSDL files describing involved web services. This mechanism makes sure that the process knows about the role of each partner and how to communicate with it.

The element *variables* defines all used variables throughout the process description. As we have already seen, variables are usually used for every message sent to the involved parties and received from them. They are utilized to handle the process execution data. A typical definition of a variable could look like this:

```
<variable messageType="productData" name="product" />
```

The name of the variable is product and the messageType refers to the message type specified in the WSDL file.

We have seen the main elements of the WSBPEL grammar and must be able to understand any normal business process defined throughout WSBPEL. Since there are several necessary steps to get a business process ready to execute, let us recap what we have to do.

Sum Up

The following steps are necessary to implement a business process:

1. Implement all necessary web services.
2. Define the WSDL for all involved web services.
3. Develop the WSBPEL process by defining partner links, variables, and the process logic.

In addition, Figure 2.9 on page 28 [Moo06] shows the relation between WSBPEL and WSDL as discussed before.

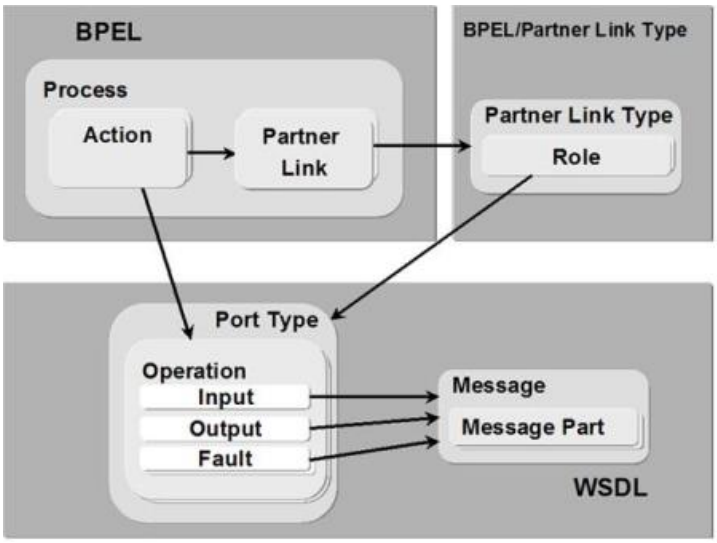


Figure 2.9.: Shows the relation between WSBPEL and WSDL

2.3.3. Business Process Execution Language Designers

It is not the only way to use an ordinary text editor for writing BPEL files. As for many XML-based specifications, there are also several graphical tools for BPEL that facilitate the work. Later in this paper (look at Section 3.5) we will be observing three BPEL Designers.

Despite these very handy tools, it is hardly possible to write a BPEL file without knowing a bit about XML and BPEL, and tweaking the native BPEL code.

2.3.4. Business Process Execution Engine

A Business Process Execution Engine (BPEE) executes, coordinates and monitors business processes. It can be thought of as the middle layer between the business process definition (i.e. BPEL file) and the actual operating system. It is the BPEE's job to expose the deployed business process to the clients, to execute them when requested, and to coordinate and monitor all running instances of all processes.

Thus, a BPEE has to cope with a lot of issues. For example, it has to provide synchronous and asynchronous message exchange patterns, flow coordination, data manipulation, and exception handling for many running instances of processes in a specific system environment.

Most of the existing BPEEs have got the following typical features:

- Native support for a specific BPEL (e.g. WSBPEL, BPEL4WS etc.)
- Support for web service standards
- Able to run on multiple platforms
- High performance and scalability
- Web-based interfaces for administration, management, monitoring and debugging

Besides the mentioned features, some products come with a very sophisticated monitoring extension. This enables the user to measure process metrics in real time. This is vital for assessing the performance of running business processes. Furthermore, the data being monitored and captured can be used to build statistics and send notifications by means of emails, texts, voice notes etc. in case of critical situations (usually based on threshold violations). Last but not least, this data can be used to optimize the business processes. At this point it might ring a bell. Right now we have closed the loop of the business process life cycle (see Figure 1.2 on page 4. It means that a new life cycle begins starting to model an improved business process.

Anyways, there are a lot of open source and proprietary implementations of BPEEs. The following list mentions just a few of them:

ActiveBPEL Engine: A very comprehensive BPEE from Active Endpoints[13]. It is open-source and written in Java.

Apache Ode: Also an open-source implementation of a BPEE that is being developed by the Apache Software Foundation[14].

FuegoBPM: It is a very well-known commercial product developed by BEA that is part of the Business Process Management suite[15].

[13] http://www.activebpel.org/
[14] http://incubator.apache.org/ode/
[15] http://www.fuego.com/

Italio: It is also a well-known implementation[16]. Its BPMS components already support WSBPEL 2.0.

Oracle BPEL Process Manager: This is also a kind of BPMS suite[17] for creating, deploying, running and managing business processes.

Twister: At the moment it seems to be the first open-source implementation of the new WSBPEL emerging standard. Now it is part of Agila BPEL[18].

As you have probably noticed, most BPEEs are part of a bigger Business Process Management suite. However, this is just a small list but provides a good starting point if somebody is looking for a BPEE to run their business processes.

So much for BPEEs but we will hear about the ActiveBPEL engine later in this paper because it will be used to run learning processes.

2.4. Conclusion

We have seen emerging concepts, technologies and tools being able to support, facilitate, and to speed up the business process life cycle. This topic was looked into because it had been found that it is vital for companies to respond quickly to changing business conditions. The only way to stay ahead of the competition is to improve and change their business processes as quickly as business conditions change. Therefore, we started looking for concepts, technologies and tools that can facilitate the Business Process Management supporting the whole business process life cycle.

The journey started at the beginning of the life cycle and discussed concepts and technologies for each step of it until the end was reached. At this point we should recall the five stages of the life cycle, that are, Model, Implement, Execute, Analysis, and Optimize.

The Business Process Modeling Notation (BPMN) was introduced as a standard specification for business process modeling that is able to support the *Model* phase. It is a graphical notation that is easy to use and to understand for all business users as well as all technical users.

Then, concepts and technologies were discussed to implement the modeled business process (*Implement* phase). It was showed that the approach of Service Oriented Architecture (SOA) and web services suits well for steady changing business models and surely plays an important role in business as it is turning into a virtual world made up of computerized infrastructure.

Furthermore, some technologies and specifications were discussed that built the basis for the success of web services and SOA. We discussed XML, SOAP, WSDL, and security related specifications. In addition, Business Process Execution Languages were

[16]http://www.intalio.com/products/server/
[17]http://www.oracle.com/technology/products/ias/bpel/index.html
[18]http://wiki.apache.org/agila/

introduced that are XML-based meta-languages used for modeling business processes in terms of orchestrating, running and controlling web services. Such modeled processes are capable of being run by execution engines.

Finally, we discussed Business Process Execution Engines (BPEEs) that are able to support the phases Execution and Analysis of the Business process life cycle. It was also mentioned that most execution engines are part of bigger Business Process Management System suites that can use the provided data, collected by the execution engine, to help optimize the business process. This covers the last phase (i.e. Optimize) of the business process life cycle.

Supporting concepts and technologies were shown for each phase but how well do they really fit together? To which extent do they support and facilitate the Business Process Management and will the dream of all managers come true to get running processes by just clicking on an OK button?

It is definitely not as easy as pushing a button. Especially the step from the very abstract model to the actual implementation is a critical point. As mentioned, a Business Process Diagram, for instance, can be turned into a skeleton alike Business Process Execution Language (BPEL) file. Technical experts are still needed to complete the BPEL file, to implement web services, and to write the WSDL files. In other words, it is difficult to synchronize and maintain BPMN and BPEL files. In addition, some BPEL specifications are strongly criticized for the absence of human participant interaction as everything is focused on web services.

However, once a process is defined through BPEL and ready for deployment, it is really only one click away from a fully functional process. In fact, when a process must be changed and the changes are within the bounds in terms of just changing the process flow or modifying some parameters, it is easy to put through the changes quickly. It is due to the fact that the step from the new business model to the implementation is not difficult.

In conclusion, it can be said that these concepts and technologies help automate and facilitate the Business Process Management better than the technologies did in the past. Technical experts are still needed to assist the implementation of changes, though. If business processes keep within the bounds of specific domains, changes can be realised and put through more automated procedures without interactions of technical experts. As a rule of thumb, it can be said that the smaller the domain, and therefore the possibilities of a business process are, the more automated the business process life cycle can be. More automation means quicker process changes and less costs for the company.

Let us end this chapter about business processes now and move on to web-based education to figure out whether these concepts, technologies and tools can be utilized to support education as well.

3. Web-based Education

This is the second big part of this paper and is completely dedicated to web-based education.

At the beginning of this chapter this topic is motivated. Questions, like why we need it and what role plays web-based education at the time and in the near future, are answered.

Further, recent applications of web-based education are shown. In addition to several implementations of web-based education systems from all over the world, the WBT-Master is described in more detail. It is a web-based education system running at Graz University of Technology. Its architecture, concepts, tools and typical scenarios are described.

In addiation, learning processes are examined to understand the difficulties web-based education systems have to deal with. We look at the nature of learning processes, their life cycle, and a typical example for such a process.

Being aware of all the problems that emerge with learning processes, learning process management is introduced. Several standardizations are shown, which have been emerged during the recent years, and are able to take the process oriented nature of learning tasks into account. Also, some time is invested to look at a specific implementation, called *Learning Process Manager*, that manages learning processes through Business Process Management and is meant to support web-based education systems (in our case the WBT-Master).

Finally, existing *Business Process Execution Language Designers* are examined to find out whether such tools can facilitate the use of the Learning Process Manager.

Let us start in the beginning. Why for god's sake do we need web-based education?

3.1. Why Web-based Education?

Why web-based education? This question might rise when starting this chapter. We will not be covering this topic in depth but it is easier to proceed with the rather technical problem when we have some motivation.

3.1.1. The Role of Web-based Education

Web-based education, web-based training and e-learning, describing more or less the same, are playing an increasingly important role. This is due to many advantages of web-based education which can cope with our fast changing environment. Whether private

or in business we have to face fast change. This means we have to learn quickly and continuously, which we cannot accomplish without flexibility. We need to be independent of the place and time where we consume education to be flexible. Also [Ste99] pointed out that the key to web-based education is *distance learning* as there is a need for distance between the student and the teacher, and a need for independence.

3.1.2. Distance Learning

The idea of distance learning is not new. We have used work books for centuries but in our fast changing environment they lack cost-efficiency and fast content change. We all know how much books cost and how long it takes to get a revised book taking new developments into account. Moreover, work books as a self-paced studying material, need high self-discipline and highly motivated students. Also the opportunity for questions and help is missing due to the absence of tutors and colleagues. In addition, we have started to use different materials to support education. Some of these are multimedia materials such as videos, audio streams, live audio/video conferences, animations, online tests and virtual discussion rooms to mention some. It seems that work books cannot keep pace with time.

3.1.3. Advances in Learning

Everyone who is working in education and consuming it knows that the best environment is the good old classroom. Students can talk, interact and learn from the teacher and the class mates as well. All participants have the occasion to listen and gather information, to practise to make their own experience and to ask and discuss problems. Since we grew up in the classroom, we are used to it and comfortable with it.

Despite of all mentioned advantages, the classroom has significant disadvantages too. It is very expensive to run a course in the classroom. It is not only a matter of money to pay for the room. We must also be aware of the time we need to get to the place and the teacher is not able to teach at a pace of every individual. Have you ever got bored at school while the teacher was explaining something again to a class mate? If yes, you know what is meant.

It seems to be evident that a combination of the classroom method and the idea of self-paced material would be a good solution for distance learning. That is the point where *web-based education* comes in.

3.1.4. Advantages and Disadvantages

First, a short list of advantages of web-based education in comparison to the classroom method is shown and then disadvantages are also pointed out to draw attention to the fact that all that glitters is not gold.

Advantages

Costs: It is expensive to develop a web-based education system but the delivery cost per course is significant cheaper than traditional teaching methods. Furthermore, there are no expenses for travel, including accommodation, subsistence costs and the room (e.g. classroom).

Convenient: Consumers are free from the boundaries of a specific place and time. It is convenient to choose the time which suits yourself and supports the idea of distance learning.

Learning Gains: Studies supported that the introduction of web-based education in higher education promotes student learning [LM98 Yil00].

Up-to-date Material: Changes of course material on the servers of the web-based education system take affect immediately and support to keep the course content up-to-date.

Information Delivery: In case of online exams the results can be independently delivered by the next scheduled class. In other words, participants of an online exam could get their results right after completing it. In fact, this makes frequent pretesting possible.

Self-paced Learning: As mentioned in the previous sections, web-based education benefits of its self-paced nature. Each individual can work at a pace that suits him or her best, bypassing studying information they already know or repeating parts they have not got yet. According to [Ste99] studies have shown savings of between 30 and 70 per cents in training time due to self-paced learning.

Access: Nearly everybody is able to get in touch with web-based education systems because of the way the information is accessed, that is, by Web browsers which are independent of the operating system and very easy to use.

Disadvantages

Bandwidth: Too less bandwidth is not a big issue these days owing to the good internet infrastructure at universities. This situation could change dramatically when multimedia material such as videos is used to a greater extent.

Acceptance: In any case, poor acceptance of the use of web-based education systems has a big negative impact. It is not enough to provide a technology which is ready for use. Users have to possess the skills to operate the system. Furthermore, they have to understand the *usefulness*, *benefits* and how it can assist effective learning to accept web-based education systems. Also student's demographics, learning styles, particular life characteristics, access to the necessary technical resources, past experience with the technology, all may play a role in the student's adoption of technology [FY05].

3.1.5. Conclusion

This chapter has shown that web-based education is a strong competitor to classroom-based education and work books, and is bound to transform education. The advantages of web-based learning were stated as well as the issue that the success of an e-learning system depends very much on the acceptance of students and teachers. Finally, it is to say that this is a hot topic which also starts to effect mobile learning on mobile devices such as Palms, PocketPC, Web Tablets, cell phones, etc.. For more information see [WST+05].

Now we must be motivated enough to head to some specific applications of web-based education discussed in the next chapter.

3.2. Application of Web-Based Education

After getting motivated in the previous section, this section shows a specific implementation of a web-based education system at Graz University of Technology (TUG). We are going to have a look at the system's architecture as well as the tools it provides.

Web-based education has been a very hot topic in recent years and especial universities put a lot of effort into the development of it to advance their teaching success. Besides the commercial and very well known e-learning systems for educational institutions, which are WebCT[1] and Blackboard[2], many institutions are developing their own solutions.

For example, a network teaching platform based on Java's J2EE[3] technology has been designed and implemented at Zhejiang University in China recently [HQH05]. J2EE was used because of the advances in Java techniques. It applies techniques of composition which makes the components reusable. Furthermore, the Java language is independent of the running environment. The platform provides user groups, online assignments, courseware and message management, online communication and submission of assignments (e.g. images, source code, zip archives).

Another example of a web-based education system was introduced at the Open University of Hong Kong (OUHK). The main aim was to support three computer programming courses with a population of about 400 students in each first year course [NCKC05]. They built a web-based programming course which provides a full programming environment accessed by simple Web browsers. Thus there is no need to face problems when setting up an Integrated Development Environment (IDE), installing Java SDK (Software Development Kit) or setting the CLASSPATH for the Java compiler.

The last example of very recent developments of web-based education shows how different the innovations are. The WTS (Web-based Teaching Support) system supports an online view of a physical classroom with desks and seats. The features are password protected access, online attendance checks, seating layout view, personal notifications, scheduled publication areas and statistical analysis [CXHK05].

[1] http://www.webct.com
[2] http://www.blackboard.com
[3] http://java.sun.com

These are just a few examples of developments at the time of writing. Now we sneak an in-depth view of the web-based education system at the Institute for Information Systems and Computer Media (IICM) at Graz University of Technology.

3.2.1. WBT-Master

"WBT-Master[4] is an innovative web-based training (WBT) tool that supports the construction and delivery of Internet based courseware and provides all other important WBT information services on the base of a unified HM-Data Model [Scea]".

In other words, WBT-Master provides a set of tools to support knowledge transfer processes. It speeds up and facilities the flow of knowledge in a personalised and on-demand fashion. The training tool makes use of current advanced web technologies and standards to transfer knowledge from people possessing it to the people who need it [Hel06]. As all mentioned examples above, WBT-Master also presents qualified material in a structured manner, visualised by ordinary Web browsers as well as it tries to shift the methodology from conventional online course to knowledge transfer in a more general sense, implementing *Web-based Tutoring, Knowledge Mining, Collaborate Problem Solving, Mentoring and Knowledge Delivery* [Sceb]. Later in this paper we will show the use of the new methodology by typical scenarios (see Section 3.2.4).

As shown below, WBT-Master implements security mechanism (for users, teams and projects) and tracing capabilities to monitor the users' activities and progress. Furthermore, as said all services are based on a unified data structure making data reusable by other people and services.

First, let us turn to the architecture of the WBT-Master and then to its set of tools. Finally, a typical tutoring scenario will be given.

3.2.2. Architecture

The WBT-Master is based on the very well-known client-server network architecture. As depicted in Figure 3.1 on page 37, the client-server architecture works on the principle of separating the client from the server which replies to requests from the client. In case of the World Wide Web (WWW), the HyperText Transfer Protocol (HTTP) regulates the communication between the client and the server. The HyperText Markup Language (HTML) is used as the data exchange format. Thus a server can be seen as a HTML data storage where all documents can be located by means of the Universe Resource Locator (URL).

WBT-Master Client-Server Architecture

WBT-Master extends the simple WWW client-server architecture by complex data objects instead of only storing HTML files, adding a Servlet engine on the server side and by using JavaScript on the client side (see Figure 3.2 on page 37).

[4] http://coronet.iicm.edu

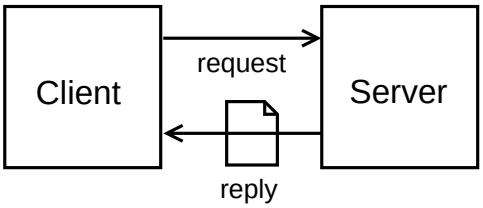


Figure 3.1.: Shows the client-server architecture

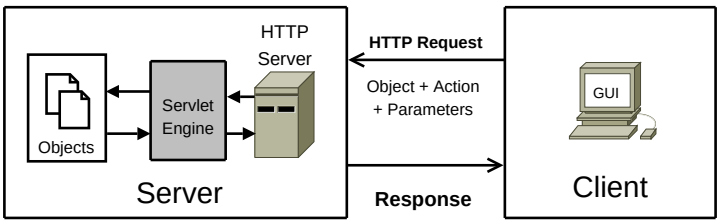


Figure 3.2.: Shows the WBT-Master client-server architecture

Data objects are complex and predefined Java objects representing documents, learning courses, portals, etc. for example. They are just instances of an abstract Java class which predefines properties and actions for a specific object. The data objects are persistent and reactive. The former means that the WBT-Master server can apply *actions* which alter a current state of the data object and the changes are available to other participants. The data objects are reactive as the object replies to an action with data with respect to the current state of the object and the user's context.

On the WBT-Master server a Java Servlet Engine extends the functionality of a standard Web server. For example, Apache Web Server[5] with the Apache Servlet module and the JServ Servlet Engine can be used. The Servlet Engine allows to run small Java programs implementing actions for data objects. Furthermore, the Servlets are build on the Java Factory pattern concept which makes it easy to change the physical format of the object even at runtime. In Section 4.3.3 the Java Factory pattern is explained in detail.

The client side is extended by JavaScript and signed Java applets to visualise all actions for a current data object, convert a user selected action into an HTTP request and visualise the action's results sent back. A Graphical User Interface (GUI) is supported (see Figure 3.3 on page 38 for an example) to bring the vast variety of possible actions to the user's screen.

[5]http://httpd.apache.org

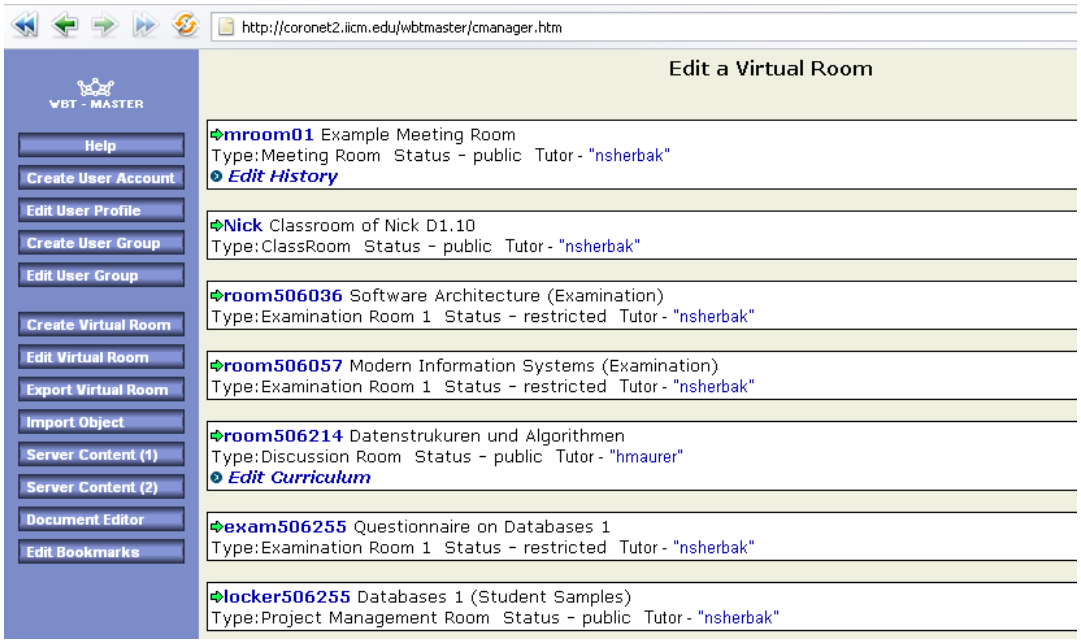


Figure 3.3.: Shows an example of the WBT-Master graphic user interface

Typical Communication

Looking at Figure 3.2 on page 37, a typical request-response sequence looks like the following:

1. The user chooses an action from the GUI inside his preferred Web browser. The JavaScript converts the user's choice into an HTTP request consisting of the URL of the data object and the desired action with its parameters. This HTTP request is sent to the WBT-Master server.

2. The HTTP request is received by the HTTP server and the parameters of the action are passed on to the Servlet Engine which performs the specific action for the specific data object. As mentioned above this might alter the current state of the object itself.

3. The data object is reactive and replies to the action with a collection of data. The Servlet Engine produces a suitable HTTP document of the object's reply for the client.

4. Finally, the HTTP server responses with the HTTP document, received from the Servlet Engine, to the former client request in step one. The client's Web browser visualises the results and further possible actions (it starts with step one again).

As we have seen, WBT-Master uses the well-known client-server architecture and extends it by state of the art Web technologies. This enables WBT-Master to manipulate

data in a very unlimited way. To put it simple, it is capable of giving solutions to complex problems. Because of the simple client-server architecture, the system can be accessed with any simple Web browser anywhere, at any time.

3.2.3. Concepts and Tools

This section deals with the concepts and tools defining the functionality of the WBT-Master system. An overview is given to show the capabilities of the system with reference to the WBT-Master User Manual [Scea].

User Management: Like many other systems, a user is defined by a unique login name and a password. Every user can provide optional records such as the phone number, the fax number, the URL of the user's home page etc.. All user-specific data are stored in a data object called *Business Card*. In addition, every user plays a role in the system. The four defined user roles are *learners, tutors, authors* and *WBT-Administrators*, where each role defines an appropriate access scheme for tools and resources of the system.

A group management is also implemented to manage collection of users (e.g. teams, learning groups etc.) and their rights to specific learning resources.

Repositories: The system deals with Multimedia documents being a combination of text, graphics, Applets, Scrips, videos and audio. All documents are stored in a repository which can be seen as a hierarchical directory structure.

Hypermedia-Data Model: The courseware consists of structured collections (or S-Collections) encapsulating members together with an internal structure. The internal structure describes the link topology between their members.

Figure 3.4 on page 40 shows an example of a simple S-Collection. Document A is the head of this collection and the arrows indicate the links between the documents. The system provides a predefined collection such as *Folder, Envelope, Menu* and *Freelinks* which maintain the link topology automatically. As you can see in this example, when browsing the collection, the direct navigation from document A to document B and C is allowed but not to document D. It is important to mention that a member document of a collection can also refer to another collection.

This object oriented model of encapsulating documents enables users to customize their own collection structures and allows collections to be reused.

Portals: The concept of Portals allows the WBT-Master system to handle external references (i.e. URLs). External static or dynamic content is fetched by the WBT-Master server and treated like local documents.

Learning Units: A learning unit is a set of learning material or other learning units presenting a certain topic. The system serves predefined templates to build learning units easily.

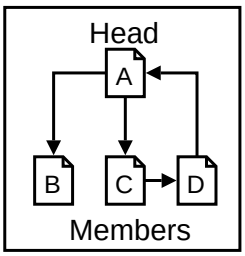


Figure 3.4.: Shows a simple example of the internal structure of an S-Collection

Knowledge Cards: Knowledge Cards contain information on learning resources for a specific topic. The idea is very similar to Learning Units but concentrates on giving information where to find resources needed to gain a certain knowledge.

Personal Desktop: The system consists of a vast number of tools and thousands of different resources making it difficult for the user to keep the overview. The user only wants to work with a few tools. A personalised graphic interface tackles this issue.

Annotations: It is possible to attach an annotation to a document existing in the system. For example, a tutor can add the latest developments or give answers to frequently asked questions (FAQ) with reference to an article. All users reading this article will be able to see the annotations.

Discussion Forums: In addition to the usual functions of a discussion forum the WBT-Master system provides user notification for new replies, a forum chat session and the feature to exchange private messages.

Brainstorming Sessions: These sessions are like discussion forums but every contribution belongs to one of the following classes: question, idea, pro argument, contra argument and comment. Every argument contribution (pro or contra) also holds a confidence attribute which is a number of points reflecting an agreement or disagreement over an idea. Thus, an automatic ranking of ideas is possible.

Mentoring Sessions: It is a special way of synchronous communication. One mentor controls what many students can see on their screens. It is comparable to the blackboard in the classroom at school. As in the classroom the students can give feedback by means of text chat or voice messages at any time they want to.

Virtual Rooms: Virtual Rooms are special training environments such as *Virtual Classrooms* and *Virtual Meeting Rooms*. For example, the former environment gives the user a very strong classroom feeling because of an existing classroom forum, a blackboard, etc..

Questionnaire: The questionnaire is the electronic equivalent to an exam carried out with pen and paper. It is a special document which records interactive answers from students to questions. The user management guarantees that only permitted students are allowed to do a questionnaire.

Progress Tracking: The tracking system enables tutors to keep track of the activities of students. It provides statistical data such as access time of learning materials and test results.

3.2.4. Typical Scenarios

In the following, a typical scenario is shown to give users, who have never used the WBT-Master system before, an idea how it works. The scenarios cover the core elements of the WBT-Master methodology and more examples can be found in the user handbook (refer to [Sceb]).

Web-Based Training Scenario

In this scenario a tutor has to conduct a training session on a regular basis. The tutor develops a special internet course with the help of the courseware author. After announcing the new course, learners can subscribe themselves and start the training session.

The following tasks and actions have to be carried out to implement the described scenario:

1. Developing a learning course on the server
   - Develop relevant training material on your local hard drive
   - Upload the materials to the WBT-Master server
   - Extend the uploaded materials with portals, additional referential documents, questionnaires etc.(see Section 3.2.3)
   - Combine the uploaded documents with other learning resources on the server
   - Publish the new training course
2. Announcing the course on the server
3. Creating user accounts and user groups
4. Working with the new learning course
   - Browse and search the course and work with the course map
   - Annotate course elements
   - Subscribe for the course

- View progress tracking info

5. Communication between tutor and learners by means of the course forum
    - Read and write contributions (and follow-ups)
    - Modify or delete contributions
    - Work with the course chat

3.2.5. Conclusion

It has been deduced that applications of web-based education are under steady development. Some applications were already able to prove their success and usability in real life. It is noticeable that many web-based education systems implement more or less the same concepts such as *client-server architecture, user groups, course forums, online assignments* and *progress tracking*, just to mention a few.

The WBT-Master system, running at the Institute for Information Systems and Computer Media (IICM) at Graz University of Technology, has also been described. The architecture of this web-based education system was discussed as well as several typical concepts of the system.

Although we have seen successfully employed web-based education systems with a vast number of tools, we will find out that this is not enough to build a mature web-based system which can cope with all arising issues. The next two chapters address these issues in more detail.

3.3. Learning Process

This section looks at learning processes to find out what web-based education systems actually have to deal with. First, it starts with the nature of learning processes to get an idea of how difficult it must be to manage them. This new insight leads us to the learning process life cycle and the link to the business process life cycle described at the beginning of this paper. Then, a typical process example is shown to illustrate a real-world scenario of a learning process. Finally, a conclusion completes this section trying to draw attention to the complex nature of learning processes and giving a good starting point to introduce the need for learning process management and its problems.

3.3.1. Process' Nature

The typical characteristic of a learning process is change. Everyone, especially institutions related to education such as, for example, universities, schools and colleges, must know that learning processes are steadily changing. In fact, it is their main job to implement the changes and provide the best education for their clients, that are, the students.

According to [Hel06] such changes can be divided into top-down and bottom-up changes. The former starts with changes in the learning context, such as organizational structures or didactical approaches and then cause the learning processes to change. The latter, in contrast, starts with changes in some of the learning process components and causes remaining learning process components to change.

All of the following top-down changes lead into changes in the learning procedure, learning execution, technical infrastructure and communication infrastructure. In other words, it changes the learning process.

Knowledge and skills: Learning material has to be changed according to the skills and knowledge of the learning participants.

Organizational structure: When, for instance, members leave, new members join or project teams change, it affects the learning process. For example, when a teacher is replaced, he or she is very likely to change the learning material or even the didactical approach.

Didactical approach: Also in education it is typically to adjust and improve to keep pace with the development and time. If a didactical method proves to be better and more successful, the old approach is adjusted or it is even replaced.

Also the following bottom-up changes have a deep impact on learning processes:

Learning content: It is vital to add, update or modify learning content to survive in the information age we are living at the moment.

Technological infrastructure: Apparently changes related to technological infrastructure and tools directly effect learning processes. For example, if you replace a tool of the e-learning system, it could change almost all running learning processes.

These typical examples show how dynamic a learning process is. Thus, we can say that learning processes have a life cycle. Let us look at it next.

3.3.2. Lifecycle

Experience showed that the learning process life cycle consists of the following phases depicted in Figure 3.5 on page 44 [Hel06]:

Model: In this phase the learning process is modelled. It starts with the creation or selection of learning content. Further, the learning procedure , the communication and collaboration practices are chosen.

Implement: This phase maps the learning procedure onto the execution procedure. This means that you have to select, integrate, configure and deploy the appropriate tools that are needed for executing the modelled learning procedure.

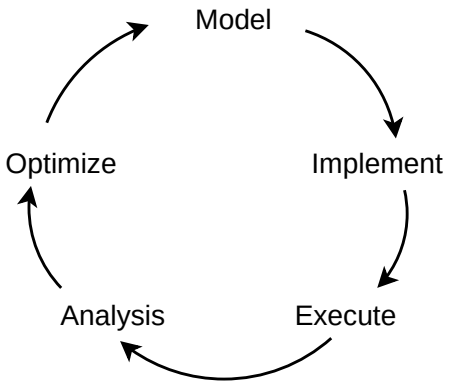


Figure 3.5.: Shows the learning process life cycle

Execute: Participants of the learning process follow the learning procedure in this phase. Executing the learning process, they are supposed to achieve the defined learning goal by means of provided tools supporting studying, communicating and collaborating with each other.

Analysis: The execution of the learning process is being monitored and analysed to collect helpful data for optimizing the process in the next step.

Optimize: The current learning process is improved to fit a particular learning situation with the provided data and analysis from the previous phase.

Have you had a déjá vu when looking at the learning process life cycle and reading what happens in each phase? It is the right time to sneak a look back at the business process life cycle (see Figure 1.2 on page 4). It is easy to notice that both life cycles are pretty similar and their phases are even the same.

In fact, business processes and learning processes have to deal with the same issues and need to be supported by a process management covering all phases of the life cycle. We have seen earlier that learning processes are steadily changing. Thus, it is crucial to put learning process changes through as quickly as possible to offer up-to-date education. As for business processes, learning process management must support all phases of the life cycle. If it does not, the process of change takes longer, needs more resources and is therefore more expensive.

To sum up, business processes and learning processes can be thought of as equal. They have the same nature, the same life cycle and come along with the same problems. As a logical result, the same concepts, technologies and tools could be applied to cope with learning processes and their life cycle.

So much for the theory about learning processes. Let us look at an example to understand clearly how a typical learning process looks like.

3.3.3. Learning Process Example

A typical example of a learning process is shown here. It illustrates a real-world scenario of distance learning which is used throughout this paper. The learning process is described by BPEL (Business Process Execution Language) and can be found at Listing A.5 on page 104.

Let us assume that a tutor has to conduct an university course dealing with the basics of the Linux operating system. Unfortunately, the person is on a research trip somewhere in the world. Therefore, he or she wants to carry out an online course involving students rather than providing reading activities only. In other words, the tutor wants the students to write and ask questions in a discussion forum and to do online exams.

Figure 3.6 on page 46 depicts this learning process starting with a reading activity about an introduction of Linux followed by another reading activity which gives another introduction of Linux written by its developers. Then, the students have the occasion to discuss the literature in a discussion forum provided by the used web-based learning system. At the next stage, the students are tested by means of an online examination. This examination is only conducted to give feedback to the learners about their current knowledge and weaknesses. Next, further reading material is served about how to install a Linux operating system. After this activity learners have again the chance to discuss open questions related to the online course. Finally, the final exam takes place but if a student does not pass it he or she has to read another tutorial about Linux and must do the exam again. This is the stage where the online course ends.

3.3.4. Conclusion

It has been seen that the typical characteristic of learning processes is change. Learning processes change steadily because, for example, learning material, organizational structures, didactical approaches and technological infrastructure are changing. Basically these changes are necessary to provide decent and up-to-date education.

In the next step the learning process life cycle was introduced. We draw attention to the fact that the business process life cycle described at the beginning of this paper consists of the same phases as the life cycle of learning processes. In addition, the importance of supporting all phases of the life cycle by learning process management was motivated.

Further, a learning process example was given to help understand how a typical process looks like. A course about the basics of the Linux operating system was used to illustrate a real word scenario.

However, the crucial conclusion is that learning processes and business processes can be thought of as being equal in terms of their life cycle and demands regarding management. As a consequence, we have suggested applying Business Process Management for managing learning processes.

Anyway, process management is the critical factor and it has to support all phases of

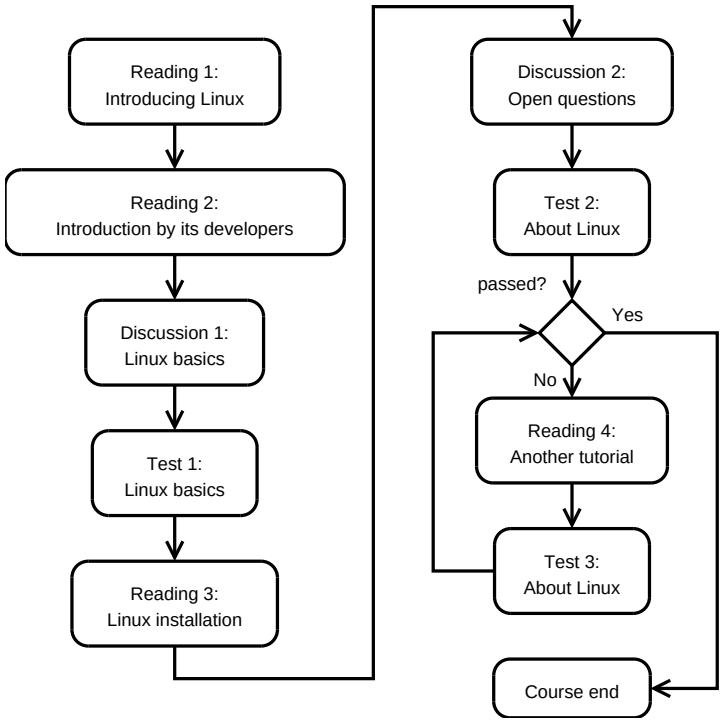


Figure 3.6.: An example of a learning process

the life cycle. Thus we are moving on to learning process management.

3.4. Learning Process Management

As we it has been seen in the previous section, *Learning Process Management* is needed to cope with the very complex learning process life cycle. Most popular web-based education systems, in Section 3.2 some were mentioned, only support individual learning tasks rather than learning processes.

During the recent years, some research has been done and several standardizations have emerged taking the process oriented nature of learning tasks into account. Among these standardizations *Sharable Content Object Reference Model (SCORM) Sequence and Navigation* and *IMS Learning Design* can be found. The following sections are going to give a brief overview about these specifications and introduce two Learning Management Systems. The Learning Management System introduced at the end, simply named *Learning Process Manager*, is covered in more detail as it is based on open Web standards and the concept to support learning processes through Business Process Management (BPM). Furthermore, it was used to develop and test the software described later in this paper.

3.4.1. SCORM Sequence and Navigation

"SCORM is a collection of standards and specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reusability of web-based learning content"[6]. The main parts of the collection are the Content Aggregation Model (SCORM-CAM), Sequence and Navigation (SCORM-NS) and Run-Time Environment (SCORM-RTE). SCORM-CAM specifies how to build and organise learning content while SCORM-RTE deals with the integration and interaction between the Learning Management System and the learners.

SCORM Sequence and Navigation is of interest here as it specifies adapting learning paths which reorganizes the presentation of learning content according to the situation of the learners. It is based on the IMS Simple Sequencing (SS) specification which specifies a method for representing authored learning experience [DT04]. SCORM-SN applies and extends this specification and defines the structure and sequencing strategy of learning activities. In addition, it specifies how navigation events, of the learners for example, can be triggered and processed.

3.4.2. IMS Learning Design

The IMS Learning Design specification supports a wide range of pedagogies in online learning and was developed by the IMS Global Learning Consortium. The mission of the non-profit organisation is to "support the adoption and use of learning technology worldwide".[7] The IMS Learning Design specification is only one specification among several others which can be found on the home page of the organisation.

IMS Learning Design provides a generic and flexible language describing learning scenarios and different pedagogical concepts. These concepts are so-called Units-of-Learning which are exchangeable between e-learning systems. A Unit-of-Learning structures all learning activities and might also refer to external learning content, external tools or services [Hel06].

[SMGMO+05] implemented a IMS complaint Learning Management System (LMS), called e-Aula, supporting several specifications such as IMS CP (for aggregating content into packages), IMS QTI (for tests and assessments), IMS LIP (for storing information about the learners) and others. The aim of the development of the LMS was to evaluate different e-learning standards. The outcome shows that it is possible to build a decent LMS with existing IMS specifications and Java Web technologies.

3.4.3. Learning Activity Management System

The Learning Activity Management System, in short LAMS, is another example of a LMS. Its concept is to separate presentation, logic and workflow and is based on the

[6] http://www.adlnet.gov/scorm/
[7] http://www.imsglobal.org

Java programming language. LAMS is meant to be a tool for "designing, managing and delivering online collaborative learning activities".[8]

[Hel06] stated that it is a fully-fledged e-learning system although it has a serious drawback. It is not possible to sequence, structure and control an external system by LAMS. As a consequence, users using their own system have to move over to LAMS to obtain LAMS's advantages. In most cases it is not a realistic option to abandon the own system which might have been running for many months or years.

3.4.4. Learning Process Manager

Now we are going to get an in-depth view of the Learning Process Manager (LPM) which was used to test the developed program discussed later in the paper (see Chapter 4). First, the architecture and the technical implementation is described. This is followed by the explanation of the learning process, how the example mentioned in Section 3.3.3 can be implemented and how the execution of this process looks like. At the end you will be able to notice the advantage as well as the big drawback of this LPM.

Figure 3.7 on page 48 depicts the architecture [Hel06] of the Learning Process Manager. The main modules of the system are represented by rounded rectangles and the arrows represent the communication protocol or interfaces between them. Let us turn now to the technical implementation of each module of the system.

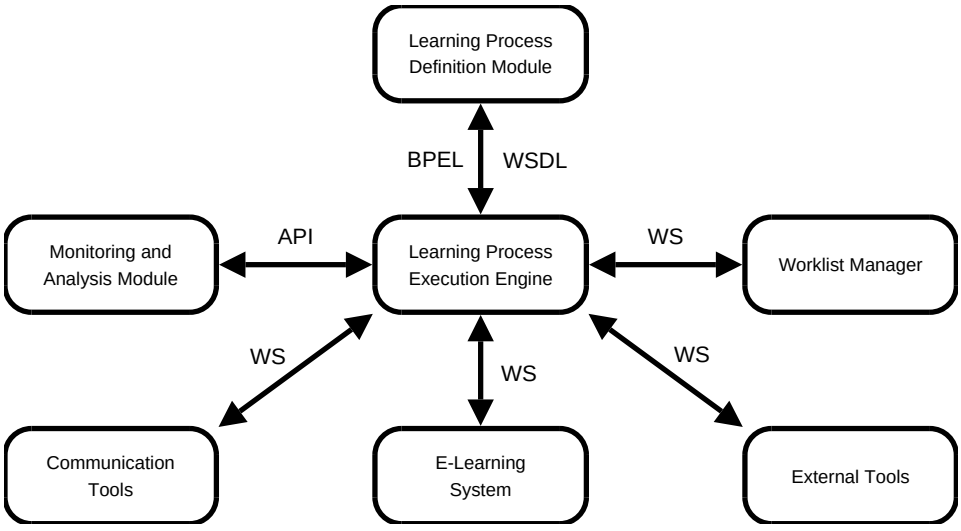


Figure 3.7.: Shows the architecture of the Learning Process Manager

[8] http://www.lamsinternational.com/

Learning Process Execution Engine

The execution engine is the main part of the system and is responsible for the execution of learning processes. They are defined by the Business Process Execution Language (BPEL) specification. All active and running processes in the execution engine are instances of one of the learning processes defined by means of BPEL. After the execution engine receives an event specified in the BPEL file it creates such a new instance. Each learning process runs in a unique execution context. This context holds information about the process needed to identify a specific running instance to, for example, exchange data between processes or make sequencing decisions according to the process definition file (i.e. BPEL file). When the condition to terminate a process is fulfilled, the execution engine destroys the instance and cleans up the execution context.

As you can see in Figure 3.7 on page 48 the execution engine's communication to other modules is mainly based on Web services. In addition, it provides an Application Programming Interface (API) to monitor and analyse the running processes.

The current execution engine is an open source BPEL engine called ActiveBPEL[9] but any BPEL conform engine can be used. The ActiveBPEL engine is written in Java and runs in any standard Servlet container. At the time of writing, Apache Tomcat[10] was used.

Learning Process Definition Module

[Hel06] said that this module is based on a so-called pedagogical vocabulary defining typical collaborative learning activities (e.g. reading, discussion, testing, etc.) and typical user roles such as teacher, student or tutor. Furthermore, learning process templates had been prepared which was defined by the learning procedure and a corresponding execution procedure. The learning procedure can be seen as a chain of learning activities defining which activity comes after another. This flow also allows branches, loops and condition statements. The corresponding execution procedure maps the learning activities of the learning procedure onto the execution layer. To put it simple, for example, the discussion activity is mapped onto a discussion forum or the writing activity is mapped onto an upload tool.

This concept goes hand in hand with the idea, discussed in earlier chapters, to separate the description of the workflow from the actual implementation but it is not implemented yet. In fact, the Learning Process Definition Module is just a collection of BPEL and WSDL files. As we know, BPEL is already the language to define how processes have to be executed and it does not provide enough abstraction for a typical user. Later, we are going to show that it is difficult for a normal user to edit BPEL files although users would have to edit just a few lines. This lack of abstraction is the main drawback of the current Learning Process Manager and therefore some time was invested to develop a tool to tackle this issue (see Chapter 4).

[9] http://www.activebpel.org
[10] http://tomcat.apache.org

Worklist Manager

It keeps track of all working and learning tasks for all users and for all processes running in the execution engine. Additionally, it provides a user interface (see Figure 3.9 on page 53) accessible by a Web browser to present the activities for a particular user. The Worklist Manager is written in Java and uses Apache Axis[11] to implement SOAP (Simple Object Access Protocol) for exchanging structured information with the execution engine.

Monitoring and Analysis Module

It provides a graphical user interface (see Figure 3.11 on page 55) to monitor properties (e.g. the time needed for execution) of all running learning processes. In addition, it is a very valuable tool for debugging the communication between the execution engine and all other modules.

E-Learning System and Tools

The e-learning system and all other tools offer their services by means of Web services. Currently, WBT-Master is the chosen e-learning system which was already described in detail (see Section 3.2.1).

Example of a Typical Scenario

In the following, a typical learning scenario will be shown to illustrate how the modules, described above, interact with each other. Let us assume that a teacher wants to set up an online course and implement the learning process depicted in Figure 3.6 on page 46. All activities such as reading, discussion and testing are supported by the e-learning system by means of Web services and HTTP responses. In our case the e-learning system is WBT-Master.

The following preliminaries are necessary to set up the course:

1. The teacher has to prepare the learning content consisting of four papers about Linux and proper questions for two exams. Then, the learning content must be uploaded to the WBT-Master server. After extending the uploaded content with additional tools (i.e. discussion forum and questionnaire) user accounts and user groups for the students need to be created. No further provisions on the WBT-Master system have to be done.

2. The next step is to model the learning procedure of the execution engine. It has already been mentioned that BPEL is used for this purpose. If the teacher is lucky, he or she might find a template provided by LPM.

[11] http://ws.apache.org/axis/

In this case, only a few lines in the BPEL template file have to be changed. Listing 3.1 on page 51 is a snip of the whole BPEL file (see Listing A.5 on page 104) describing our learning procedure. For example, the lines *6, 7* and *8* have to be altered in order to point the first reading activity to the right resource in the e-learning system (line *6*) and to submit a title and a message for the user to the LPM (line *7-8*). This has to be done for every activity in the BPEL template.

Listing 3.1: Shows where to change the BPEL file for a reading activity

```
1  <copy>
2    <from>
3      <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" soapenc
           :arrayType="soapenc:string[4]" xsi:type="soapenc:Array">
4        <item xsi:type="soapenc:string">reading</item>
5        <item xsi:type="soapenc:string">student</item>
6        <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/
             wbtmaster/courses/linux1_start0.htm</item>
7        <item xsi:type="soapenc:string">title=Introducing Linux</item>
8        <item xsi:type="soapenc:string">message=First read this!!</
             item>
9      </in1>
10   </from>
11   <to part="in1" variable="wi" />
12 </copy>
```

If the teacher does not find the right template for the online course, a new BPEL file has to be written. In spite of the fact that parts of existing BPEL template files can be taken over to build the new file, there is no doubt that a person who is not familiar with XML or similar technologies will have difficulties.

After deploying the new BPEL file into LPM the online course is ready for use.

Figure 3.8 on page 52 shows the communication between the Execution Engine, the Worklist Manager, the E-Learning System, a student and a teacher. To keep it simple, only the communication of the first reading activity and the final examination of our example will be described. All other activities communicate likewise.

First, a student logs into the e-learning system using his or her username and password. Then, for the first time, the student connects to the Worklist Manager by means of a Web browser. The Worklist Manager sends a message to the Execution Engine which creates a new instance of a learning process and its associated execution context. As described in the BPEL file, the Execution Engine sends a message to the Worklist Manager registering the first reading activity. The message holds information about the title of the document which should be read, a message for the student and where the document is located in the e-learning system (i.e. URL). From this point on, the student

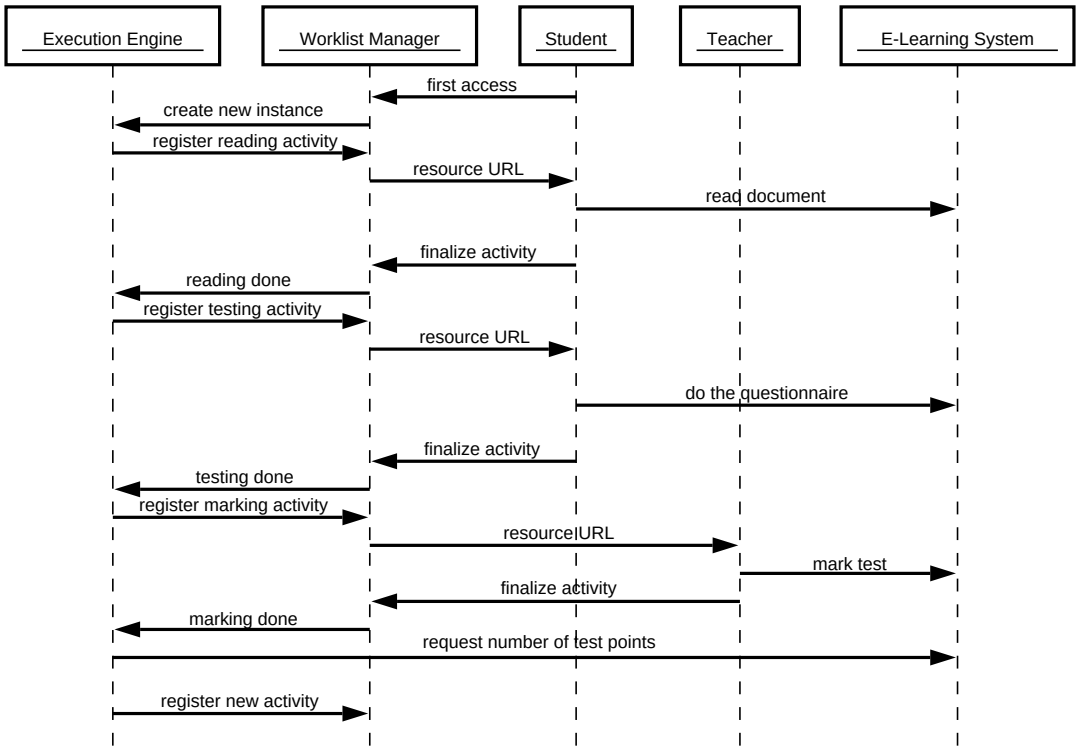


Figure 3.8.: Shows the communication between modules of the Learning Process Manager for a reading and a testing activity

will see a new reading activity on his or her task list. The reading activity can be carried out by clicking on the URL referring to the document in the e-learning system. After finishing the reading, the student finalizes the activity by using the Worklist Manager's graphical user interface. Finally, the Worklist Manager immediately sends a message to the Execution Engine informing that the reading activity is done. This completes the first activity.

As you can see in Figure 3.8 on page 52, the beginning of the testing activity is the same but does not end up at the same point due to the fact that the teacher also has to mark the test taken by the student. The Execution Engine registers the marking activity for the teacher after receiving the message that the test was done. Like for the student, a task list is provided for the teacher. When the teacher checks his or her list, he will find the new marking activity and a link to the service for correcting and marking the test. As before, the activity is finished by finalizing it by means of the Worklist Manager which sends a message to the Execution Engine. At this stage the marking activity is completed but in order to decide whether the student passed the test or not, one more step has to be done. The Execution Engine sends a message to request the number of obtained test points to a Web service of the e-learning system implementing this feature. Finally, the Execution Engine has got all information needed to decide which branch to

take in the learning process.

A few screenshots were taken to get a better understanding of the look and feel regarding the graphical user interface of the Worklist Manager and the Monitoring Module. Figure 3.9 on page 53 shows the graphical user interface of the Worklist Manager for the student. You can see all completed activities and that currently the student has to wait until the teacher marks the test. Similarly, Figure 3.10 on page 54 demonstrates what the teacher sees. He can keep track of the student's progress and see his or her active activities (e.g. marking a test). As stated earlier, a Monitoring Module exists monitoring all running processes. Figure 3.11 on page 55 shows the process details for our specific example at a specific time. To be more accurate and as you can see the screenshot was taken right after registering the marking activity at the Worklist Manager.

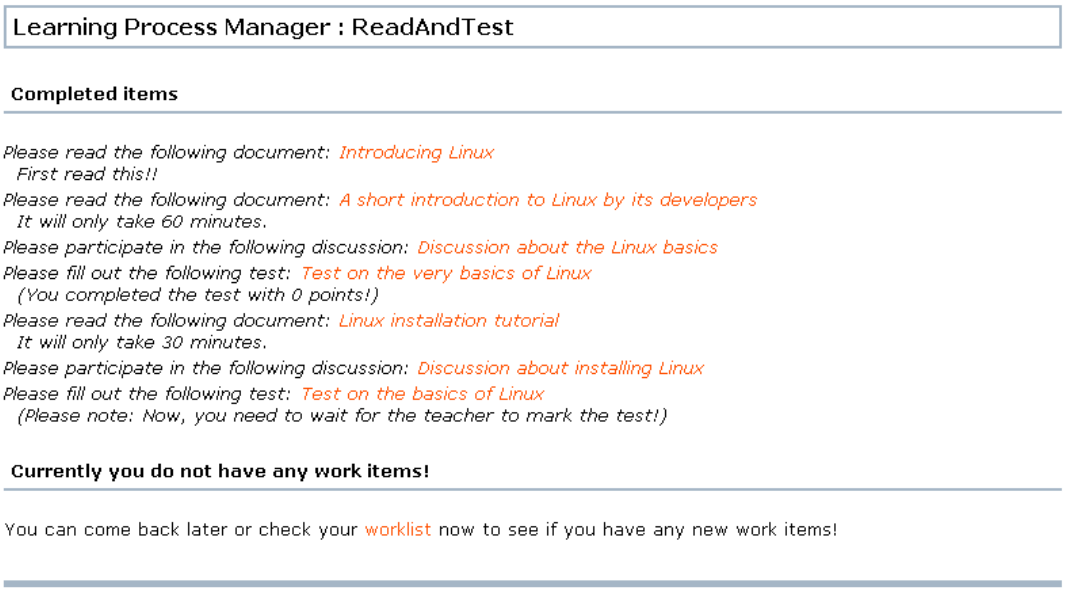


Figure 3.9.: Shows an example of the graphical user interface of the Learning Process Manager for a student

Now we have talked enough about Learning Process Management and want to conclude in the next section.

3.4.5. Conclusion

Two standardizations have been discussed which support learning technology. *SCORM Sequence and Navigation* and *IMS Learning Design* are a set of mature specifications used by developers worldwide. As shown, a very recent evaluation of e-learning standards proved that it is possible to build a Learning Management System with existing IMS specifications.

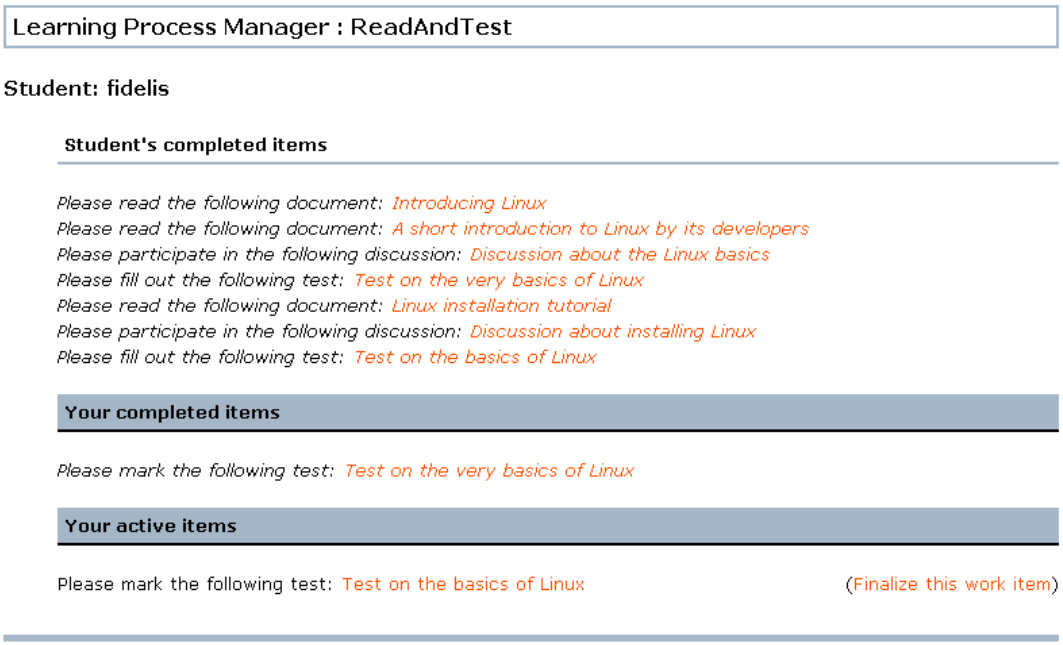


Figure 3.10.: Shows an example of the graphical user interface of the Learning Process Manager for a teacher

In addition to specifications some time has been invested to explain the function of the Learning Process Manager. The combination of an existing e-learning system (WBT-Master in this case) and the Learning Process Manager has a big advantage. This system uses a huge amount of tools and functions provided by the e-learning system and treats learning tasks as learning processes with technology for business processes. Technology for modeling, executing and analysing of business processes are suitable for learning processes because of their very similar nature. Although we can take advantage of sophisticated technology invented for Business Process Management, one drawback must be admitted. In fact, it is the same problem that occurs everywhere when BPEL is used. The abstraction level of this language is not high enough to allow a normal user to write it without knowledge about XML technologies and a slight technical background. In other words, a *Human-BPEL Interface* is needed.

The next section is going to find out whether tools exist to fulfil the demands of a Human-BPEL Interface.

3.5. Business Process Execution Language Designers

In the following, three Business Process Execution Language (BPEL) Designers are represented which provide a Human-BPEL interface in the manner of speaking. First, an overview of these BPEL designers is given followed by an in-depth view of one designer.

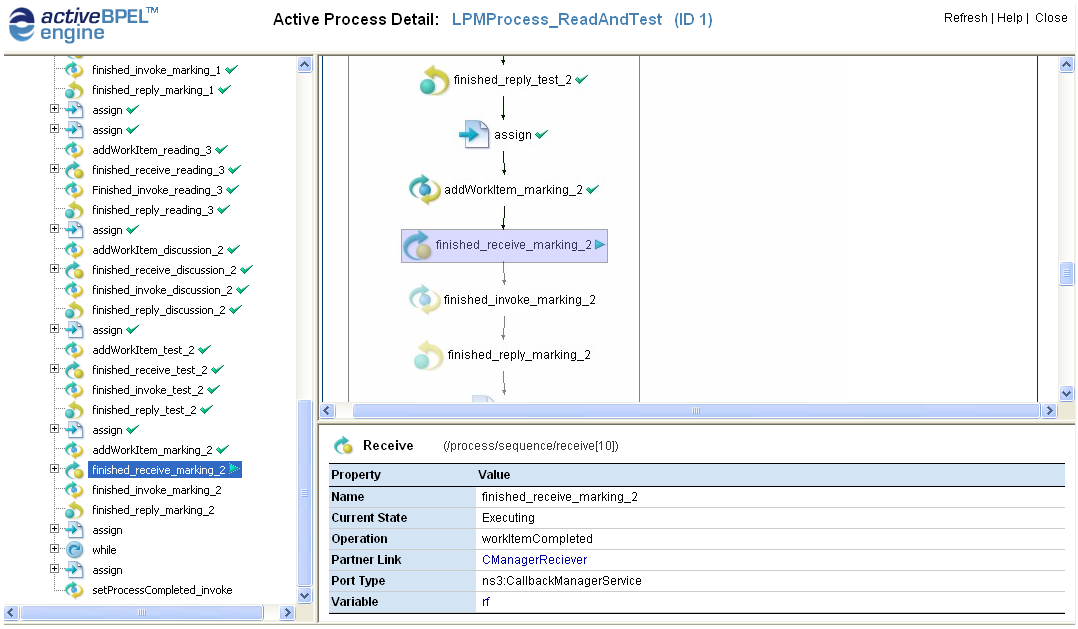


Figure 3.11.: Shows the monitoring capability of the Execution Engine for a specific running learning process

This view covers every step from the installation to the usage of the BPEL tool. Finally, it will be concluded whether such a tool is suitable to support ordinary users for changing BPEL files introduced by the Learning Process Manager (see Section 3.4.4).

3.5.1. Overview

During the research we came across three BPEL designers. All of them are based on Eclipse[12] which provides a development platform and application framework for building software. Eclipse is open source and written in Java. Furthermore, it gives developers the freedom of choice because it supports software development in a multi-language, multi-platform and multi-vendor environment. In this case all BPEL designers consist of the Eclipse application framework and a BPEL plug-in written by the developers of the vendor. The following BPEL designers were found:

BPEL Designer Editor: This editor is an open source project within the Eclipse application framework. The project's goal is to "add comprehensive support to Eclipse for the definition, editing, deploying, testing, and debugging of BPEL4WS flows (Business Process Definition Language for Web Services) in vendor neutral environments"[13]. At the time of writing the project is only a proposal which emerged

[12]http://www.eclipse.org/
[13]http://www.eclipse.org/proposals/bpel-designer/

from unsatisfied developers. Each major software vendor supporting BPEL4WS such as BEA, IBM, Microsoft, Oracle and SAP is pursuing its own runtime engine for BPEL. Thus, every vendor provides its own specific tools to deal with BPEL. BPEL designers are among these tools and developers are calling for a vendor independent solution especially because BPEL4WS is a developing standard. However, it is still a proposal and no source line has been written so far but the project's timeline states that the first version is coming in the fourth quarter of 2006.

Oracle's BPEL Designer for Eclipse: The software vendor Oracle also provides a BPEL designer tool. It is a plug-in for Eclipse which allows developers to model BPEL processes visually. This designer is very similar to the ActiveBPEL designer described in more detail below.

Oracle's BPEL Designer is freely available for development and testing purposes but if you deploy a process into production you have to buy a licence from Oracle.

ActiveBPEL Designer: ActiveBPEL Designer is a visual tool for creating, testing and deploying applications based on the BPEL standard. This Eclipse plug-in was written by Active Endpoints[14], a provider of BPEL solutions. It provides native BPEL support and creates 100% compliant BPEL code by means of its visual tools. All BPEL 1.1 activity constructs and some WSBPEL 2.0 (Web Services Business Process Execution Language) key constructs are implemented. In addition, the developer can toggle between a diagram, a swimlane and a code view of the project. Besides the fact that diagram components of a project can easily be reused in another project, ActiveBPEL Designer offers visual expression editing controls for building scripts such as JavaScript, XPath and XQuery. Also, it is possible to simulate and debug the runtime execution in visual mode. In this mode breakpoints can be set, message content can be examined, changed and also return values for unavailable services are allowed to be defined. The tool also helps to generate all necessary process definition files and deployment descriptors needed to run the process on the execution engine.

ActiveBPEL Designer used to cost about 1000USD before February 2006 but it is free of charge now. Before February it was possible to register for an evaluation period to get a license key for 30 days. Now Active Endpoints is shaking up the BPEL market by giving the designer tool away for free. This step might have been taken as a result of the development of open source BPEL designers such as Eclipse's Designer Editor for example. ActiveBPEL is not open source yet, though.

Let us move on. How to get, install and use Active Endpoints ActiveBPEL Designer to modify the BPEL template files used by the Learning Process Manager (see Section 3.4.4).

[14] http://www.active-endpoints.com/

3.5.2. ActiveBPEL Designer

The next steps show how ActiveBPEL Designer can be used as a Human-BPEL interface to change BPEL template files used by the Learning Process Manager (refer to Section 3.4.4).

Installation

First, you have to fill in the request form[15] to get a response by E-mail telling you the download location of the ActiveBPEL Designer and the license key. Beware of the download size of approximately 115MB.

The installation is very easy and uncomplicated. Open the downloaded archive and start the installation program. If Java is installed, a wizard leads you through the installation process. In case of a missing Java installation you get an error as the designer is based on Eclipse which is written in Java.

After installing the program you have to choose the workspace folder and enter the license key. When the program is started for the first time, you have to choose the workspace folder where all projects are going to be stored. In addition, a license error window will pop up to remind you to activate the copy of the program. Click on activate and copy and paste the license key you received from Active Endpoints. The program can be used now.

Import existing files

An existing BPEL file and its related WSDL files have to be imported into the ActiveBPEL designer before any changes can be made. First, a new project must be created and named. Then, our existing BPEL file (see Listing A.5 on page 104), we want to change, has to be imported into the new project. At the next stage, all necessary WSDL files must be copied into the new project folder. Finally, all WSDL files can be added to the web references and we are able to double click and view the process example depicted in Figure 3.12 on page 58. Now we are ready to change the BPEL template file to meet our needs.

Change the BPEL file

You might remember that we had two cases in our typical scenario of the Learning Process Manager (look at Section 3.4.4). In case of finding a suitable BPEL template reflecting the teachers needs, he or she only has to alter some code lines. To be more accurate, SOAP messages have to be changed to point to the right resource in the e-learning system.

Figure 3.13 on page 59 shows the steps which have to be done to change SOAP messages for a reading activity. You have to be familiar with the BPEL standard and the architecture of the BPEL template file to know where to make the modifications. In

[15] http://www.active-endpoints.com/products/activebpeldes/download.html

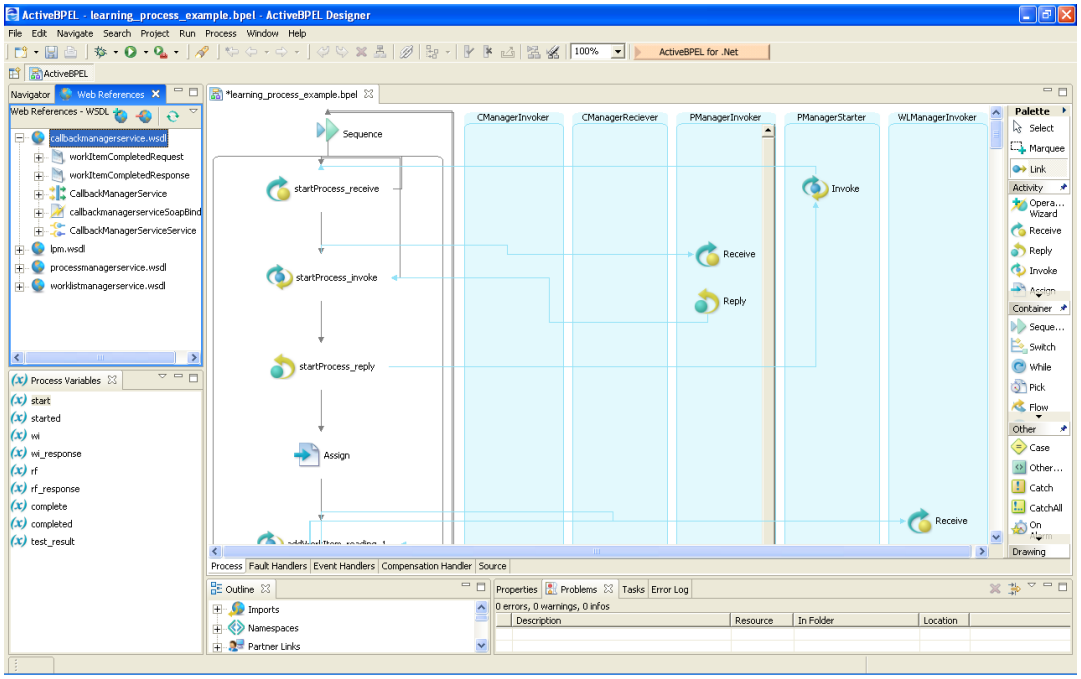


Figure 3.12.: Shows the ActiveBPEL Designer in action. The diagram view is shown as well as the swimlanes of our example BPEL file.

this example, first, the copy operation, placed before the invoke operation to register the reading activity, has to be marked (labeled with *one* in the figure). Finally, the user has to click on the button labeled with *two* to change the SOAP messages in the emerged new window (labeled with *three*). As you can notice the new window shows just a code snip of the original BPEL file.

ActiveBPEL Designer surely can be used to restructure the activities of the BPEL template file or even to build a new file from scratch. In case the teacher is not satisfied with the BPEL template file, ActiveBPEL Designer is a professional tool to change the flow of activities and add other if needed. Although the BPEL designer provides a comprehensive and distinct visual tool to deal with BPEL development, it just reflects the underlying BPEL source files in a visual way. In other words, it does not add any level of abstraction and the user has to be familiar with BPEL technology. Beside of this issue ActiveBPEL Designer is a very helpful tool for every BPEL developer.

3.5.3. Conclusion

In this section three BPEL tools have been looked into. All three, Eclipse's Designer Editor, Oracle's and ActiveBPEL Designer for BPEL support defining, editing, deploying, testing and debugging business processes by means of BPEL. Furthermore, all are based on Eclipse and serve the user with a visual BPEL presentation with the help of

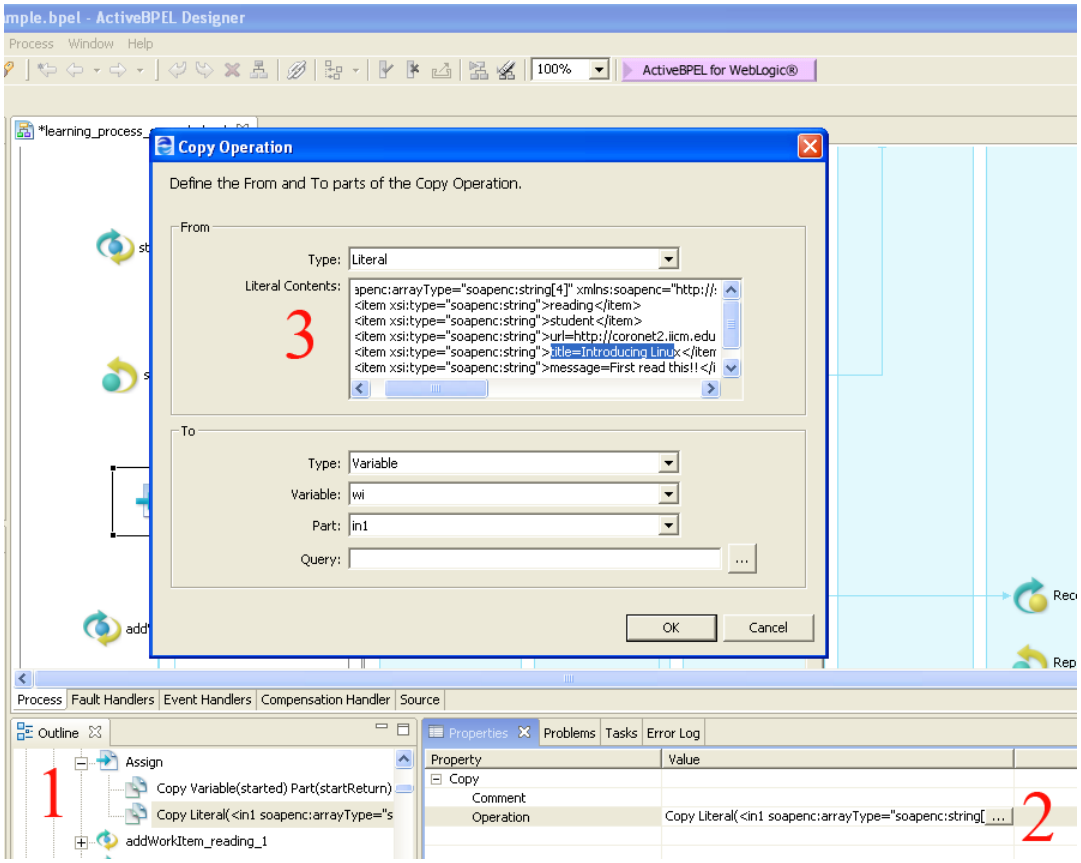


Figure 3.13.: Shows the steps to change SOAP messages in our BPEL file

diagram views.

A closer look has been taken at the ActiveBPEL Designer. It is not time-consuming to get the free program and to go through the installation procedure. In addition, it has been shown what steps are necessary to change the BPEL file of our running example. This practical test was essential to see whether these tools are suitable to provide a so called BPEL-Human interface.

In conclusion, it can be said that these kind of BPEL tools do not provide a BPEL-Human interface we are looking for. We are looking for an interface which is suitable for an ordinary user without knowledge about BPEL and related technologies. All these BPEL tools are meant to assist people who have this specific knowledge. Thus, it can be said that these tools do not tackle the problem of giving more abstraction to the ordinary user. This means that we have to come up with our own software solution.

The next chapter is going to describe this solution and its development.

4. Software Development of a Human-BPEL Interface

This chapter is dedicated to the software development of a new tool providing a Human-BPEL interface. As software development is not equal to programming the first part of this chapter illustrates what software engineering is all about. Besides giving a brief overview of some development models, it is shown why design is an essential part of software development. Then, all typical software development phases such as requirements, design, programming, testing and operation are covered in detail showing all steps of the development of the new tool. Finally, a kind of tutorial is written to show how everybody can extend the functionality of the program easily.

Alright then, let us focus on software engineering, how it emerged, why we need it, and what techniques it comes along with.

4.1. Software Engineering

Although computer science is still a young field the demands have changed from the very early years to today. The first computers were built in the mid 1940s and when we compare the profession of building a house and building software it can be said that building software is still in its infancy. Despite of it being so young the demands of software have extremely changed. At the beginning programs were small in size and were used by technical staff to cope with rather technical problems. Programs used to run offline to calculate results which were used later. The situation has changed, though.

Today programs are getting more and more complex. Developers have to collaborate to be able to write larger programs within a suitable time to meet the user's needs. The typical user has also changed. The programmers and technical staff are not the future users anymore. As we can see every day, almost everybody regardless of any technical background uses programs at work or at home. The fact that programs run online and communicate with each other adds even more complexity. As a consequence of the mounting complexity, the programs are delivered too late, do not behave as the user expects and contain many errors.

To tackle these problems and calling for better software the term *Software Engineering* was introduced. Actually, the term software engineering was meant as a kind of provocation. Why is it not possible to build software like a bridge? Why can we not start from a theoretic basis and use proven designs and construction techniques to build software like we do when we build a bridge [vV93]? These questions obviously raised at the first

North Atlantic Treaty Organisation (NATO) conference in Garmisch (Germany) where the following definition of software engineering was given [NR69]:

> Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The latest accepted definition is this one from the Institute of Electrical and Electronics Engineers (IEEE) issued in 1993 [IEE93]:

> Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Thus, software engineering is not the same as programming. Programming is an important ingredient but not the only one. Imagine a builder building a house. He does not start off piling up bricks. He or she rather takes requirements of the future inhabitants into account and starts designing. The builder or better designer at this stage has to look at factors such as how many people will live there, the financial situation, whether a family or just friends will live there, the hobbies of the inhabitants, etc. to come up with a house which meets the needs of its future users. After the design is agreed the construction can begin.

Over the recent years it has become best practice to apply the same strategy in software development. It is no good starting programming straight away. The software development process has to be structured and the following sections are going to show how it can be obtained and what development models exist to support it.

4.1.1. Structured Development Life Cycle

The structured development life cycle breaks the process of developing software into several activities. Figure 4.1 on page 61 shows the activities starting with *requirements analysis*. The arrows indicate that these activities are performed sequentially. In other words, some software development tasks have to be completed before others can be started.

Figure 4.1.: Shows the structured development life cycle

In the following, we are going to look at these activities in detail keeping in mind that the development of our program was carried out the same way. All of these activities are described in a very similar way in [GM98], [Som01] and [vV93].

Requirements Analysis: This activity forms the foundation of the whole software development cycle. The aim is to get a complete description of the problem. In other words it is investigated what the final software product should do and what its goal is.

In order to identify the set of requirements to archive the goal the communication between the future users and the programmers is crucial. A small misunderstanding, at this stage usually has a big impact on the final software. That is, the huge gap between what the users want and what the programmers have delivered. To avoid any kind of misunderstanding teams consisting of programmers, users and sometimes managers are built. It is their goal to reach an agreement on the requirements the new program must satisfy. During this process developers and users have to interact and try to understand each others by means of discussions, interviews or reading through documents. At the end, after reaching an agreement within the team a document is written summarizing all requirements. This document is called *Requirements Specification*.

[GM98] writes that this final document should have the following qualities:

- Correctness. The specification should correctly identify and describe each of the requirements the system must satisfy. The specification should be without error.
- Completeness. The specification should include every relevant requirement.
- Consistency. The specification should not contain conflicting requirements.
- Clarity. The specification should be written in a language understandable by both users and programmers.

Both users and developers must also be aware that the requirements specification is very helpful to avoid being unfairly blamed if the developed program does not meet the user's needs and they are unhappy.

Design: The design activity helps to reduce costs and to focus on solving a problem. The main advantage of starting with a design first is that it is much cheaper to change your mind while planning the design than to change already implemented code lines. The process of building a house serves as a good example again. It is easy to grasp that erasing a line on the blueprint is cheaper than tearing down a wall. Moreover, abstraction mechanisms help to master software complexity and help to focus on solving the problems. Over the years software developers started following software development methodologies. A methodology is simply a prescribed way of using abstraction mechanisms and consists of three things [GM98]:

- A process - the step-by-step activities and related deliverables that are used to model and construct software.

- A notation - a representation, often graphical, of the subsystems that make up a system and the way they interact.
- A set of heuristics - rules of thumb of figures of merit that give the designer guidance about how the artifact being built should work.

The aim of abstraction mechanisms is to hide details and show only relevant information to be able to concentrate on the important parts of the problems. Some key words supporting abstraction are *data hiding, modularization, encapsulation* and *abstract data types* as well as *structured programming*. The latter introduced subroutines also called functions. Subroutines operate on pieces of data passed between the subroutines. Another mechanism to support abstraction is *object-oriented programming* (OOP). In OOP, classes are the components to build programs. A class holds the data and serves the legal operations that can be performed on it. All these concepts make source code more readable and maintainable and allow developers in combination with notation tools to concentrate on the real problem during the design activity.

In contrast to the requirements analysis activity, a typical team consists of mainly programmers. They design a model solving the problems stated in the requirements specification. Thus, this activity relies very much on the requirements analysis. Usually the whole model is broken down into several parts and the programmers try to specify what every part should do and how they communicate between each other.

The results of the design activity are rather technical documents describing a model, its parts, their communication and their interfaces to meet the requirements stated in the requirements specification document. It also serves as a starting point for the programming activity.

Programming: The next stage is to produce source code and deliver an executable program. Teams of programmers implement parts specified by the design. If they can rely on a good design, the software development can be accelerated as programmers are able to work on their own parts separately. This shows how important the previous design activity really is.

Programmers not only have to write source code in a programming language and by means of tools integrated in IDEs (Integrated Development Environment) offering editors, compilers, debuggers and a document revision control system for example. They also have to focus on writing easy to read source code which is well-documented and which produces a reliable and correct program.

The result of this activity is an executable program specified by the design and meeting the needs of the requirements stated in the requirements specification.

Testing: In fact, as every developer probably knows, this activity is not the only testing activity during the whole software development process. If it was the case, we

would not have to bother about testing what has been done so far before this stage. No programmer would suggest that he or she is able to write hundred of code lines without compiling and testing whether the program does what it is supposed to do. Actually, the idea of testing is kept in mind throughout all activities because the earlier an error is found, the cheaper it is to fix it.

The testing activity consists of two parts at this point in the chain. First, the individual parts of a programmer or team are tested. If all parts pass the tests the entire system is tested. To test the entire system is crucial because it has to be checked whether the parts are compatible or not. Some people just do not see the need for testing the entire system and do not endorse the phrase that "the whole is greater than the sum of its parts" and therefore it needs additional attention. An ant colony might be a good example to support the previous phrase. Scientists have found out that an ant just follows four very simple rules in order to forage food. These rules are simple enough to build an artificial ant simulating ant's behaviour on a computer. When you put many ants together to an ant colony they forage food in a very effective manner and the behaviour of the whole ant colony is far more complex and sophisticated than the behaviour of a single ant. This example shows that the whole is greater and more different than the sum of its parts.

The results of this activity are logs of tests performed, which are an indicator of the quality of the software, but cannot prove that it is free of errors. To prove it error-free, all possible test cases related to all possible input data would have to be performed. Even for a very simple program it is impossible to carry out all test cases in a decent time. Thus, we must be aware that tests show the quality of the software but do not prove it error-free.

Operation: This is the ultimate goal of all previous activities, that is, to deliver a system that is satisfactory for operation. During this activity users might come up with new ideas and requirements which can be implemented or we might find unknown bugs to be fixed. Because of this behaviour this activity is also called maintenance.

Although software is intangible and does not wear off, it becomes less useful and more difficult to change with time. At some point the software is replaced or abandoned.

We have seen all five activities of the software development life cycle and the next section is going to give a brief overview of the most well-known development models supporting the software development process.

4.1.2. Software Development Models

The software development life cycle which was previously discussed is not the only model of how software can be developed. In fact, it is one of the easiest models but most of the others, we are going to introduce, are derived from it. As it is not the aim of this

paper to teach software engineering, the software development models are not covered in too much detail.

The Waterfall Model

The waterfall model is a variation of the software development life cycle. As Figure 4.2 on page 65 shows, it consists of the same activities. The difference to the software development life cycle is that returning and backtracking to a preceding activity is possible if it is really necessary. As mentioned earlier, testing should be applied at the end of almost every activity to be able to control the quality of the product. Thus, sometimes it is crucial to backtrack to an earlier activity to adjust the requirements or fix the design for example. This is indicated by the dotted arrows. The amount of backtracking must be kept as small as possible, though.

In spite of being able to return to previous stages, the emphasis is put on the careful analysis before starting to build the new program and the whole model is meant to be one way only. In other words, the usual way is downstream, hence the name waterfall model. This behaviour enables the software development process to be better controlled. For instance, the process is better time controlled because you know exactly where you currently are in the development process.

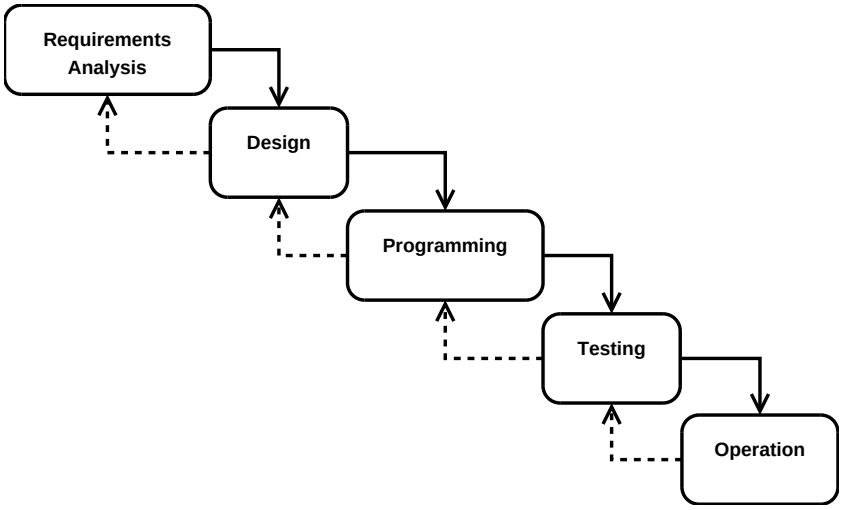


Figure 4.2.: Shows the waterfall model of software development

This model is called document-driven as each activity produces a lot of documents. They are carefully reviewed before proceeding to the next activity in order to reduce the risk to return back.

The waterfall model is a typical *top-down* model. The development starts off with generalisation and abstraction. It represents the system by means of documents and plans at a stage where no source code is actually written. The next steps go into more

detail until the concrete source code is delivered. It is known that the requirements analysis activity and design activity are very time consuming. Sometimes, especially for small projects, developers ask themselves whether it is worth investing time to write down the requirements and to design before coding. In some cases it is really more effective to skip these phases and to follow up *bottom-up* models. The prototyping model and the incremental model, both are bottom-up models, are going to be shown next.

The Prototyping Model

The prototyping model follows the approach of building a prototype rather than defining the requirements specification. There is no harm in putting the requirements specification activity aside as there is no use for it. It turned out that users very often do not know what they really want and what is possible. Thus, they are not able to contribute to the requirements specification document especially at this very early stage. Moreover, the requirements specification is often not clear enough to the users and they simply do not understand it.

Instead of wasting time agreeing to requirements this model builds prototypes to understand the users' needs. Prototypes are stripped down systems which do not provide the full functions due to the fact that time and costs have to be reduced to build it. A very common approach is to build just the user interface without implementing the processing logic. It is sufficient because this is exactly what users see and work with.

The system is built with the help of the users' feedback. The users can experiment with the prototypes and agree to or reject particular behaviour and features. Developers can modify the present prototype or even start a new one if it has to be redesigned. Finally, after some iterations, the processing logic is added and the software is ready for use.

This approach guarantees a high user satisfaction but also has disadvantages. Users feel involved in the development process and it is more likely to meet their needs with the final product. The drawback is that it is not known how many iterations are needed to finish the product and thus the development process can not be easily controlled.

For big projects, the prototyping model is used as a tool to get the requirements specification document. When the requirements are written down a top-down model is applied to build the software. If the prototyping model is used only, it is also called *evolutionary model*.

The Incremental Model

This model stands between the waterfall model and the prototyping model and combines their elements. It breaks up the whole system into pieces and delivers the basic piece as soon as possible. Unlike prototyping it exposes an operational product to the users instead of a prototype. After the first basic piece is delivered it can be tested by future users. They can give feedback to the developers telling whether the users' needs were understood or not. Taking this feedback into account, developers are able to improve

the product, add a new piece and expose a new version to the users. The waterfall model is applied to develop each software version.

This combination tackles problems of the waterfall and prototyping model. Using the former model, users do not see the system after completing the requirements analysis for a very long time until it is completed. Hence it follows that the users do not feel closely involved in the development and to work with the new product comes suddenly and means a big change.

Unlike the prototyping model the incremental model does not suffer from the absence of an overall plan and the lack of development control.

To understand how a system can be broken up into pieces or chunks, we look at an example of a word processing application. The first piece consists of the basic file management and functions for editing the document. The next software version might add other piece of features, that are, advanced editing and advanced page layout. Eventually, the final piece to add could be spell and grammar checking.

The Spiral Model

For the sake of completeness the spiral model is also mentioned. It represents the software process as a spiral rather than a sequence of activities, as shown above. The important distinction between the mentioned models and this model is the explicit consideration of risks. Furthermore, it encompasses other software process models where appropriate.

If you want to learn more about the spiral model, look at [Som01] on page 55 or at [vV93] on page 42.

4.1.3. Conclusion

It has been shown why we need software engineering and how it is defined. Software engineering emerged out of the desire to cope with the mounting complexity of software applications. This profession was introduced to support the creation and maintenance of software by means of technologies and methods proven suitable for this purpose.

As it has been mentioned, programming is an ingredient of the whole software development process but not the only one. It is crucial to structure and pay attention to all activities of the software development life cycle. For example, the requirements analysis and design activities are important to identify what has to be built and how should it be built. The approach to define the goal and plan the design before starting to write source code has one big advantage. It can reduce costs because the earlier a problem is solved the cheaper it is. It is obviously more expensive and time consuming to fix a problem after delivering the product.

Besides the software development life cycle other models to support the software development process have been introduced. That are, the waterfall, the prototyping, the incremental and the spiral model. They all have their advantages but also their disadvantages. However, an appropriate model or a mixture of models has to be chosen for every project.

In our case, which is the development of the Human-BPEL interface, the waterfall model is going to be used. This model is appropriate as we want to control the development process as well as being able to do testing throughout the process and backtracking to a previous activity if necessary. Because of the fact that the requirements were given and no users were involved the waterfall model was preferred to the prototyping model. Further, the project is rather small and the team of developers consists of one person only. Thus, it was intended to find the right mixture between planning and programming to prevent the planning activities from being too time consuming.

This closes the theoretical preparation for the software development and we are going to start off with the first activity.

4.2. Requirements

Let us recap the present situation before defining any requirements.

The whole web-based education system consists of three parts which are the WBT-Master, the Learning Process Manager and the Human-BPEL interface tool. The WBT-Master (see Section 3.2.1) provides a vast number of tools to support web-based education. For instance, such tools are user management, learning units, discussion forums, and questionnaires. As discussed, despite all these tools the WBT-Master system cannot deal with learning processes and their life cycle.

Therefore, the Learning Process Manager was introduced to manage learning processes. The main parts of the Learning Process Manager are the Learning Process Execution Engine and the Learning Process Definition Module (refer to Figure 3.7 on page 48). To put it simple, BPEL files are executed by the execution engine which communicates with the WBT-Master system and other tools by means of web services. We came to the conclusion that BPEL files do not provide enough abstraction and are too difficult to handle for normal users. The Human-BPEL interface tool shall tackle this problem.

Keeping this situation in mind, we are ready to specify the requirements of the tool. They are divided into three groups of requirements. These are the creation of a valid BPEL file, the interfaces and extensions, and the documentation.

4.2.1. Create a Valid BPEL File

The Human-BPEL interface tool has to create a valid BPEL file automatically from a very simple XML file (look at Figure 4.3 on page 69). The XML file must be defined and has to be precise, concise, simple and clear to understand. Moreover, the BPEL file must be suitable for the Learning Process Manager, this means, it has to be capable of being employed and executed immediately by the Learning Process Execution Engine.

The tool must support the following learning activities to be able to simulate a typical learning process:

- Reading activity

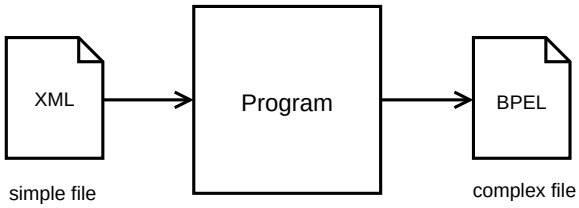


Figure 4.3.: Shows the basic idea of the Human-BPEL tool

- Discussion activity
- Test activity

Beside these activities it has to support at least one structured activity like sequencing, switching or a while loop.

All these requirements reflect the basic functionality of the Human-BPEL interface tool. The following two sections show the necessary additional requirements.

4.2.2. Interfaces and Extensions

It must be possible to extend the XML input file by new learning activities and structured activities. This means that the design and structure of the program have to be modular enough so that another programmer can easily extend the program's processing unit.

In addition, the tool has to provide a documentation mode generating a valid example of an XML input file. This file should help the program's users to write his or her own input file. It shows what activities are already implemented, how the correct syntax looks like, what XML tags and attributes exist, and what they are good for.

Another requirement is to develop the program in such a way that subsequent developments can use it to build a graphical user interface (GUI). It is planned to set up another project to create a GUI to define and change learning processes in a very convenient manner. This program must define an interface so that the development of the GUI will be able to be done separately.

Finally, the program needs to be capable of running on the command line because it is intended to run automatically in scripts or by other programs. Thus, it must take all necessary parameters for perfect operation through the command line.

4.2.3. Documentation

As this program is supposed to be developed further and extended by other programmers, an extra emphasis has to be given to documentation and how the source code is written. The code must be easy to read and contain comments where appropriate. Also debug information needs to be provided to help other programmers fix errors. Moreover, it has to be written in a consistent style and has to follow a code convention.

A tutorial must be written to assist other programmers in extending the program's processing unit by new activities. It has to show all source code files which have to be modified and all necessary steps which have to be taken. It should enable a person to extend the program without wasting time to understand the whole source code.

4.2.4. Overview

Completing the requirements activity, a brief list of all requirements is given below. This summary should help to keep the overview of all features that have to be implemented.

- Create a BPEL file from an XML input file suitable for the Learning Process Manager.
- Implement the reading, the discussion and the test activity besides of one structured activity.
- The program's processing unit and the XML input file have to be extendable by new learning activities and structured activities.
- Implement a documentation mode that is capable of creating an example XML input file.
- Take into account that further development intends to build a GUI on the top of this program.
- The program must be able to run on the command line.
- The source code must contain comments and follow a code convention.
- Write a tutorial how to extend the program's processing unit and the XML input file.

4.3. Design

At this stage of the development process we concentrate on the way the software can be built to meet the requirements. In order to be able to focus on designing and solving problems, abstraction mechanisms are used rather than any source code. Figures, descriptions and class diagrams are utilized for this purpose.

The next section describes the main idea of the design followed by sections going into more details about each part of the whole.

4.3.1. Main Idea

Create the BPEL File

As mentioned in the preceding section, the main goal is to create the very complex BPEL file used by the Learning Process Manager's execution engine from a very simple XML file which describes the learning process. For instance, a teacher wants to define the learning process for a university course dealing with the basics of the Linux operating system (see Section 3.3.3). Instead of changing or writing the BPEL file from scratch, we want to offer the teacher the possibility to define the workflow in a very simple XML file. Thus, the program has to transform the simple XML workflow file into a complex and deployable BPEL file.

To achieve this goal we can exploit the fact that the BPEL file consists of patterns that occur again and again related to the learning activities. After examining several BPEL files, specific patterns were noticed. For example, look at Listing A.5 on page 104. The assign tag from line number 32 to 49 and the assign tag from line number 58 to 75 only differ in some SOAP messages sent to the Worklist Manager. The remaining structure is totally the same. These patterns can be found everywhere in the BPEL file.

Figure 4.4 on page 71 shows the basic design of the program. The idea is to write a template file for each supported learning activity (e.g. reading, discussion, questionnaire, etc.). Each template only contains the information and BPEL structure that are related to its learning activity. To put it simple, for instance, the template file for the reading activity only holds the information that every reading activity has in common. The information which is different from every specific reading activity and is missing in the template file, must be provided in the XML input file (also called workflow file). Now, and as depicted in Figure 4.4 on page 71, the workflow file and the template files hold enough information to create the final BPEL file. This behaviour of the program will be called *normal mode* throughout this paper.

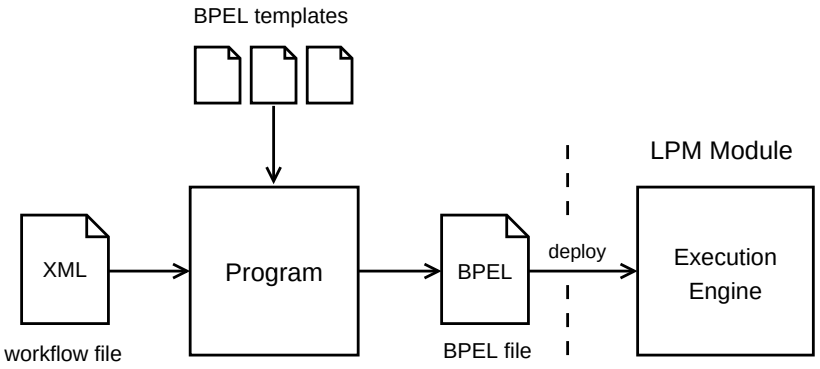


Figure 4.4.: Shows the program operating in normal mode

Documentation Mode

Figure 4.5 on page 72 shows the *documentation mode* producing an example input file suitable as an input file for the normal mode mentioned above. As stated in the requirements specification, a documentation mode is required to help the user write the workflow file.

A typical scenario looks like the following. A user runs the program's documentation mode before creating the desired workflow file. As a result, the user gets an example of a workflow file containing all present available learning activities with meaningful parameter values, the right syntax and comments. Then, the user can use this example to create the personal workflow file to obtain the final BPEL file by running the program in normal mode.

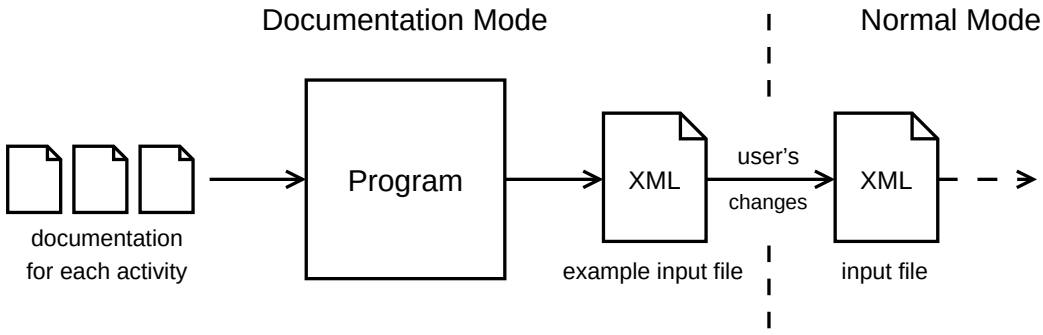


Figure 4.5.: Shows the program operating in documentation mode

For this purpose, documentation provided for each learning activity is processed and merged by the documentation mode. For the sake of the possibility to extend functionality in terms of new learning activities, an open documentation feature has to be designed.

This feature is meant to work hand in hand with the person who extends the program by a new learning activity. After extending a new activity, he or she must write the appropriate documentation. If it is missing, the program will terminate with a clear error to make sure that a documentation exists for each learning activity. This approach makes sense because the person who implements a new activity is supposed to know best what parameters are needed. If all necessary documentation files exist, the documentation mode merges them and produces the example workflow file (look at Figure 4.5 on page 72).

Extendable

It is crucial to be able to add new learning activities easily. Without this ability, this tool would be useless. Therefore, the program's design must support other developers by ensuring an easy way to extend the processing unit defining all learning activities.

The *Factory pattern* seems to be just the right design for this objective. We are going to have a closer look at this pattern later on (see Section 4.3.3).

Further, the XML input file shall also be the interface for the GUI development in the future. Over the recent years XML has established itself as a kind of de facto standard for exchanging data. It is already used for a wide range of applications and there will not be any harm in following this growing trend. In fact, the precise and strict syntax, the way data can be structured, and its flexibility make it a good choice to serve as the interface for the future GUI application.

Programming Language and Documentation

The whole program will be written in Java[1]. The main advantage compared to many other programming languages is the feature of its cross-platform binary portability. This means that the very same Java program will run on all operating systems like OS/2, Windows or Linux without being compiled again. In addition, Java is an object-oriented programming language and therefore supports our object-oriented design. As with almost all other programming languages, a Java program can be invoked by means of another script or the command line. This ability meets the program's specifications too.

Finally, we must not forget the documentation efforts required and stated in the requirements specification. To meet this need, *Javadoc*[2] will be used to generate HTML documentation for our classes. This will help other programmers to become familiar with the source code and smooth the way to further developments. Moreover, the Java Code Conventions will be applied to contribute to collaborating programming and software maintenance. It reflects the Java coding standards presented in the Java Language Specification and covers everything regarding file names, file organisation, intendation, comments, declarations, statements, naming conventions and white spaces [KND+97].

4.3.2. The Java Classes

In the following, the main Java classes are described and how they interact with each other. First, class diagrams are introduced utilized to show the design of the program. This is followed by the explanation of classes' functions and how the classes are connected in order to fulfil the given tasks.

Before starting right off with any class diagrams, there is one more thing to think about. The name of the program has to be chosen. Let us take something short but meaningful. The name shall be *BuildBPEL*.

[1] http://java.sun.com
[2] http://java.sun.com/j2se/javadoc/

Class Diagrams

Class diagrams are a very helpful tool in the object-oriented design process. They support the developer to identify and show the classes and the objects that will make up the program.

Figure 4.7 on page 76 represents the class diagram for the BuildBPEL class. As you can see, the class diagram is divided into three areas. The first area at the top of it holds the name of the class. All attributes of the class are written in the area below and the methods of the class are stated in the lowest area.

The syntax of the attributes is defined by Definition 4.1 where *attribute* is the name of the attribute, *type* is its data type and *initialValue* is its initial value. Except the *attribute* everything else can be omitted, as for instance, the *initialValue* that was not needed in any class diagram in this chapter.

$$\text{attribute} : \text{type} = \text{initialValue} \tag{4.1}$$

Definition 4.2 shows the syntax for the methods where *method* is the name of the method, *returnType* is the data type returned by the method and *argumentList* is an optional list of arguments. The list of arguments consists of entries separated by commas where each entry taking the same form as the attributes defined above.

$$\text{returnType method (argumentList)} \tag{4.2}$$

Furthermore, the class diagram gives more detailed information by prefixing the attributes and methods. Different symbols can be spotted in front of every record with the following meaning:

♯ stands for a protected attribute or method

− stands for a private attribute or method

\+ stands for a public attribute or method

Now we know how to read class diagrams and we are moving on to understand the design of the main classes and how they interact with each other. The classes are divided into the main class and document classes representing the program's core, and the sequence classes implementing learning activities.

Main Class and Document Classes

The design of the classes is derived from Figure 4.4 on page 71 and Figure 4.5 on page 72. When you look at these two figures, you can see that you do not have to look for a design as it is already given. In fact, each part of the figure represents a separate object and its function. Thus, the approach to model each object as a class would be appropriate where four objects can be identified, the workflow input file, the program

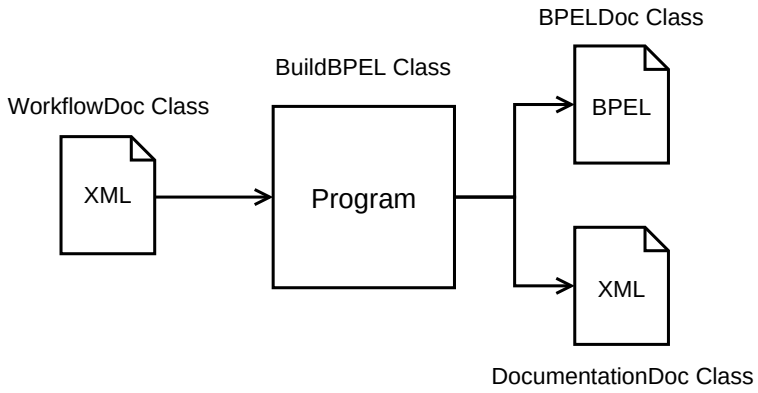


Figure 4.6.: Shows the main classes where the normal and documentation mode are combined

itself, the BPEL output file and the example input file when operating in documentation mode.

Turning these objects into classes, the following classes can be defined (look at Figure 4.6 on page 75):

BuildBPEL: This class models the BuildBPEL tool and holds the *main()* method. It is the starting point for every operation. As it is depicted in Figure 4.7 on page 76, it contains all parameters of the tool required to operate properly. These necessary parameters are, for instance, the name and path of the input file (also called workflow file) and the output file (BPEL file). Also, the directories of the template files and the documentation files have to be provided. Last but not least, attributes, that is whether to enable debugging or to run the program in documentation mode have to be set. Default values for all these parameters are defined but they can be overwritten when invoking the program.

WorkflowDoc: The WorkflowDoc class (see Figure 4.8 on page 77) represents the workflow file that is the input file. Besides storing the workflow file (the data type is *Document*), it implements all legal operations such as printing out the whole file for logging purposes and maintaining a list of all learning activities defined in the user's workflow file.

BPELDoc: The basic structure of this class is similar to the WorkflowDoc class. It also represents an XML file since BPEL is written in XML. In contrast to the WorkflowDoc class which only reads workflow files, it provides methods to build skeleton BPEL files from scratch and to add learning activities. Looking at Figure 4.9 on page 78, the job of the *buildHeader* method is to generate a skeleton or frame BPEL file while the *addSequence* method takes care of adding learning activities to it. The name *addSequence* was chosen rather than *addActivities* because in

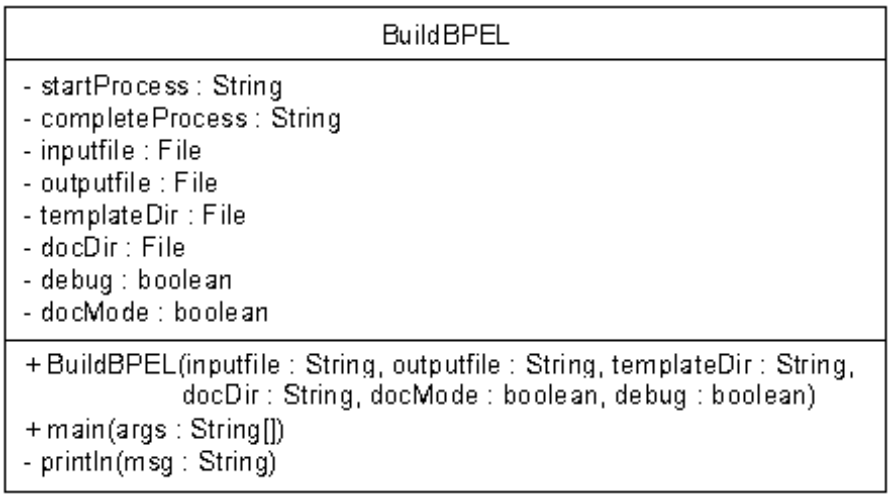


Figure 4.7.: Shows the BuildBPEL class

BPEL a process consists of sequence tags. Furthermore, as easily spotted in the class diagram, the *saveDoc* method saves the BPEL document to the designated output file.

DocumentationDoc: The DocumentationDoc class (see Figure 4.10 on page 79) is the last document class. It processes all documentations of activities and puts them together to one example input file (refer to Figure 4.5 on page 72). The methods *generateDoc* and *saveDoc* contribute to this task.

More details about all classes, their attributes and methods come with the program. This brief overview of the main class and documentation classes is enough to get the gist of the design. A detailed description of all classes, attributes and methods is available through JavaDoc documentation.

Putting all classes together

We have seen four classes building the program's core but how are they meant to work together? The following sequence shows how the program works in normal mode:

1. The program is invoked with parameters by the user. The values of the parameters or, in case the user does not want to change them, default values are stored in the *BuildBPEL* object. This object also holds the main method and is the starting point of the program.

2. A *WorkflowDoc* object is created reading the workflow file (input file).

3. The skeleton BPEL output file is built by creating the *BPELDoc* object.

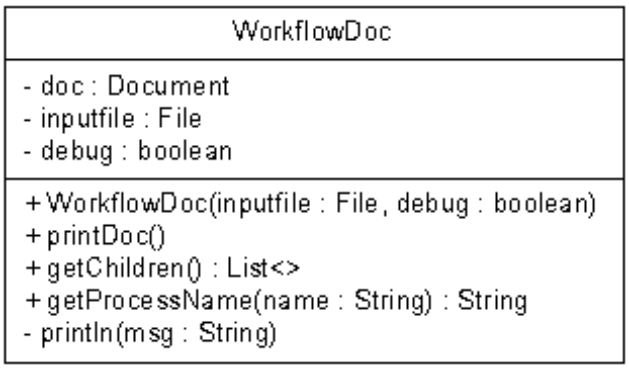


Figure 4.8.: Shows the WorkflowDoc class

4. The *main* method obtains a list of activities provided by the *WorkflowDoc* object. Each activity's name is passed on to the *BPELDoc* object by means of a loop in order to add new sequences to the skeleton BPEL document.

5. The last step is to save the BPEL document held by *BPELDoc* to the output file.

These steps describe the normal mode which is very different to documentation mode. In documentation mode the user invokes the program as before but has to turn on the documentation mode in the parameter's list. Only for this mode, necessary parameters will be taken into account. Next, a *DocumentationDoc* object is created parsing and processing the documentation files for each implemented learning activity. This operation builds the example workflow file that is finally saved to the output file.

We have seen how the normal mode and documentation mode are planned to work but the manner in which the *BPELDoc* object adds new sequences has been kept as a secret yet. In addition, in previous sections attention was drawn to the fact that learning activities must be able to be extended easily. Thus, the next section is going to describe both, the design which enables other programmers to add new learning activities without much effort, and how the *BPELDoc* adds new sequences.

4.3.3. The Factory Method Pattern

Figure 4.11 on page 79 shows the Factory Method pattern sometimes also called Virtual Constructor. Looking at the general form, you can see that the Creator and Product class are abstract classes or interfaces because of their Italic font. ConcreteProduct and ConcreteCreator are classes implementing all necessary methods.

The Factory Method allows its client objects to work only with the interface defined by the abstract Product class or interface. This keeps the complexity in all the ConcreteProduct classes. To put it simple, the Product acts as an interface defining all legal operations that a ConcreteProduct must at least be able to perform. This design makes

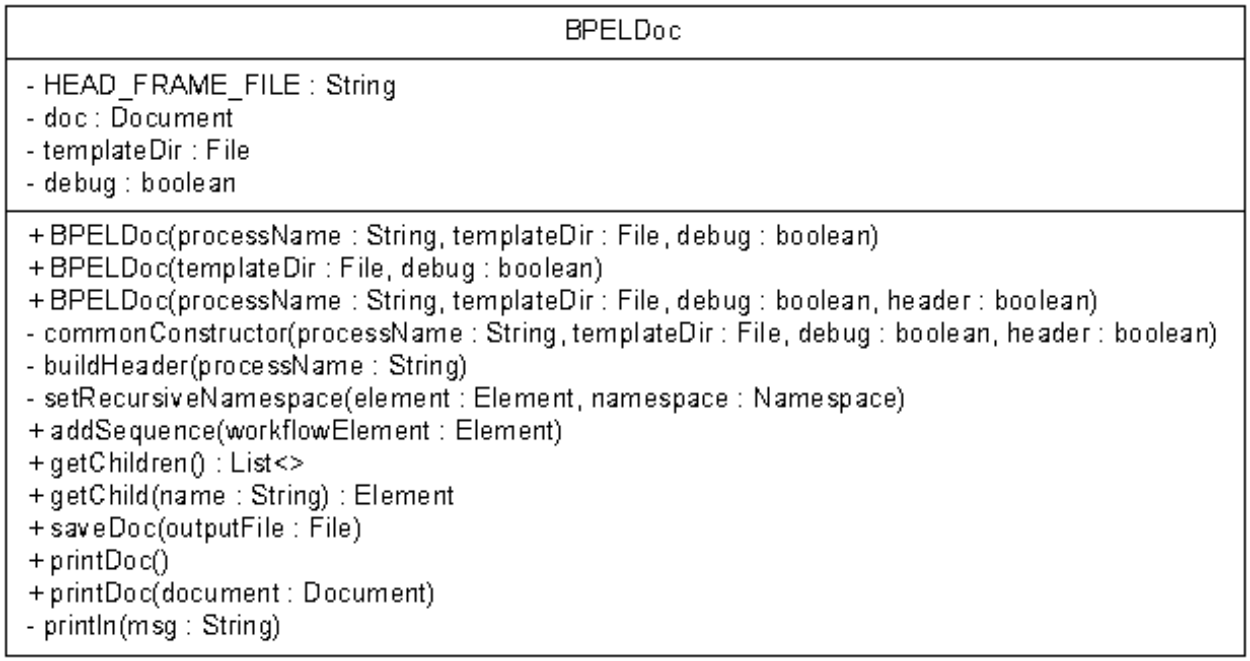


Figure 4.9.: Shows the BPELDoc class

sure that every particular product (ConcreteProduct) takes care of its implementation itself. At the same time the client objects using specific products do not have to know which product they work with as their public methods are the same. This design is commonly known as inheritance or implementing interfaces.

In addition, the Factory Method pattern supplies the client objects with Creator classes to be completely independent of the product line. The *factoryMethod()* implemented by a ConcreteCreator class returns a specific product which implements all public methods defined by the abstract Product class or interface. That means that all client objects using the *factoryMethod()* do not have to bother about which specific product is returned and their source code can remain untouched when new concrete products are added to the product line.

Let us move on to our specific implementation of the Factory Method pattern as depicted on the right side of Figure 4.11 on page 79. You can probably identify the only difference to the general form. The abstract Creator is missing due to the situation that no creators are needed but one. In our case, the concrete creator is the SeqeuenceFactory. It implements the *getSequence()* method returning a specific Sequence like ReadingSequence, DiscussionSequence, TestSequence etc.. As described above, this design enables programmers to extend the sequences without touching the source code of the actual program's logic.

With this knowledge in mind we can describe the sequence classes which were left

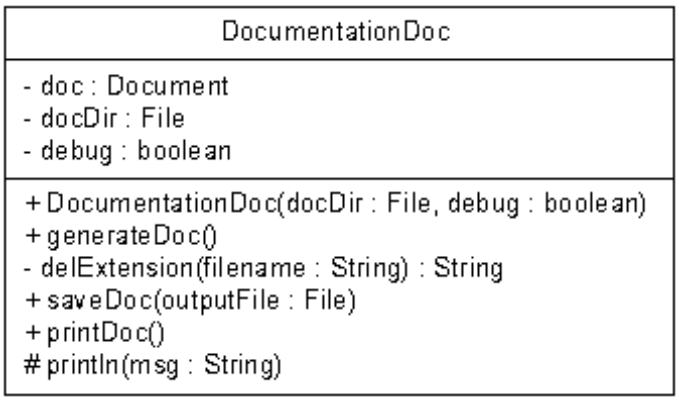


Figure 4.10.: Shows the DocumentationDoc class

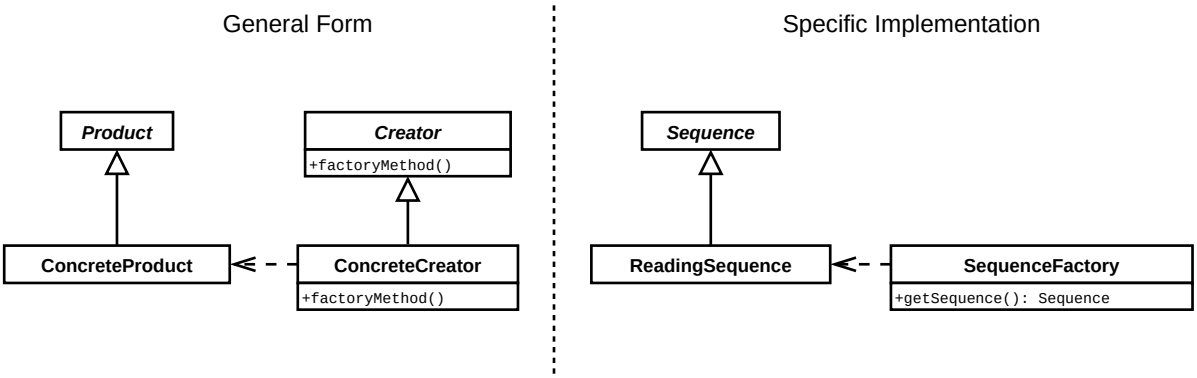


Figure 4.11.: Shows the Factory Method pattern. The general form on the left is compared to the specific implementation on the right

unmentioned in the previous section.

Sequence Classes

The following classes deal with learning activities and are designed with respect to the Factory Method pattern. They enable programmers to add, delete and modify learning activities without having to revise the program's main code.

Do not be confused by the words activity and sequence. They have the same meaning in this paper. We are talking about a reading activity when the term reading sequence is used.

Sequence: It is an abstract class defining methods every sequence must provide (see Figure 4.12 on page 80). It also comes with helper methods to assist programmers when writing a particular new sequence class, that is, for example, a reading

sequence class. To be more accurate, helper methods exist to alter the text and attributes in XML documents.

The abstract class *getAllElements()* has to be implemented by particular sequences themselves. This means that each concrete sequence, for example, the reading sequence, takes care of generating valid BPEL sequences by using the template file and the information provided by the workflow file.

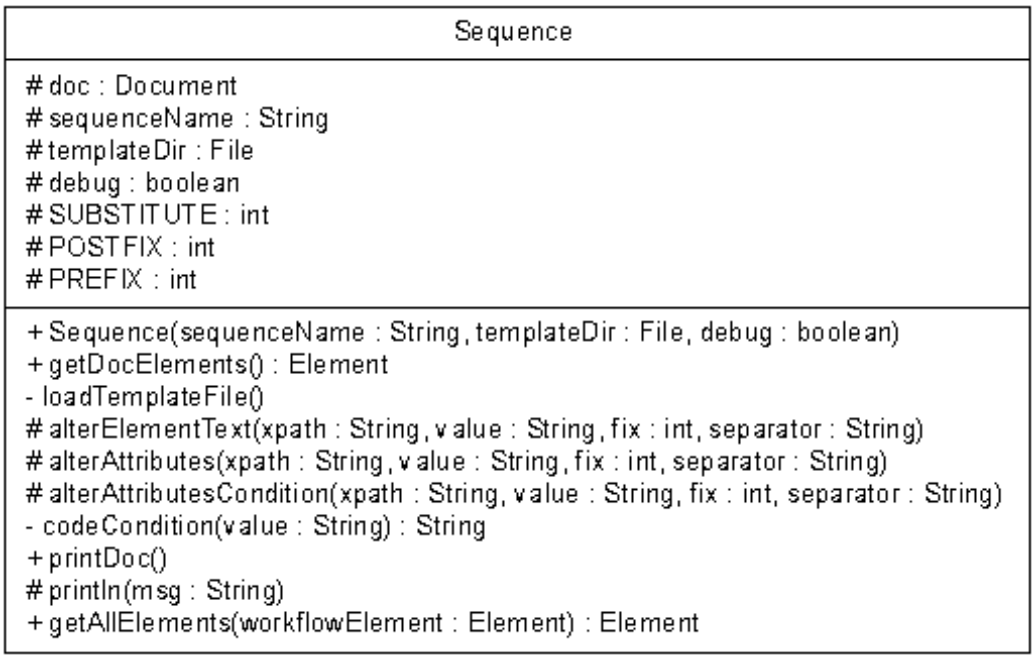


Figure 4.12.: Shows the Sequence class

SequenceFactory: As it can be seen in Figure 4.13 on page 80, it is a typical concrete factory class. Besides the constructor, there is just another method (*getSequence()* returning a specific sequence according to the sequence name passed on.

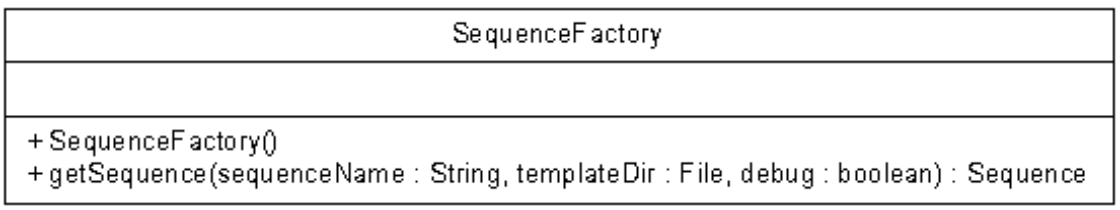


Figure 4.13.: Shows the SequenceFactory class

ReadingSequence: This class is just one example for a particular sequence class. It was mentioned before that only the *getAllElements()* method has to be implemented beside of the constructor of course (look at Figure 4.14 on page 81). This method alters the template file by applying helper methods and taking information provided in the workflow file into account. It returns the modified template file as an XML document to the client's object which uses this chunk of information to build the whole BPEL file.

All sequence classes work likewise and there is no limit to add new learning sequences easily.

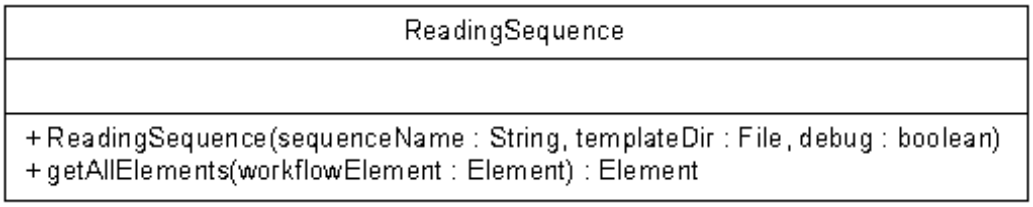


Figure 4.14.: Shows the ReadingSequence class

In this section the Factory Method pattern has been explained and all related classes have been described. We know the design of the classes and how they are planned to work. The next two sections are coming up with stuff that has to be defined and planned before starting to write source code. We are going to look at all used file formats and parameters the program should take.

4.3.4. File Formats

The program has to work with three types of files, these are, the workflow input file, the template files and documentation files for each learning activity. The workflow file is the only file that is created by the user. All other files are written by programmers when adding a new learning activity.

The Workflow Input File

This file is the only file which is visible to the user and therefore has to be as simple as possible. It is the right time to mention again why we are developing this program. Earlier, we came to the conclusion that writing the BPEL file needed by the Learning Process Manager is too difficult for a normal user. We wanted to find a solution with a higher level of abstraction. Now this workflow file is the interface to the ordinary user. It must be as simple as possible and at the same time it must provide all information that is necessary to generate the rather complex BPEL file.

Listing 4.1 on page 82 shows how a short workflow input file looks like. We have to follow the syntax of XML as it is an XML file. Every input file starts with a process tag and an attribute name to label the process. The process element is the root element and holds all learning activities. In this example a reading activity is followed by another reading and discussion activity. Each activity has to define a unique name to identify itself (i.e. reading_1, reading_2 and discussion_1). In addition, each element defining an activity consists of further parameters that are required for proper operation. The reading activity, for instance, just needs the URL of the reading resource, its title and a message for the user conveyed by the Worklist Manager.

Listing 4.1: Simple workflow input file

```
<process name="Linux_Operating_System">
 <reading name="reading_1">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/
     linux1_start0.htm</url>
  <title>Introducing Linux</title>
  <message>First read this!!</message>
 </reading>
 <reading name="reading_2">
  <url>http://www.linux.org/info/index.html</url>
  <title>A short introduction to Linux by its developers</
     title>
  <message>It will only take 60 minutes.</message>
 </reading>
 <discussion name="discussion_1">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/lb_forum.
     htm</url>
  <title>Discussion about the Linux basics</title>
 </discussion>
</process>
```

The parameters for the reading, discussion, testing and testmarking activities have already been defined. When there is a need to extend the program by a new learning activity, the necessary parameters have to be found out and a documentation file for this new activity has to be written. It is vital to provide it as the user must know which parameters are needed. You can read more about documentation files a bit later.

If you want to see a more complex workflow, look at Listing A.2 on page 101 which was used to describe our process example in previous chapters.

Template Files

Each learning activity has got its own template file consisting of BPEL code lines. It was mentioned earlier that these template files contain BPEL patterns that are used again and again. Look at Listing A.3 on page 102 to see what all reading activities have

in common. You might find out quickly that the only differences between this snip and the final BPEL code (see Listing A.5 on page 104) are the SOAP messages. In case of the reading activity, we are talking about the URL, the title and a message. Can you see the link between the parameters of the reading activity in the workflow file and the reading template file? Now it should be obvious why the workflow file had to provide this information. Simply because it is missing in the template file.

Documentation Files

Listing 4.2 on page 83 shows the documentation file for the reading activity. Besides the right syntax, all available parameters such as the URL, the title and a message are documented. According to the program's design, it does not work if a documentation file is missing. This ensures that the user knows all already implemented learning activities and their parameters.

Listing 4.2: Documentation file for the reading activity

```
<process>
 <!-- unique name for the activity -->
 <reading name="">
  <!-- url to the reading resource -->
  <url></url>
  <!-- the title of the reading -->
  <title></title>
  <!-- message for the users -->
  <message></message>
 </reading>
</process>
```

For both, documentation files and template files, the file names are the same as the related learning activities in the workflow file (e.g. reading, discussion, etc.).

4.3.5. Program's Parameters

The last step before writing the program is to think about the program's parameters. What information does the program need when being invoked on the command line?

In normal mode the following is needed:

- The path and name of the input XML file to be processed
- The same information about the output BPEL file
- The directory of the XML template files

In contrast to normal mode the documentation mode is different. This mode works without input file and therefore the list of parameters looks as follows:

- Path and name of the desired output file
- The directory of the documentation files

Finally, a parameter to switch between the two modes and to turn on or off debugging information is going to be added as well.

After developing the main idea and design, defining the classes, describing all file formats and set the program's parameters we are ready to start writing code. The designing activity took some time but now we can focus on the programming activity without thinking about the problem to be solved.

4.4. Programming

This stage of the software development process is to produce source code and an executable program. We must keep in mind that the result of this activity must be an easy to read and well-documented source code as well as an executable program meeting the needs of our requirements. The preceding design activity helps to obtain good results in decent time.

This chapter shows what was required to write the program and mentions extraordinary implementations. First, the whole programming environment is looked at. Then, we pay attention to external Application Programming Interfaces (APIs) used besides the libraries of the Java programming language. Finally, the recursive design of structured activities is discussed as an example for a remarkable implementation.

Let us just start off with the programming environment as it was also the first step to do before writing the first line of source code.

4.4.1. Programming Environment

In order to choose the right programming environment we need to know and take into account what we have to produce. Thanks to the previous stages of the software development process, some facts are already given. Besides the design we know that the Java programming language shall be used. Furthermore, the Java Code Conventions must be applied and also a JavaDoc has to be generated. In addition, we want to automate the compilation process as much as possible and want to use a revision control for the whole project. Thus, we were looking for an integrated development environment (IDE) that meets these needs.

An IDE usually refers to a type of software that helps programmers to develop software. It normally consists of a source editor, a compiler, automation tools to build the executable, a debugger and sometimes a revision control. Many developers might think about Eclipse[3], Microsoft Visual Studio[4] and many others now. They are decent

[3] http://www.eclipse.org/
[4] http://msdn.microsoft.com/vstudio/

products but almost all of them lack effective text editing. Although a lot of other activities have to be done during the programming activity, text editing includes writing, changing, moving around and searching is still the main task and has to be done very effectively. This was the reason why Vim was chosen. Read *Seven Habits of Effective Text Editing*[5] if you are curious about a short introduction.

"Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems"[6]. Vim was not meant to be a fully fledged IDE but with a few efforts it is possible to set up a very effective development environment.

The following list describes the tools used to set it up:

Vim: Vim is the core program and the editor of the development environment. It utilizes other tools to get specific tasks done. Although it uses them, the user does not have to leave Vim. All actions can be started within Vim and also the results are displayed there. This behaviour is exactly the same as for an IDE.

Further, Vim's functionality can be easily extended by plugins. For instance, taglist and buffer explorer plugins are used to visualize and jump between methods, attributes, classes and all project files.

We are not going into much more details about Vim's features. The last thing to mention is that it has a huge amount of them which are not even covered by all existing books dedicated to Vim.

Java: Sun's Java SE Development Kit 1.5 (JDK)[7] is used as a compiler and an interpreter. Both, compiler and interpreter, are invoked by Apache Ant that is discussed next.

Apache Ant: Apache Ant[8] is a Java-based build tool. It is like GNU Make[9] but without its limitations. As Make is shell-based you limit yourself to the operating system. The author of Ant wanted to develop across multiple platforms. Thus, in contrast to Make, Ant is based on Java classes so that it can run on all operating systems. Additionally, its configuration files are based on XML. In summary, Ant is cross-platform and very fast when it comes to compiling many Java classes but it has not as much expressive power as Make that is based on shell commands.

Ant is used to automate development tasks such as, compiling, executing, testing, producing JavaDoc, creating distributions and running the revision control.

Subversion: Subversion[10] is a version control system or also called revision control system. It is free, open source and meant to be a modern replacement for CVS[11]

[5] http://www.moolenaar.net/habits.html
[6] http://www.vim.org/
[7] http://java.sun.com/javase/
[8] http://ant.apache.org/
[9] http://www.gnu.org/software/make/
[10] http://subversion.tigris.org/
[11] http://www.nongnu.org/cvs/

(Concurrent Versions System). Both manage files and directories over time and keep track of all changes ever made to them. This ability allows the users to recover older versions and examine the history of how data has changed with time.

Such a revision control system is typically used for the implementation of software projects as it enables several programmers to collaborate. Everyone has access to every version of every file in the project at any time and everywhere the system can be accessed.

All these tools make up a very decent development environment supporting the programming activity to a great extend.

4.4.2. External Application Programming Interfaces

Beside of the core Application Programming Interfaces (APIs) included in the Java Development Kit, JDOM[12] and Apache Xerces[13] were used.

JDOM

Java Document Object Model (JDOM) is an open source API for reading, writing and manipulating XML from within Java code. It represents an XML document to Java programmers in a very straightforward and intuitive manner. It, for instance, uses Java collections that are well known and therefore behaves like Java as it would be expected. Users do not have to be experts in XML to get their work done. For all these reasons, JDOM was preferred to SAX[14] (Simple API for XML) and DOM[15] (Document Object Model).

JDOM interoperates with SAX and DOM. It can read from and write to SAX and DOM sources. This makes it possible to reuse existing program components implementing another XML document representation.

A JDOM's philosophy is that it should be fast and lightweight. JDOM operates more quickly than DOM and as quickly as SAX, despite the fact that it has many more features [HM].

Xerces

Xerces is an open source XML parser developed by the Apache XML Project. It is widely used by the XML community and is also the standard parser for JDOM. This validating XML parser enables your program to read and write XML data. In our case, JDOM uses it when reading an XML file or writing the XML document to an output file.

[12]http://www.jdom.org/
[13]http://xerces.apache.org/
[14]http://www.saxproject.org/
[15]http://www.w3.org/DOM/

Over the recent years several implementations of Xerces have been developed. Now, there are versions written in C++ (Xerces-C), Java (Xerces-J1 and Xerces-J2) and in Perl (Xerces-P).

4.4.3. Remarkable Implementation

At the end of this chapter we want to look at a part of the program's implementation that is more remarkable and maybe more difficult to understand than the others. We already know how learning activities like, for example, reading activities are implemented but structured activities, such as the basic ones, sequence, switch and while, have not been mentioned yet. In fact, one structured activity, that is *while*, was implemented related to the requirements specification.

It is treated like any other activity but uses an own BPEL document object to overcome its recursive nature. The difference between the while activity and all the other activities, we have heard of so far, is that it can embrace all activities too. This also means that the while activity cannot only consist of many other learning activities, it can contain other while constructs too. Therefore, it has recursive nature and every level of recursion can be thought of as a separate BPEL document object with its activities. This is exactly what the implementation does.

The while activity uses a stripped BPELDoc class to face the recursion. As we know, the BPELDoc object generates the BPEL file by means of putting BPEL headers and BPEL sequences generated by the Sequence objects together. The stripped BPELDoc class does not add the BPEL header and as a result it returns only sequences as all Sequence classes do. In other words, this stripped class can be used to implement a while loop embracing other learning activities and even other loops while returning the same structure of information as every learning activity does.

Figure 4.15 on page 88 depicts an example. It shows fictitious representations of BPEL documents. The BPELDoc object, as described before, builds the BPEL header and obtains the BPEL source code for the learning activities through the methods of the implemented Sequence classes (i.e. ReadingSequence, DiscussionSequence, WhileSequence, etc.). Actually, the while loop is also an implemented Sequence class but as you can see it uses the stripped BPELDoc class to build the BPEL document for this loop. After finishing the loop, which can include other while loops, the BPEL elements are returned.

This implementation example ends the programming activity of the software development process and we are moving on to the testing activity.

4.5. Testing

This section describes the efforts which were made to test the quality of the new software. Human testing methods are applied and the software is tested in a real-life environment. It is also checked whether all requirements were met by the developed software.

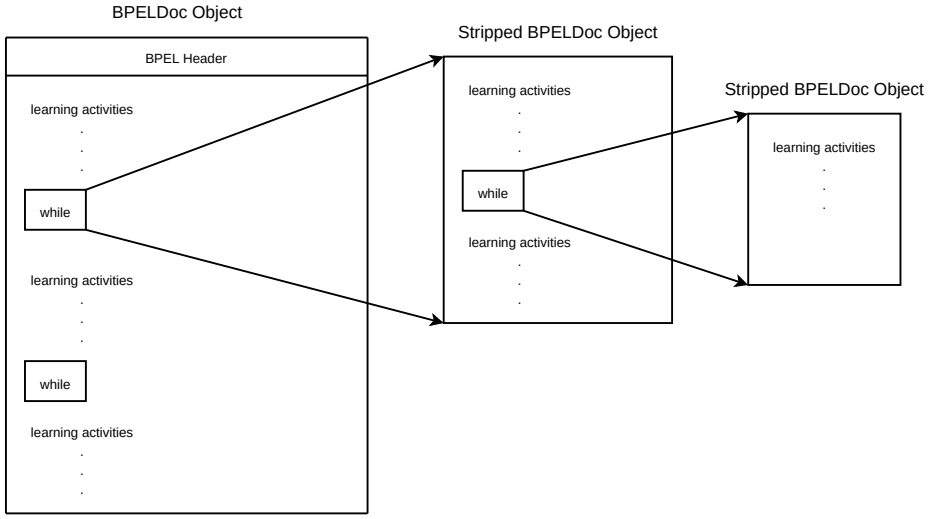


Figure 4.15.: Shows the recursive implementation of the while activity

As suggested under the Structured Development Life Cycle (see Section 4.1.1), testing was done during the whole developing process to find and fix errors as soon as possible. For example, after adding a new functional unit in the programming activity, the code was compiled and executed to check whether it comes up with the expected results.

At this stage we are testing the quality of the software before it is shipped out. We are examining the behaviour of the whole rather than its parts and want to find out whether the software works as required and expected.

To succeed during this phase, the attitude to testing is significant. One primary reason of poor program testing is the fact that programmers see testing as a process of demonstrating that errors are *not* present or to show that the program performs correctly. This is definitely not the correct approach to find errors. A more appropriate definition is [Mye04]: *Testing is the process of executing a program with the intent of finding errors.* Keeping this definition in mind, let us walk through all tests.

4.5.1. Code Inspection

Code inspection is a human testing method to revise source code. Nowadays it is widely accepted that to execute a program on a machine is not the only way to test it. During a code inspection the programmer goes through every statement thinking about its correctness and its logic. In addition, the source code is inspected with reference to a checklist of historically common programming errors [Mye04].

The big advantage of this method is that if you find an error you know exactly where you have to fix it. If you find errors testing the program by executing it, you have to find out first which code lines cause problems. During a code inspection you identify a statement, which will cause an error, and are able to fix it right there. This can save a

lot of time. Thus, it is worth inspecting the code carefully.

4.5.2. Testing in Real Environment

Besides inspecting the source code, the new software (i.e. BuildBPEL) was tested in the following real environment by means of real-life scenarios. Figure 4.16 on page 89 represents the architecture of our environment.

You might want to take a look at the architecture of the Learning Process Manager (LPM) again (see Figure 3.7 on page 48) to see how they are related. The Active BPEL Engine is LPM's Learning Process Execution Engine, WBT-Master is the E-Learning System and the BuildBPEL tool is part of LPM's Learning Process Definition Module.

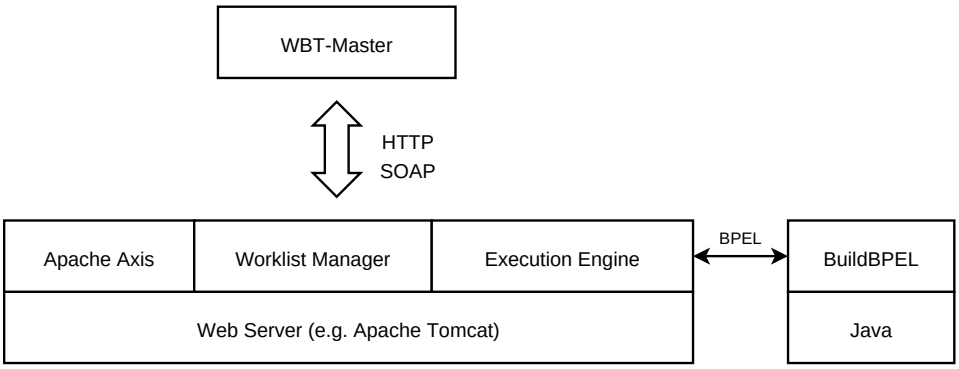


Figure 4.16.: Shows the architecture of the real testing environment

Let us now explain the parts of the environment:

Apache Tomcat: Apache Tomcat[16] is used to act as a web container for Apache Axis, the Worklist Manager and the BPEL engine. Its installation and configuration is very easy and related documentation can be found on Tomcat's home page.

Apache Axis: Apache Axis[17] is an open source Web service framework implementing the SOAP server and several utilities and API's for generating and deploying Web service applications. Utilizing it, developers can expose Java code as Web services very easily. This is exactly what it is used for. Almost the whole communication of the system is carried out by means of Web services, HTTP and SOAP messages.

Besides the Java version of Axis, which is used in this case, a C++ implementation is also available.

Worklist Manager: Section 3.4.4 described it in detail but basically it keeps track of all working and learning tasks for all users and for all processes running in the execution engine.

[16] http://tomcat.apache.org/
[17] http://ws.apache.org/axis/

Execution Engine: It is responsible for the execution of learning processes defined by BPEL. The current implementation of it is an open source BPEL engine called ActiveBPEL[18]. For more details about the execution engine go back to Section 3.4.4.

BuildBPEL: It is the tool we have been developing the whole chapter. As we know, it is written in Java and creates BPEL files, suitable for the BPEL execution engine, out of simple workflow files.

WBT-Master: The WBT-Master acts as the e-learning system described in Section 3.2.1.

The tests were carried out by means of different but typical workflow files. First, each of these workflow files was converted into BPEL through the new BuildBPEL tool and deployed in the execution engine. Finally, instances of Web browsers were used to connect to the Worklist Manager to prove the correctness of the learning processes.

After these tests, we can be sure to deliver a correct working program for learning scenarios it is very likely to be used for.

4.5.3. Check Against Requirements Specification

The final step in the testing activity is to check whether the program meets the requirements specified earlier in this development process. The probably easiest way to compare the requirements and what has actually been implemented is to describe the final product with reference to the specification overview provided in Section 4.2.4.

- The main function of the program is to create a BPEL, which is suitable for the Learning Process Manager, out of a simple XML workflow file. The generated BPEL file has been successfully tested to determine whether it works with the Learning Process Manager.
- The reading (ReadingSequence class), the discussion (DiscussionSequence class) and the test activities (TestSequence class and TestmarkingSequence class) have been implemented besides one structured activity (i.e. while).
- The use of object-oriented programming and the Factory Method pattern guarantee that other learning activities and structured activities can be added to the program's processing unit and the input file.
- The program comes with a documentation mode being able to generate an example XML workflow file. This file contains all implemented learning and structured activities, their comments and the right syntax.

[18]http://www.activebpel.org/

- The workflow file shall also act as the data format for further developments that might be a GUI, for example.
- The program is written in Java and due to the fact that all needed parameters are passed through the command line it can be used in automated scripts as well.
- The source code not only contains of a lot of comments, JavaDoc has also been generated. Also, Java Code Conventions was applied throughout the whole program.
- A tutorial, which is dedicated to how functionality can be added, has been written. It can be found later in this paper (look at Section 4.7).

Apparently the developed program meets the requirements and it has been worth putting so much effort and time into the project. Now let us move on to the last activity in the development process, the operation.

4.6. Operation

This section is dedicated to all users and explains the usage of the program. It is divided into the two operation modes. For each mode all parameters will be given and examples will be used to make their meaning clear.

In contrast to many other programs, being invoked on the command line, this program defines default values for every parameter. This means that if the user is satisfied with the default values, he or she can run the program without passing any arguments.

Now we will have a closer look at the normal mode.

4.6.1. Normal Mode

This mode generates a BPEL file from an XML workflow file. For this purpose, the program needs the path to the input and output file. Further, it has to know where the template files are located.

The best way to get to know the program is to run it. As it is written in Java, you will need at least a working Java Runtime Environment. Java version 1.5.0 is used for the time being. If you are using the distribution coming with this paper, you can simply run `run.bat` on most Microsoft Windows operating systems or issue the following command on the command line:

```
java -classpath lib/buildbpel.jar;lib/ net.go2fidelis.buildbpel.BuildBPEL
```

If `Java` is installed and in your PATH environment variable you must see this output:

```
Running BuildBPEL with following new values:
{}
Listing of the default values:
{template-dir=templates/, debug=off, output=output.xml, input=input.xml,
documentation-dir=doc/, documentation-mode=off}
```

The output above shows you all values of all existing parameters. `BuildBPEL` was not invoked with any parameters. Thus, all default values were used and no new values were taken into account. In fact, the program has already generated a BPEL file. This is possible due to the fact that this distribution comes with the necessary directory structure and files that suit the default values. Consequently, an XML workflow file, named `input.xml`, was taken to generate the BPEL output file with the name `output.xml`. You can find these files when you look into the directory where the program was invoked from. You can also check out Listing A.5 on page 104 and Listing A.2 on page 101 to see the content of these files.

It is time to explain the parameters in more detail although most of them are pretty self-explaining.

input: Defines the absolute or relative path to the workflow input file. The existence of this file is compulsory in this mode. Default value: `input.xml`

output: Sets the absolute or relative path and name where the BPEL output file shall be written. An existing directory structure and sufficient rights to save the file are compulsory. Default value: `output.xml`

template-dir: Defines the directory where all template files are stored. A wrong path or missing template files will result in termination of the program. Default value: `templates/`

debug: Switches debug messages on and off. Valid values: `on/off`. Default value: `off`

The last two parameters have not been described as they do not have any importance in this mode. Changing `documentation-mode` to `on`, certainly changes the mode and effects the operation, though.

Example

This example should illustrate the use of individual parameter values. All parameters, except for the template directory, are set by the user:

```
#java -cp lib/buildbpel.jar;lib/ net.go2fidelis.buildbpel.BuildBPEL \
--input=linux_course.xml --output=process.bpel --debug=on

Running BuildBPEL with following new values:
{debug=on, output=process.bpel, input=linux_course.xml}
```

```
Listing of the default values:
{template-dir=templates/, debug=off, output=output.xml, input=input.xml,
documentation-dir=doc/, documentation-mode=off}
```

So much for the normal mode. The documentation mode is very similar.

4.6.2. Documentation Mode

This mode generates an example workflow file suitable as input file in normal mode. The program does not need any other files beside of the documentation files in order to fulfil this task.

Again, the best way to explain the function is to look at an example. The invocation of the following command generates a skeleton workflow file (see an example at Listing A.4 on page 103):

```
java -cp lib/buildbpel.jar;lib/ net.go2fidelis.buildbpel.BuildBPEL \
--documentation-mode=on
```

Running this command, the already familiar output can be seen on the screen:

```
Running BuildBPEL with following new values:
{documentation-mode=on}
Listing of the default values:
{template-dir=templates/, debug=off, output=output.xml, input=input.xml,
documentation-dir=doc/, documentation-mode=off}
```

Now the documentation file has been written. You can use any editor to check the content of `output.xml`. Since we did not change the value of the output parameter, the default value (i.e. output.xml) was used.

In this mode the two parameters `template-dir` and `input` are not used. Thus, their values are of no interest. All other parameters have the following meaning:

documentation-mode: Switches between normal and documentation mode. Valid values: `on/off`. Default value: `off`

output: Defines the absolute or relative path and name where the generated output shall be written. Default value: `output.xml`

documentation-dir: Defines the directory where all documentation files are stored. A wrong path or missing documentation files will result in termination of the program. Default value: `doc/`

debug: Switches debugging on and off as in normal mode. Default value: `off`

We have seen all parameters and what they are good for. Now users ought to be able to run the program without having difficulties. In case anything goes wrong, despite all the explanations, meaningful error messages will be given to help the user.

This is more or less the point where the new tool is released and the software development process ends. Only one task is left to meet all demanded requirements. A precise step by step tutorial has to be written dealing with the question how to extend functionality and add new learning activities. This is exactly what we are going to do next.

4.7. How to Extend Functionality?

This section gives an idea how the functionality of the final program can be extended. A step-by-step tutorial explains how a new learning activity can be implemented.

4.7.1. Add New Learning Activity

This tutorial provides an in-depth view of some source code and is aimed to guide a developer who is about to add a new learning activity. It is recommended that the developer knows a bit about Java, XML and BPEL. However, in any case, it is best practice to look at files, which already exist, and examples to get an idea of how it functions.

The tutorial is divided into three steps. First, it is explained how to build a new BPEL template file for a new activity. Then, it is shown how a new sequence class can be added. Finally, we draw attention to the documentation file for new activities.

The whole tutorial is based on the reading activity, which is already implemented, so that we can talk about concrete examples.

Write a New Template File

The best way to start developing a new activity is to write the BPEL code for it. In our case, we take over an existing BPEL file running in the Learning Process Manager's execution engine and add the wanted activity.

After incorporating the new BPEL code we try to identify the new code and copy it to a new template file. Let us assume we are developing the reading activity. Listing A.3 on page 102 shows the BPEL code of it. This code listing is already freed from information about a particular reading activity such as, for example, the actual SOAP messages and names and variables of invoke elements. You might notice that some attributes still have a string assigned to it. This is due to the fact that they are used as postfix or prefix strings to make the final generated BPEL source more readable. However, the template file exists exactly of this code. Thus, we only have to put it into a new file named after the activity we want to use in the workflow file to identify the new activity. In our example the name is *reading.xml* and it must be saved in the folder where all

other template files are stored. Do not forget that the root element of every template files is *sequence*. So much for generating a new template file.

Write a New Sequence Class

The next step is to write a new sequence class for our activity. The easiest way to do this is to open and save an existing Sequence class (e.g. DiscussionSequence) under a new name (e.g. ReadingSequence). After changing the name, the constructor and comments of the new class, we can focus on implementing the *getAllElements()* method.

There are two methods existing to facilitate the process of adding specific information to the template code to generate the final BPEL code. The following Java code shows an example how to alter the name attribute of the invoke element:

```
alterAttributes("/sequence//invoke/@name",
    workflowElement.getAttributeValue("name"), POSTFIX, "_");
```

The method takes four parameters. The first parameter is an XPATH string identifying the attributes to change. The second parameter passes the new string for the attributes. The last two parameters tell the method whether to overwrite values stated in the template file or to use these values as postfix or prefix with a separator. In other words, the code lines above change the name of the invoke element to the name provided in the workflow file.

The second helper method can be used to alter text elements. In our case the following code changes SOAP messages:

```
alterElementText("/sequence//in1/item[self::item='url=']",
    workflowElement.getChildText("url"), POSTFIX, "");
```

This method takes exactly the same type of parameters as the method before. The code adds the URL provided in the workflow file to the *item* element with the value `url=`.

With these two methods and some other defined in the abstract Sequence class all necessary modifications can be done within the *getAllElements()* method.

Finally, the SequenceFactory class has to be extended by the new Sequence class. Open SequenceFactory.java and add the following code to the getSequence method:

```
if(sequenceName.equals("reading")) {
return new ReadingSequence(sequenceName, templateDir, debug);
}
```

Now all code modifications are completed. The last step is to write the documentation file.

Documentation File

The documentation file is easy to write but compulsory. Take your time to write it as it is the only way to present all features of the new activity to the user. Watch out for typing errors and wrong XML syntax because it is very likely to be used as a template for workflow input files. It is recommended commenting on each element. Look at Listing 4.2 on page 83 to see an example. Do not forget the root element (i.e. process) and to save the file in the right documentation folder with all the others. The file name has to be the same name as the name of the template file. In our case it is *reading.xml*.

After this step the new learning activity can be used in the workflow file.

4.8. Conclusion

In this chapter the complete development process of the Human-BPEL interface tool has been described. At the beginning there was a need to solve a problem and now a program was written to tackle it.

We did not just start off programming. Some time was invested to look at software engineering from a theoretical point of view. Looking at the history of software development and the definition of software engineering, it was found out that software engineering emerged out of the desperate call for better software by applying proven technologies and practices.

Over the recent years it has become best practice to structure the software development process. A way to apply this strategy is the structured development life cycle. It and its five activities has been described in detail and it has been pointed out how important each activity is and how each activity contributes to the whole process. We have also seen some well-known software development models and finally it was decided to use the Waterfall model.

The first step in the development process was to find out the requirements. The problem we wanted to solve was addressed and the specifications were written down as precise and concise as possible. The main task of the program, all demanded learning activities for the final version and all interfaces and extentions were specified.

The design of the program came next. A lot of time was invested to design the main idea and class diagrams as well as to determine and define all file formats that are used. It must be mentioned that this activity took a lot of time but during the programming activity we were able to appreciate the previous efforts. It was really worth going through the design activity first.

The programming activity was completed without any remarkable problems. After setting up the programming environment consisting of Vim, Java, Apache Ant, Subversion and some external API's, the way was free to write code with references to the class diagrams defined earlier. The programming activity could be described as smooth except of a few difficulties in terms of XML Namespaces.

Then, a real testing environment was installed to test the implementation. The final program was tested by means of real-life scenarios. In addition, code inspection was

applied and it was checked whether the product meets the requirements specification or not.

The last activity of the software development dealt with the usage of the program and its parameters. The parameters are rather self explaining and the program is easy to use. This was the end of the development process. It has to be drawn attention that despite each activity being performed sequentially, sometimes it was necessary to backtrack to a preceding stage. This was kept within limits, though.

In conclusion, it must be said that the software development process was pretty straightforward and turned out to be successful. Now a user, that is, in most cases a teacher, is able to generate a BPEL file from a very simple workflow file. He or she does not have to know anything about BPEL and the generated file can be immediately deployed and executed by the Learning Process Manager's execution engine (i.e. BPEL engine). A teacher can create and manage online courses easily by means of the e-learning system (i.e. WBT-Master), the Learning Process Manager and this new tool. This is exactly what we were looking for. In the future, a GUI could be build on the top of the BPEL tool to help the user write the workflow XML file.

5. Conclusion and Outlook

Now it is time to sum up what we have seen in this paper and say a few words about future development.

In the first part out of three it was shown why Business Process Management is critical for companies and what concepts, technologies and tools are available to support it. Companies have to face steady changing business environments and therefore have to cope with quickly changing business processes to stay competitive. Concepts and specifications were introduced to facilitate the Business Process Management by means of supporting the whole business process life cycle. We came to the conclusion that the new concepts, technologies and standardised specifications had improved the situation compared to the past but especially for big process changes technical experts and time for application development are still needed. It is quite a good approach for slight process changes, though.

The next part tried to figure out whether learning tasks could be supported by Business Process Management's concepts and technologies discussed at the beginning of this paper. Web-based education (also called e-learning and web-based training) is the future. Learning tasks were examined and it was found out that, in fact, they have the same life cycle as business processes. Hence it follows that they can also be treated like business processes and also concepts and technologies for Business Process Management can be applied.

That theory wanted to be proven and the Learning Process Manager (LPM) was evaluated that manages learning processes by means of Business Process Management. We came across the fact that it lacks the support for the first phase of the process life cycle, that is, the modeling of the process for ordinary users. Besides this drawback, it proved to work well with the ideas, concepts and technologies used by Business Process Management.

It was not found a satisfying tool that was able to tackle that issue and started developing our own solution. The third and last part of this paper described it in detail.

In general it can be said that all the concepts, technologies and tools we introduced in this paper work well in terms of the total automation of changing processes. We have to look at the context, though. This means that it depends on the type of process changes. As long as these changes keep within the bounds of possibilities, which means that it is not necessary to develop new or to change services (e.g. web services), the automation is seamless. If changes cross these bounds and it is not enough just to orchestrate and coordinate services, and new services have to be developed, technical experts and development teams are still needed. In this case it is obvious that the pledged automation is not given.

Overall, it seems that the way of process management goes into the right direction. There are a few drawbacks that should be addressed and solved in the future, though. For example and as you probably might have noticed, the technical experts building BPEL files have to face many other necessary files and specifications (e.g. WSDL, SOAP, etc.). Despite of many tools assisting the experts, it is easy to lose the overview and get confused especially in a bigger environment. There is definitely a need for simpler concepts. For instance, Representational State Transfer (REST) is a software architectural style that refers to a set of architectural principles that would make some specifications unnecessary (e.g. SOAP and WSDL in our case) if everybody agrees to this concept[Pre].

Finally, it can be said that even concepts and technologies for supporting process change are scheduled to be changed.

> "To improve is to change;
> to be perfect is to change often." – Winston Churchill

A. APPENDIX

A.1. SOAP Message with Web Service Security

Listing A.1: SOAP Message with Web Service Security

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <S:Header>
  <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
   <m:action>http://fabrikam123.com/getQuote</m:action>
   <m:to>http://fabrikam123.com/stocks</m:to>
   <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
  </m:path>
  <wsse:Security
   xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
   wsse:UsernameToken Id="MyID">
   <wsse:Username>Zoe</wsse:Username>
  </wsse:UsernameToken>
  <ds:Signature>
   <ds:SignedInfo>
    <ds:CanonicalizationMethod
     Algorithm=
     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod
     Algorithm=
     "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#MsgBody">
     <ds:DigestMethod
      Algorithm=
      "http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
   <ds:KeyInfo>
    <wsse:SecurityTokenReference>
     <wsse:Reference URI="#MyID"/>
    </wsse:SecurityTokenReference>
   </ds:KeyInfo>
  </ds:Signature>
 </wsse:Security>
</S:Header>
<S:Body Id="MsgBody">
 <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
  QQQ
 </tru:StockSymbol>
</S:Body>
 </S:Envelope>
```

A.2. Example of a Workflow File

Listing A.2: Example of a workflow file

```
<process name="LPMProcess_ReadAndTest">
 <reading name="reading_1">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/linux1_start0.htm</url>
  <title>Introducing Linux</title>
  <message>First read this!!</message>
 </reading>
 <reading name="reading_2">
  <url>http://www.linux.org/info/index.html</url>
  <title>A short introduction to Linux by its developers</title>
  <message>It will only take 60 minutes.</message>
 </reading>
 <discussion name="discussion_1">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/lb_forum.htm</url>
  <title>Discussion about the Linux basics</title>
 </discussion>
 <test name="test_1">
  <results_id>test</results_id>
  <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
  <title>Test on the very basics of Linux</title>
  <message>(Please note: When you finish the test you will need to wait for the teacher
      to mark it!)</message>
  <completed_message>(Please note: Now, you need to wait for the teacher to mark the
     test!)</completed_message>
 </test>
 <testmarking name="marking_1">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
  <title>Test on the basics of Linux</title>
  <test_name>linux_quiz-nsherbak0605041034</test_name>
  <results_id>test</results_id>
 </testmarking>
 <reading name="reading_3">
  <url>http://www.upscale.utoronto.ca/GeneralInterest/Harrison/LearnLinux/Module1.pdf</
     url>
  <title>Linux installation tutorial</title>
  <message>It will only take 30 minutes.</message>
 </reading>
 <discussion name="discussion_2">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/lb_forum.htm</url>
  <title>Discussion about installing Linux</title>
 </discussion>
 <test name="test_2">
  <results_id>test</results_id>
  <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
  <title>Test on the basics of Linux</title>
  <message>(Please note: When you finish the test you will need to wait for the teacher
      to mark it!)</message>
  <completed_message>(Please note: Now, you need to wait for the teacher to mark the
     test!)</completed_message>
 </test>
 <testmarking name="marking_2">
  <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
  <title>Test on the basics of Linux</title>
  <test_name>linux_quiz-nsherbak0605041034</test_name>
  <results_id>test</results_id>
 </testmarking>

 <while_test_results condition="less 11">
  <reading name="reading_4">
   <url>http://www.selflinux.org/</url>
   <title>Another Linux tutorial</title>
   <message>(Unfortunately, you didn't pass the test! Try to improve your knowledge by
      reading this document!)</message>
  </reading>
  <test name="test_3">
```

```
59      <results_id>test</results_id>
60      <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
61      <title>Test on the basics of Linux</title>
62      <message>(Please note: When you finish the test you will need to wait for the
            teacher to mark it!)</message>
63      <completed_message>(Please note: Now, you need to wait for the teacher to mark the
            test!)</completed_message>
64    </test>
65    <testmarking name="marking_3">
66      <url>http://coronet2.iicm.edu/wbtmaster/courses/linux_quiz_room.htm</url>
67      <title>Test on the basics of Linux</title>
68      <test_name>linux_quiz-nsherbak0605041034</test_name>
69      <results_id>test</results_id>
70    </testmarking>
71   </while_test_results>
72  </process>
```

A.3. Template of the Reading Activity

Listing A.3: Template of the Reading Activity

```
1   <sequence>
2    <!-- add a reading activity to the worklist -->
3    <assign>
4     <copy>
5      <from part="startReturn" variable="started"/>
6      <to part="in0" variable="wi"/>
7     </copy>
8     <copy>
9      <from>
10      <in1 soapenc:arrayType="soapenc:string[4]" xmlns="" xmlns:bpws="http://schemas.
            xmlsoap.org/ws/2003/03/business-process/" xmlns:ns1="http://coronet.iicm.edu/
            lpm/worklistmanagerservice" xmlns:ns2="http://coronet.iicm.edu/lpm/
            processmanagerservice" xmlns:ns3="http://coronet.iicm.edu/lpm/
            callbackmanagerservice" xmlns:ns4="http://coronet.iicm.edu/lpm/bpel" xmlns:
            soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.
            org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
            type="soapenc:Array">
11        <item xsi:type="soapenc:string">reading</item>
12        <item xsi:type="soapenc:string">student</item>
13        <item xsi:type="soapenc:string">url=</item>
14        <item xsi:type="soapenc:string">title=</item>
15        <item xsi:type="soapenc:string">message=</item>
16      </in1>
17     </from>
18     <to part="in1" variable="wi"/>
19    </copy>
20   </assign>
21   <invoke inputVariable="wi" name="addWorkItem" operation="addWorkItem" outputVariable="
         wi_response" partnerLink="WLManagerInvoker" portType="ns1:WorkListManagerService
         "/>
22   <!-- wait for a message from the client that the reading activity has been finished
         -->
23   <receive name="finished_receive" operation="workItemCompleted" partnerLink="
         CManagerReciever" portType="ns3:CallbackManagerService" variable="rf">
24    <correlations>
25     <correlation set="CS1"/>
26    </correlations>
27   </receive>
28   <invoke inputVariable="rf" name="Finished_invoke" operation="workItemCompleted"
         outputVariable="rf_response" partnerLink="CManagerInvoker" portType="ns3:
         CallbackManagerService"/>
```

```
29  <reply name="finished_reply" operation="workItemCompleted" partnerLink="
        CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"/>
30 </sequence>
```

A.4. Example Output File in Documentation Mode

Listing A.4: An example output file of the program running in documentation mode

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <process name="">
3    <!--This is an automatically generated file by BuildBPEL working in documentation
         mode.It shows possible sequences and their settings in no specific order.-->
4    <reading name="">
5      <!-- url to the reading resource -->
6      <url />
7      <!-- the title of the reading -->
8      <title />
9      <!-- message for the users -->
10     <message />
11   </reading>
12   <discussion name="">
13     <!-- URL to the resource -->
14     <url />
15     <!-- Title of the discussion -->
16     <title />
17   </discussion>
18   <test name="">
19     <!-- The same id of the results must be used in the testmarking sequence!!!! -->
20     <results_id />
21     <!-- URL to the resource -->
22     <url />
23     <!-- Title of the test -->
24     <title />
25     <!-- Message for the student before starting the test-->
26     <message />
27     <!-- Message for the student after doing the test -->
28     <completed_message />
29   </test>
30   <testmarking name="">
31     <!-- You must use the same ID as in the test sequence if you want to mark that test
           -->
32     <results_id />
33     <!-- URL to the resource -->
34     <url />
35     <!-- Title of the activity -->
36     <title />
37     <!-- Name of the test -->
38     <test_name />
39   </testmarking>
40   <while_test_results condition="less 11">
41     <!-- Under this structured activity you can use all available sequences but end
          with a test and testmarking sequence since the condition is about test results
          -->
42     <test name="">
43       <results_id />
44       <url />
45       <title />
46       <message />
47       <completed_message />
48     </test>
49     <testmarking name="">
50       <url />
```

```
51        <title />
52        <test_name />
53        <results_id />
54      </testmarking>
55    </while_test_results>
56  </process>
```

A.5. Learning Process Example Described through BPEL

Listing A.5: Example of a learning process described through BPEL

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:bpws="
       http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:ns1="http://coronet.
       iicm.edu/lpm/worklistmanagerservice" xmlns:ns2="http://coronet.iicm.edu/lpm/
       processmanagerservice" xmlns:ns3="http://coronet.iicm.edu/lpm/
       callbackmanagerservice" xmlns:ns4="http://coronet.iicm.edu/lpm/bpel" xmlns:xsd="
       http://www.w3.org/2001/XMLSchema" name="LPMProcess_ReadAndTest" suppressJoinFailure
       ="yes" targetNamespace="http://coronet.iicm.edu/lpm/bpel/process">
3    <partnerLinks>
4      <partnerLink myRole="PManager" name="PManagerStarter" partnerLinkType="ns4:
           PMLinkType" />
5      <partnerLink name="PManagerInvoker" partnerLinkType="ns4:PMLinkType" partnerRole="
           PManager" />
6      <partnerLink name="WLManagerInvoker" partnerLinkType="ns4:WLMLinkType" partnerRole
           ="WLManager" />
7      <partnerLink myRole="CManager" name="CManagerReciever" partnerLinkType="ns4:
           CMLinkType" />
8      <partnerLink name="CManagerInvoker" partnerLinkType="ns4:CMLinkType" partnerRole="
           CManager" />
9    </partnerLinks>
10   <variables>
11     <variable messageType="ns2:startRequest" name="start" />
12     <variable messageType="ns2:startResponse" name="started" />
13     <variable messageType="ns1:addWorkItemRequest" name="wi" />
14     <variable messageType="ns1:addWorkItemResponse" name="wi_response" />
15     <variable messageType="ns3:workItemCompletedRequest" name="rf" />
16     <variable messageType="ns3:workItemCompletedResponse" name="rf_response" />
17     <variable messageType="ns2:setProcessCompletedRequest" name="complete" />
18     <variable messageType="ns2:setProcessCompletedResponse" name="completed" />
19     <variable name="test_result" type="xsd:int" />
20   </variables>
21   <correlationSets>
22     <correlationSet name="CS1" properties="ns4:process_identifier" />
23   </correlationSets>
24   <sequence>
25     <receive createInstance="yes" name="startProcess_receive" operation="start"
           partnerLink="PManagerStarter" portType="ns2:ProcessManagerService" variable="
           start" />
26     <invoke inputVariable="start" name="startProcess_invoke" operation="start"
           outputVariable="started" partnerLink="PManagerInvoker" portType="ns2:
           ProcessManagerService" />
27     <reply name="startProcess_reply" operation="start" partnerLink="PManagerStarter"
           portType="ns2:ProcessManagerService" variable="started">
28       <correlations>
29         <correlation initiate="yes" set="CS1" />
30       </correlations>
31     </reply>
32     <assign>
33       <copy>
```

```
34          <from part="startReturn" variable="started" />
35          <to part="in0" variable="wi" />
36        </copy>
37        <copy>
38          <from>
39            <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[4]" xsi:type="soapenc:Array">
40              <item xsi:type="soapenc:string">reading</item>
41              <item xsi:type="soapenc:string">student</item>
42              <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/linux1_start0.htm</item>
43              <item xsi:type="soapenc:string">title=Introducing Linux</item>
44              <item xsi:type="soapenc:string">message=First read this!!</item>
45            </in1>
46          </from>
47          <to part="in1" variable="wi" />
48        </copy>
49      </assign>
50      <invoke inputVariable="wi" name="addWorkItem_reading_1" operation="addWorkItem"
            outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
            WorkListManagerService" />
51      <receive name="finished_receive_reading_1" operation="workItemCompleted"
            partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
            rf">
52        <correlations>
53          <correlation set="CS1" />
54        </correlations>
55      </receive>
56      <invoke inputVariable="rf" name="Finished_invoke_reading_1" operation="
            workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
            portType="ns3:CallbackManagerService" />
57      <reply name="finished_reply_reading_1" operation="workItemCompleted" partnerLink="
            CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
            />
58      <assign>
59        <copy>
60          <from part="startReturn" variable="started" />
61          <to part="in0" variable="wi" />
62        </copy>
63        <copy>
64          <from>
65            <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[4]" xsi:type="soapenc:Array">
66              <item xsi:type="soapenc:string">reading</item>
67              <item xsi:type="soapenc:string">student</item>
68              <item xsi:type="soapenc:string">url=http://www.linux.org/info/index.html</
                    item>
69              <item xsi:type="soapenc:string">title=A short introduction to Linux by its
                    developers</item>
70              <item xsi:type="soapenc:string">message=It will only take 60 minutes.</item
                    >
71            </in1>
72          </from>
73          <to part="in1" variable="wi" />
74        </copy>
75      </assign>
76      <invoke inputVariable="wi" name="addWorkItem_reading_2" operation="addWorkItem"
            outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
            WorkListManagerService" />
77      <receive name="finished_receive_reading_2" operation="workItemCompleted"
            partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
            rf">
78        <correlations>
```

```
79          <correlation set="CS1" />
80        </correlations>
81      </receive>
82      <invoke inputVariable="rf" name="Finished_invoke_reading_2" operation="
            workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
            portType="ns3:CallbackManagerService" />
83      <reply name="finished_reply_reading_2" operation="workItemCompleted" partnerLink="
            CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
            />
84      <assign>
85        <copy>
86          <from part="startReturn" variable="started" />
87          <to part="in0" variable="wi" />
88        </copy>
89        <copy>
90          <from>
91            <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                string[4]" xsi:type="soapenc:Array">
92              <item xsi:type="soapenc:string">discussion</item>
93              <item xsi:type="soapenc:string">student</item>
94              <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/lb_forum.htm</item>
95              <item xsi:type="soapenc:string">title=Discussion about the Linux basics</
                    item>
96            </in1>
97          </from>
98          <to part="in1" variable="wi" />
99        </copy>
100     </assign>
101     <invoke inputVariable="wi" name="addWorkItem_discussion_1" operation="addWorkItem"
            outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
            WorkListManagerService" />
102     <receive name="finished_receive_discussion_1" operation="workItemCompleted"
            partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
            rf">
103       <correlations>
104         <correlation set="CS1" />
105       </correlations>
106     </receive>
107     <invoke inputVariable="rf" name="finished_invoke_discussion_1" operation="
            workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
            portType="ns3:CallbackManagerService" />
108     <reply name="finished_reply_discussion_1" operation="workItemCompleted" partnerLink
            ="CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response
            " />
109     <assign>
110       <copy>
111         <from part="startReturn" variable="started" />
112         <to part="in0" variable="wi" />
113       </copy>
114       <copy>
115         <from>
116           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                string[7]" xsi:type="soapenc:Array">
117             <item xsi:type="soapenc:string">questiontest</item>
118             <item xsi:type="soapenc:string">student</item>
119             <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/linux_quiz_room.htm</item>
120             <item xsi:type="soapenc:string">title=Test on the very basics of Linux</
                    item>
121             <item xsi:type="soapenc:string">message=(Please note: When you finish the
                    test you will need to wait for the teacher to mark it!)</item>
122             <item xsi:type="soapenc:string">observable=test_results</item>
```

```
123             <item xsi:type="soapenc:string">completed_message=(Please note: Now, you
                    need to wait for the teacher to mark the test!)</item>
124           </in1>
125         </from>
126         <to part="in1" variable="wi" />
127       </copy>
128     </assign>
129     <invoke inputVariable="wi" name="addWorkItem_test_1" operation="addWorkItem"
          outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
          WorkListManagerService" />
130     <receive name="finished_receive_test_1" operation="workItemCompleted" partnerLink="
          CManagerReciever" portType="ns3:CallbackManagerService" variable="rf">
131       <correlations>
132         <correlation set="CS1" />
133       </correlations>
134     </receive>
135     <invoke inputVariable="rf" name="finished_invoke_test_1" operation="
          workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
          portType="ns3:CallbackManagerService" />
136     <reply name="finished_reply_test_1" operation="workItemCompleted" partnerLink="
          CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
          />
137     <assign>
138       <copy>
139         <from part="startReturn" variable="started" />
140         <to part="in0" variable="wi" />
141       </copy>
142       <copy>
143         <from>
144           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[8]" xsi:type="soapenc:Array">
145             <item xsi:type="soapenc:string">testmarking</item>
146             <item xsi:type="soapenc:string">teacher</item>
147             <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/linux_quiz_room.htm</item>
148             <item xsi:type="soapenc:string">title=Test on the basics of Linux</item>
149             <item xsi:type="soapenc:string">target_endpoint_address=http://coronet2.
                    iicm.edu/axis/services/gettestresultservice</item>
150             <item xsi:type="soapenc:string">target_namespace=http://coronet.iicm.edu/
                    lpm/gettestresultservice</item>
151             <item xsi:type="soapenc:string">test_name=linux_quiz-nsherbak0605041034</
                    item>
152             <item xsi:type="soapenc:string">notify_observable=test_results</item>
153           </in1>
154         </from>
155         <to part="in1" variable="wi" />
156       </copy>
157     </assign>
158     <invoke inputVariable="wi" name="addWorkItem_marking_1" operation="addWorkItem"
          outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
          WorkListManagerService" />
159     <receive name="finished_receive_marking_1" operation="workItemCompleted"
          partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
          rf">
160       <correlations>
161         <correlation set="CS1" />
162       </correlations>
163     </receive>
164     <invoke inputVariable="rf" name="finished_invoke_marking_1" operation="
          workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
          portType="ns3:CallbackManagerService" />
165     <reply name="finished_reply_marking_1" operation="workItemCompleted" partnerLink="
          CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
          />
```

```
166     <assign>
167       <copy>
168         <from part="in4" query="/in4/string" variable="rf" />
169         <to variable="test_result" />
170       </copy>
171     </assign>
172     <assign>
173       <copy>
174         <from part="startReturn" variable="started" />
175         <to part="in0" variable="wi" />
176       </copy>
177       <copy>
178         <from>
179           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[4]" xsi:type="soapenc:Array">
180             <item xsi:type="soapenc:string">reading</item>
181             <item xsi:type="soapenc:string">student</item>
182             <item xsi:type="soapenc:string">url=http://www.upscale.utoronto.ca/
                    GeneralInterest/Harrison/LearnLinux/Module1.pdf</item>
183             <item xsi:type="soapenc:string">title=Linux installation tutorial</item>
184             <item xsi:type="soapenc:string">message=It will only take 30 minutes.</item
                    >
185           </in1>
186         </from>
187         <to part="in1" variable="wi" />
188       </copy>
189     </assign>
190     <invoke inputVariable="wi" name="addWorkItem_reading_3" operation="addWorkItem"
          outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
          WorkListManagerService" />
191     <receive name="finished_receive_reading_3" operation="workItemCompleted"
          partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
          rf">
192       <correlations>
193         <correlation set="CS1" />
194       </correlations>
195     </receive>
196     <invoke inputVariable="rf" name="Finished_invoke_reading_3" operation="
          workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
          portType="ns3:CallbackManagerService" />
197     <reply name="finished_reply_reading_3" operation="workItemCompleted" partnerLink="
          CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
          />
198     <assign>
199       <copy>
200         <from part="startReturn" variable="started" />
201         <to part="in0" variable="wi" />
202       </copy>
203       <copy>
204         <from>
205           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[4]" xsi:type="soapenc:Array">
206             <item xsi:type="soapenc:string">discussion</item>
207             <item xsi:type="soapenc:string">student</item>
208             <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/lb_forum.htm</item>
209             <item xsi:type="soapenc:string">title=Discussion about installing Linux</
                    item>
210           </in1>
211         </from>
212         <to part="in1" variable="wi" />
213       </copy>
214     </assign>
```

```
215     <invoke inputVariable="wi" name="addWorkItem_discussion_2" operation="addWorkItem"
            outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
            WorkListManagerService" />
216     <receive name="finished_receive_discussion_2" operation="workItemCompleted"
            partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
            rf">
217       <correlations>
218         <correlation set="CS1" />
219       </correlations>
220     </receive>
221     <invoke inputVariable="rf" name="finished_invoke_discussion_2" operation="
            workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
            portType="ns3:CallbackManagerService" />
222     <reply name="finished_reply_discussion_2" operation="workItemCompleted" partnerLink
            ="CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response
            " />
223     <assign>
224       <copy>
225         <from part="startReturn" variable="started" />
226         <to part="in0" variable="wi" />
227       </copy>
228       <copy>
229         <from>
230           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
                  string[7]" xsi:type="soapenc:Array">
231             <item xsi:type="soapenc:string">questiontest</item>
232             <item xsi:type="soapenc:string">student</item>
233             <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/linux_quiz_room.htm</item>
234             <item xsi:type="soapenc:string">title=Test on the basics of Linux</item>
235             <item xsi:type="soapenc:string">message=(Please note: When you finish the
                    test you will need to wait for the teacher to mark it!)</item>
236             <item xsi:type="soapenc:string">observable=test_results</item>
237             <item xsi:type="soapenc:string">completed_message=(Please note: Now, you
                    need to wait for the teacher to mark the test!)</item>
238           </in1>
239         </from>
240         <to part="in1" variable="wi" />
241       </copy>
242     </assign>
243     <invoke inputVariable="wi" name="addWorkItem_test_2" operation="addWorkItem"
            outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
            WorkListManagerService" />
244     <receive name="finished_receive_test_2" operation="workItemCompleted" partnerLink="
            CManagerReciever" portType="ns3:CallbackManagerService" variable="rf">
245       <correlations>
246         <correlation set="CS1" />
247       </correlations>
248     </receive>
249     <invoke inputVariable="rf" name="finished_invoke_test_2" operation="
            workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
            portType="ns3:CallbackManagerService" />
250     <reply name="finished_reply_test_2" operation="workItemCompleted" partnerLink="
            CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
            />
251     <assign>
252       <copy>
253         <from part="startReturn" variable="started" />
254         <to part="in0" variable="wi" />
255       </copy>
256       <copy>
257         <from>
258           <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
                  http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="soapenc:
```

```
                string[8]" xsi:type="soapenc:Array">
259             <item xsi:type="soapenc:string">testmarking</item>
260             <item xsi:type="soapenc:string">teacher</item>
261             <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                    courses/linux_quiz_room.htm</item>
262             <item xsi:type="soapenc:string">title=Test on the basics of Linux</item>
263             <item xsi:type="soapenc:string">target_endpoint_address=http://coronet2.
                    iicm.edu/axis/services/gettestresultservice</item>
264             <item xsi:type="soapenc:string">target_namespace=http://coronet.iicm.edu/
                    lpm/gettestresultservice</item>
265             <item xsi:type="soapenc:string">test_name=linux_quiz-nsherbak0605041034</
                    item>
266             <item xsi:type="soapenc:string">notify_observable=test_results</item>
267           </in1>
268         </from>
269         <to part="in1" variable="wi" />
270       </copy>
271     </assign>
272     <invoke inputVariable="wi" name="addWorkItem_marking_2" operation="addWorkItem"
          outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
          WorkListManagerService" />
273     <receive name="finished_receive_marking_2" operation="workItemCompleted"
          partnerLink="CManagerReciever" portType="ns3:CallbackManagerService" variable="
          rf">
274       <correlations>
275         <correlation set="CS1" />
276       </correlations>
277     </receive>
278     <invoke inputVariable="rf" name="finished_invoke_marking_2" operation="
          workItemCompleted" outputVariable="rf_response" partnerLink="CManagerInvoker"
          portType="ns3:CallbackManagerService" />
279     <reply name="finished_reply_marking_2" operation="workItemCompleted" partnerLink="
          CManagerReciever" portType="ns3:CallbackManagerService" variable="rf_response"
          />
280     <assign>
281       <copy>
282         <from part="in4" query="/in4/string" variable="rf" />
283         <to variable="test_result" />
284       </copy>
285     </assign>
286     <while condition="bpws:getVariableData('test_result') &lt; 11">
287       <sequence>
288         <assign>
289           <copy>
290             <from part="startReturn" variable="started" />
291             <to part="in0" variable="wi" />
292           </copy>
293           <copy>
294             <from>
295               <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi
                      ="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="
                      soapenc:string[4]" xsi:type="soapenc:Array">
296                 <item xsi:type="soapenc:string">reading</item>
297                 <item xsi:type="soapenc:string">student</item>
298                 <item xsi:type="soapenc:string">url=http://www.selflinux.org/</item>
299                 <item xsi:type="soapenc:string">title=Another Linux tutorial</item>
300                 <item xsi:type="soapenc:string">message=(Unfortunately, you didn't pass
                        the test! Try to improve your knowledge by reading this document!)
                      </item>
301               </in1>
302             </from>
303             <to part="in1" variable="wi" />
304           </copy>
305         </assign>
```

```
306         <invoke inputVariable="wi" name="addWorkItem_reading_4" operation="addWorkItem"
                 outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
                WorkListManagerService" />
307         <receive name="finished_receive_reading_4" operation="workItemCompleted"
                partnerLink="CManagerReciever" portType="ns3:CallbackManagerService"
                variable="rf">
308           <correlations>
309             <correlation set="CS1" />
310           </correlations>
311         </receive>
312         <invoke inputVariable="rf" name="Finished_invoke_reading_4" operation="
                workItemCompleted" outputVariable="rf_response" partnerLink="
                CManagerInvoker" portType="ns3:CallbackManagerService" />
313         <reply name="finished_reply_reading_4" operation="workItemCompleted"
                partnerLink="CManagerReciever" portType="ns3:CallbackManagerService"
                variable="rf_response" />
314         <assign>
315           <copy>
316             <from part="startReturn" variable="started" />
317             <to part="in0" variable="wi" />
318           </copy>
319           <copy>
320             <from>
321               <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi
                      ="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="
                      soapenc:string[7]" xsi:type="soapenc:Array">
322                 <item xsi:type="soapenc:string">questiontest</item>
323                 <item xsi:type="soapenc:string">student</item>
324                 <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                        courses/linux_quiz_room.htm</item>
325                 <item xsi:type="soapenc:string">title=Test on the basics of Linux</item
                        >
326                 <item xsi:type="soapenc:string">message=(Please note: When you finish
                        the test you will need to wait for the teacher to mark it!)</item>
327                 <item xsi:type="soapenc:string">observable=test_results</item>
328                 <item xsi:type="soapenc:string">completed_message=(Please note: Now,
                        you need to wait for the teacher to mark the test!)</item>
329               </in1>
330             </from>
331             <to part="in1" variable="wi" />
332           </copy>
333         </assign>
334         <invoke inputVariable="wi" name="addWorkItem_test_3" operation="addWorkItem"
                outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
                WorkListManagerService" />
335         <receive name="finished_receive_test_3" operation="workItemCompleted"
                partnerLink="CManagerReciever" portType="ns3:CallbackManagerService"
                variable="rf">
336           <correlations>
337             <correlation set="CS1" />
338           </correlations>
339         </receive>
340         <invoke inputVariable="rf" name="finished_invoke_test_3" operation="
                workItemCompleted" outputVariable="rf_response" partnerLink="
                CManagerInvoker" portType="ns3:CallbackManagerService" />
341         <reply name="finished_reply_test_3" operation="workItemCompleted" partnerLink="
                CManagerReciever" portType="ns3:CallbackManagerService" variable="
                rf_response" />
342         <assign>
343           <copy>
344             <from part="startReturn" variable="started" />
345             <to part="in0" variable="wi" />
346           </copy>
347           <copy>
348             <from>
```

```
349                <in1 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi
                       ="http://www.w3.org/2001/XMLSchema-instance" soapenc:arrayType="
                       soapenc:string[8]" xsi:type="soapenc:Array">
350                  <item xsi:type="soapenc:string">testmarking</item>
351                  <item xsi:type="soapenc:string">teacher</item>
352                  <item xsi:type="soapenc:string">url=http://coronet2.iicm.edu/wbtmaster/
                         courses/linux_quiz_room.htm</item>
353                  <item xsi:type="soapenc:string">title=Test on the basics of Linux</item
                         >
354                  <item xsi:type="soapenc:string">target_endpoint_address=http://coronet2
                         .iicm.edu/axis/services/gettestresultservice</item>
355                  <item xsi:type="soapenc:string">target_namespace=http://coronet.iicm.
                         edu/lpm/gettestresultservice</item>
356                  <item xsi:type="soapenc:string">test_name=linux_quiz-nsherbak0605041034
                         </item>
357                  <item xsi:type="soapenc:string">notify_observable=test_results</item>
358                </in1>
359              </from>
360              <to part="in1" variable="wi" />
361            </copy>
362          </assign>
363          <invoke inputVariable="wi" name="addWorkItem_marking_3" operation="addWorkItem"
                 outputVariable="wi_response" partnerLink="WLManagerInvoker" portType="ns1:
                WorkListManagerService" />
364          <receive name="finished_receive_marking_3" operation="workItemCompleted"
                partnerLink="CManagerReciever" portType="ns3:CallbackManagerService"
                variable="rf">
365            <correlations>
366              <correlation set="CS1" />
367            </correlations>
368          </receive>
369          <invoke inputVariable="rf" name="finished_invoke_marking_3" operation="
                workItemCompleted" outputVariable="rf_response" partnerLink="
                CManagerInvoker" portType="ns3:CallbackManagerService" />
370          <reply name="finished_reply_marking_3" operation="workItemCompleted"
                partnerLink="CManagerReciever" portType="ns3:CallbackManagerService"
                variable="rf_response" />
371          <assign>
372            <copy>
373              <from part="in4" query="/in4/string" variable="rf" />
374              <to variable="test_result" />
375            </copy>
376          </assign>
377        </sequence>
378      </while>
379      <assign>
380        <copy>
381          <from part="startReturn" variable="started" />
382          <to part="in0" variable="complete" />
383        </copy>
384      </assign>
385      <invoke inputVariable="complete" name="setProcessCompleted_invoke" operation="
            setProcessCompleted" outputVariable="completed" partnerLink="PManagerInvoker"
            portType="ns2:ProcessManagerService" />
386    </sequence>
387  </process>
```

Bibliography

[AAA+] Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guízar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter König, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu. Web Services Business Process Execution Language Version 2.0. Available from: `http://www.oasis-open.org/committees/download.php/20074/wsbpel-specification-draft.html` [cited August 29, 2006].

[BPSM+04] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible Markup Language (XML) 1.1. February 2004. Available from: `http://www.w3.org/TR/2004/REC-xml11-20040204/` [cited July 26, 2006].

[BSW] Stefan Berndes, Alexander Stanke, and Kai Wörner. *Business Process Modelling*. Springer. Business Process Management for Open Processes: Method and Tool to Support Product Development Processes.

[Cha06] James Champy. People and Process. *ACM Queue*, 4-2, March 2006.

[Cov] Robin Cover. Security Assertion Markup Language (SAML). Available from: `http://xml.coverpages.org/saml.html` [cited August 12, 2006].

[CXHK05] Kai Cheng, Limin Xiang, Toyohiko Hirota, and Ushijima Kazuo. A Web-Based Classroom Environment for Enhanced Residential College Education. *Lecture Notes in Computer Science*, 3583:56–65, 2005.

[DT04] Philip Dodds and Schawn E. Thropp. SCORM Sequencing and Navigation Version 1.3.1. July 2004. Available from: `http://www.adlnet.gov/downloads/files/67.cfm`.

[FY05] Heidi Fung and Allan Yuen. Student Adoption Towards Web-Based Learning Platform. *Lecture Notes in Computer Science*, 3583:14–25, 2005.

[GM98] Stephen Gilbert and Bill McCarty. *Object-Oriented Design in Java*. Mitchell Waite, 1998.

[Ham97] Michael Hammer. *Beyond Reengineering*. HarperBusiness, 1997.

[Hel06] Denis Helic. Technology-Supported Management of Collaborative Learning Processes. Paper available at Institute for Information Systems and Computer Media at Graz University of Technology, Austria, 2006.

[HM] Jason Hunter and Brett McLaughlin. Easy Java/XML integration with JDOM, Part 1.

[HQH05] Qinming He, Ling Qiu, and Zhen He. Design and Implementation of a J2EE-Based Platform for Network Teaching. *Lecture Notes in Computer Science*, 3583:49–55, 2005.

[IEE93] IEEE. IEEE Standards Collection: Software Engineering. IEEE Standard 610.12-1990, 1993.

[Jon06] Peter De Jong. Going with the Flow. *ACM Queue*, 4-2, March 2006.

[Jur] Matjaz B. Juric. A Hands-on Introduction to BPEL. Available from: `http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html` [cited August 30, 2006].

[KND+97] Peter King, Patrick Naughton, Mike DeMoney, Jonni Kanerva, Kathy Walrath, and Scott Hommel. Java Code Conventions. September 1997. Available from: `http://java.sun.com/docs/codeconv/` [cited July 6, 2006].

[Ley] Frank Leymann. Web Services Flow Language (WSFL 1.0). Available from: `http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf` [cited August 29, 2006].

[LM98] James Laffey and Dale Musser. Attitudes of preservice teachers about using technology in teaching. *Journal of Technology and Teacher Education*, 6:223–241, 1998.

[Mie06] Derek Miers. Best Practice BPM. *ACM Queue*, 4-2, March 2006.

[Moo06] Kumar Raj Moorthy. An Introduction to BPEL. May 2006. Available from: `http://www.developer.com/services/print.php/36093812` [cited September 9, 2006].

[Mye04] Glenford J. Myers. *The Art of Software Testing*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2004.

[Nai04] Bhagat Nainani. Closed Loop BPM using Standards based Tools. November 2004. Available from: `http://www.oracle.com/technology/products/ias/bpel/pdf/bpm.closedloop.pdf` [cited June 23, 2006].

[NCKC05] S.C. Ng, S.O. Choy, R. Kwan, and S.F. Chan. A Web-Based Environment to Improve Teaching and Learning of Computer Programming in Distance Education. *Lecture Notes in Computer Science*, 3583:279–290, 2005.

[NR69] Peter Naur and Brian Randell. Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee. 1969.

[OAS] OASIS. Web Services Security. Available from: `http://www-128.ibm.com/developerworks/library/specification/ws-secure/` [cited August 12, 2006].

[OAS04] OASIS. Web Services Security v1.0 (WS-Security 2004). March 2004. Available from: `http://www.oasis-open.org/specs/index.php#wssv1.0` [cited August 12, 2006].

[OR] Martin Owen and Jog Raj. BPMN and Business Process Management. Introduction to the New Business Process Modeling Standard.

[Ort05] Ed Ort. Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools. April 2005. Available from: `http://java.sun.com/developer/technicalArticles/WebServices/soa2/` [cited July 28, 2006].

[Pre] Paul Prescod. REST and the Real World. Available from: `http://www.xml.com/pub/a/ws/2002/02/20/rest.html` [cited September 10, 2006].

[Roh] Michael Rohloff. *Business Process Modelling*. Springer. An Object Oriented Approach to Business Process Modelling.

[Scea] Nikolai Scerbakov. WBT-Master Manual (v.07). Available from: `http://coronet.iicm.edu/manual/manual.zip` [cited March 18, 2006].

[Sceb] Nikolai Scerbakov. WBT-Master User Handbook (v.07). Available from: `http://coronet.iicm.edu/manual/handbook.zip` [cited March 18, 2006].

[SF03] Howard Smith and Peter Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, 2003.

[SMGMO+05] Jose Luis Sierra, Pablo Moreno-Ger, Ivan Martinez-Ortiz, Javier Lopez-Moratalla, and Baltasar Fernandez-Manjon. Building Learning Management Systems Using IMS Standards: Architecture of a Manifest Driven Approach. *Lecture Notes in Computer Science*, 3583:144–156, 2005.

[Som01] Ian Sommerville. *Sofware Engineering*. Pearson Education Limited, 6th edition, 2001.

[Ste99] Colin Steed. *Web-based Training*. Gower Publishing Limited, 1999.

[Tha] Satish Thatte. XLANG: Web Services for Business Process Design. Available from: `http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm` [cited August 29, 2006].

[vV93] Hans van Vliet. *Software Engineering: Principles and Practice*. John Wiley and Sons, 1993.

[W3Ca] W3C. Extensible Markup Language (XML). Available from: `http://www.w3.org/XML/` [cited July 26, 2006].

[W3Cb] W3C. SOAP Version 1.2. Available from: `http://www.w3.org/TR/soap/` [cited August 11, 2006].

[W3Cc] W3C. Web Services Description Language (WSDL) 1.1. Available from: `http://www.w3.org/TR/wsdl` [cited August 11, 2006].

[Wes99] Bernhard Westfechtel. Process Management. *Lecture Notes in Computer Science*, 1646:3–50, 1999.

[Whi] Stephen A. White. Business Process Modeling Notation. Available from: `http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf` [cited June 22, 2006].

[WST+05] Minjuan Wang, Ruimin Shen, Ren Tong, Fan Yang, and Peng Han. Mobile Learning with Cellphones and PocketPCs. *Lecture Notes in Computer Science*, 3583:332–339, 2005.

[Yil00] Soner Yildirim. Effects of an Educational Computing Course on Preservice and Inservice Teachers: A Discussion and Analysis of Attitudes and Use. *Journal of Research on Computing in Education*, 32:479–495, 2000.

[ZR] Olaf Zukunft and Frank Rump. *Business Process Modelling*. Springer. From Business Process Modelling to Workflow Management: An Integrated Approach.