A Transformation of Business Process Models into Software-Executable Models Using MDA*

Nuno Santos[1], Francisco J. Duarte[2],
Ricardo J. Machado[2], and João M. Fernandes[2]

[1] CCG - Centro de Computação Gráfica, Guimarães, Portugal
[2] Centro Algoritmi – Universidade do Minho, Braga/Guimarães, Portugal

Abstract. Traditional software development projects for process-oriented organizations are time consuming and do not always guarantee the fulfillment of the functional requirements of the client organization, and thus the quality of the resulting software product. To reduce the time spent for developing software and improve its quality, we adopt the inclusion of automation in some parts of the software development process. Thus, in this paper, we propose a model transformation approach to derive an executable model for the business processes of a given organization. We execute a mapping between processes (described with a business process execution language) and software components. We also propose a supporting software architecture based on an Enterprise Service Bus and on Java Business Integration, and we use an already defined methodology to execute the model transformation project.

Keywords: business process, JBI, model-driven architecture, MDA, enterprise service bus, ESB.

1 Introduction

Business Process Management (BPM) [1] is a discipline followed by organizations where business processes are required to exist, either by quality norms or by internal directives. Additionally, to cope with the requirements of the business processes, the software development process must properly support them [2,3].

In every organization, it is desirable to reduce the time and the cost to implement business processes in software systems. An aggravating factor during the development of software to support business processes is the diversity of applications used in a real-world business context, which causes integration problems.

We base our approach on the Model-Driven Architecture (MDA) [4] initiative from the OMG. We use two types of models: a Platform-Independent Model (PIM), representing the business process, and a Platform-Specific Model (PSM), allowing the PIM to be executed in software. With our business process-based

* This work is financed by Fundos FEDER through Programa Operacional Fatores de Competitividade – COMPETE e por Fundos Nacionais através da FCT – Fundação para a Ciência e Tecnologia no âmbito do Projeto: FCOMP-01-0124-FEDER-022674.

D. Winkler, S. Biffl, and J. Bergsmann (Eds.): SWQD 2013, LNBIP 133, pp. 147–167, 2013.

approach, the complexity to implement in software the functional requirements derived from business processes is reduced because, among others, of the automation used in model transformations.

The effort to improve the quality of the resulting software product results in a better fulfillment of the functional requirements expressed in the business processes because of the diminishing gaps between business process models and software that our approach facilitates. In projects that adopt model-driven techniques, the model transformations are crucial for the overall success of the project, because they allow moving features from abstract models into software-executable ones, without loosing quality.

We use an Enterprise Service Bus (ESB), the Apache ServiceMix[1], for the PSM implementation. Typically, software solutions based on ESBs are loose-coupled, use reliable messaging mechanisms, and integrate different software technologies.

In this paper, we present a model-driven transformation approach for implementing business process models into software. The considered approach reduces the complexity to implement the business models into software, thus improving the overall quality of the information system. Our transformation approach is part of the Business Implementation Methodology (BIM) [5], which adopts reference models of business processes to provide business best practices to the client organization. However, it is important to note that the approach is sufficiently generic to be adopted in different methodological contexts.

In section 2, we present the state of art, namely the phases and states of the BIM, the MDA based model transformations, and the Apache ServiceMix ESB. In section 3, we propose a quantitative method to select the most appropriate language for modeling the business processes of a software development organization, including explicitly considering the specific staffing environment of a project. Section 3 also describes a case study, executed at Bosch Car Multimedia Portugal. Section 4 presents the business process model transformations, according to the MDA principles. We claim the adequateness of this approach to move from a business process model into a software-executable model, following BIM. First, we establish a correlation between the four states that business process models pass through in BIM, and the states of the PIM and the PSM defined in MDA. A business process model at the PIM level, ready to be transformed into software, is then established. The transformation process is completed by mapping platform-independent elements of the business process model into platform-specific ones. The resulting business process model at the PSM level is presented in section 5. In section 6, the conclusions of the work and proposals for future work are discussed.

2 Model-Driven Implementation of Business Processes

2.1 BIM

BIM is a methodology specially targeted for implementing in software the business processes of a given organization. This methodology proposes the use of

[1] `http://servicemix.apache.org`

best practices in the business domains and allows the customization of a business process according to the specific needs of the organization. It also promotes the building of a software solution with components of different technologies. BIM is composed of four phases (Fig. 1): the 'Selection' of the adequate generic business processes; the 'Definition' of the business processes to use; the 'Concretization' of the business processes into the software system; and the 'Implementation' in software of the various elements that compose the process.

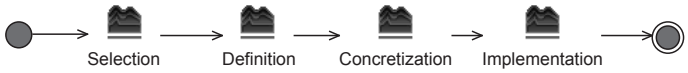

Fig. 1. The four phases of the BIM ([5])

For each phase, BIM describes a corresponding state of the process framework (PF) (Fig. 2). The PF is an artifact of the BIM methodology representing the business processes at different implementation phases. Once the necessary requirements to complete each phase are fulfilled, a new state is assigned to the PF. The state of the PF is mainly defined by the state of the business process model. The four states defined in the methodology are 'Generic', 'Instantiated', 'Runnable' and 'Software Implemented'.

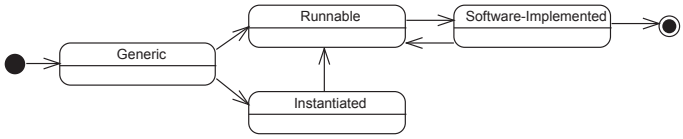

Fig. 2. The Process Framework states in BIM ([5])

The bi-directional state transformations from 'Runnable' and 'Software-Implemented' are possible by having latent runnable business processes moved into a software-implemented state and vice versa.

2.2 Model-Driven Architecture

In an MDA-guided project, after modeling the business processes, one can obtain a software-executable business process model; this is basically a transformation from a PIM into a PSM. For these kinds of transformations, the commonly accepted standard is OMG MOF Query/View/Transformation (MOF QVT) language [6]. It allows the creation of relations between models based in transformation rules. A transformation of PIM business processes, modeled by UML 2 Activity Diagrams, into a PSM in BPEL, through Regular Expression Language (REL) is demonstrated in [7]. The same kind of transformation is described in [8], using the ATLAS Transformation Language (ATL) [9]. A similar transformation is described using the Object Constraint Language (OCL) rules in [10].

Another kind of approach is proposed in [11], which begins by designing a CIM business process model in EPC, then continues by transforming the CIM business process model into a platform-independent one in BPMN, and finally obtains the platform-specific business process model in BPEL. Another approach is presented in [12], which describes a transformation of a CIM modeled in BPMN into a PIM modeled in UML, using either use case or class diagrams.

One of the characteristics of an MDA project is the clear separation between the specification of the system functionalities and the description of how the platform resources are used. An MDA project suggests the following:

- both the environment and the requirements of the system are specified (CIM);
- the system is specified independently from the platform that supports it (PIM);
- the platform is specified;
- a platform is chosen for the system;
- the specification of the system is transformed into specifications containing details of a specific platform (PSM).

The PSM is obtained from the transformation process that takes the PIM as the input. The transformation of the PIM into the PSM is accomplished by combining the PIM with specific details of the platform (Fig. 3).

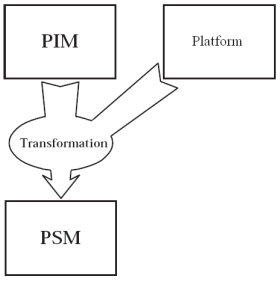

Fig. 3. Transformation from PIM into PSM [13]

Model marking (represented by the activities inside the circle of Fig. 4) is an approach, proposed by OMG for model transformations, that is performed by indicating the PIM elements that are transformed into PSM elements. In the mapping task, relationships between the PIM elements and the PSM one are established. For example, one can create a mapping that relates classes in the model with Java classes. Mappings must comply with the characteristics of both the business models and the programming language.

A PIM element can be related to several PSM elements, and, similarly, a PSM element can be related to several PIM elements. Once a mapping is defined, the execution of the transformation results in code generation.

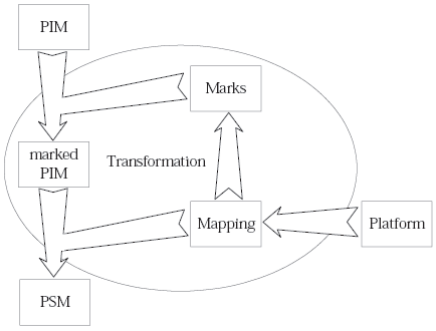

Fig. 4. Model transformation in MDA [13]

2.3 Apache ServiceMix

After model transformations, the resulting PSM model is a software-executable solution. This solution may require the integration with other applications. The integration can be achieved by using hubs or brokers as a middleware between applications. There are some commonly used approaches for enterprise application integration, like the Spring framework [14]. Spring provides a supporting software platform to facilitate the development and execution of business processes by using, among other capabilities, the inversion of control software pattern. Spring can support all the three common layers of a distributed application: user interface, business rules and entities, and the persistence of data. Integration can also be achieved by using ESB-based software frameworks, which allow developing distributed software in a loose-coupled manner. ESBs suggest the usage of a bus, instead of several brokers. Normally, ESBs contain normalized message routers to extract orchestrations from the different software components and place them in a central repository. Orchestrations can be edited without changing the different software components. ESB also include some others handy features, like reliable message buses to guarantee message exchange, or clustering to allow scalability. The core functionalities of an ESB are defined in [15] as being location transparency, transport protocol conversion, message transformation, message routing, message enhancement, security, monitoring, and management. A set of usage patterns for ESBs is presented in [16].

In this work, we use the Apache ServiceMix 4, which is a widely accepted, open source and open standards based ESB solution. ServiceMix may bring benefits to software development projects, like low cost and high quality of the resulting product. It is based on OSGi technology [17] and includes support to the Java Business Integration (JBI) specification [18]. JBI defines a framework for integrating applications, based in added components that interoperate through a method of normalized message exchanges. This method is based in the WSDL 2.0 specification of Message Exchange Patterns (MEPs) [19]. JBI defines two types of components: Service Engines (SEs) and Binding Components (BCs). SEs provide business and processing logic, for instance for processing data or to

implementing business rules. BCs provide communication between the ESB and its exterior, working as a bridge between input and output data.

During compilation time, in order to deploy a component into ServiceMix, a Service Unit (SU), which provides component instantiation to the ESB, is used. Each SE or BC instantiation requires a SU that has the instantiated component definition. A SU can be executed in the ESB, if Service Assemblies (SAs) are used. A SA is a collection of one or more SUs. JBI components are unable to interpret SUs, unless SUs are packaged inside a SA.

3 Selection of a Business Process Language

To assure the quality of the software resulting from a business processes implementation project, it is advisable to select a business process language compatible with the organization where processes will run. To achieve that purpose, we include in this section a comparison between five business process modeling languages: BPMN [20], BPEL [21], XPDL [22], YAWL [23], and Coloured Petri Nets (CPNs) [24].

Several languages are reviewed in [25], namely by describing their technological characteristics and their strengths and weaknesses. Twelve business process languages are compared in [26], according to a representation model proposed in [27], to establish differences to their representational capabilities in the information system domain. The most common approach to compare the modeling capabilities of the business process languages is the set of workflow patterns defined in [28], which shows if the representation of a business process workflow is directly supported by the language.

The process of selecting a business process language should not be restricted to the comparison of the workflow patterns. An organization should not just be concerned with technological issues. Thus, based in [29], we propose that the selection process should be enlarged, being based on the three strategies of the triangle shown in Fig. 5: information systems, organizational, business.

The triangle relates the business strategy with the information system (IS) strategy and the organizational strategy. The selection process for the adopted business process language took into account the information systems strategic triangle. Since IS strategy is related to the definition of the implementation of the business processes in the IS solution, we base our comparison analysis on the workflows that each language supports. Regarding the organizational strategy, we base the comparison on the preferences of the development team. In what concerns the business strategy, our comparison takes into account a set of important aspects related with the alignment of the business process and the business strategy.

3.1 Information Systems Strategy

To compare the business process modeling languages by their added-value in the design and execution of the business processes, a study was performed based

Fig. 5. The information systems strategic triangle (adapted from [29])

in the language functionality, using the workflow patterns defined in [28]. The results of this comparison are described in Table 1. The table derives from a collection of previous comparisons, which can be seen in [30,23,31,32]. If the language supports directly the pattern, it is represented in the table by a "+". If it is supported indirectly, it is represented by a "0". Finally, if it is not supported at all, it is represented by a "-". The workflow patterns descriptions are not detailed in this paper.

Table 1. Workflow Patterns based Comparison

Nr.	Workflow Patterns	BPML	BPEL	XPDL	YAWL	CPN
1	Sequence	+	+	+	+	+
2	Parallel split	+	+	+	+	+
3	Syncronization	+	+	+	+	+
4	Exclusive choice	+	+	+	+	+
5	Simple merge	+	+	+	+	+
6	Multi-choice	-	+	+	+	+
7	Syncronizing merge	-	+	+	+	-
8	Multi-merge	0	-	-	+	+
9	Discriminator	-	-	+	+	-
10	Arbitrary cycles	-	-	+	+	+
11	Implicit termination	+	+	+	-	-
12	Multiple instance without syncronization	+	+	+	+	+
13	Multiple instances with a priori design time knowledge	+	+	-	+	+
14	Multiple instances with a priori runtime knowledge	-	-	-	+	-
15	Multiple instances without a priori runtime knowledge	-	-	-	+	-
16	Deferred choice	+	+	-	+	+
17	Interleaved parallel routing	-	0	-	+	+
18	Milestone	-	-	-	+	+
19	Cancel activity	+	+	-	+	0
20	Cancel case	+	+	-	+	-

3.2 Organizational Strategy

The need for this comparison lies in the fact that the organization's software development team members will be the users of the chosen business process language. It is then necessary to conclude which business process language is the best identified with the profile and skills of its users.

Surveys were performed at Bosch Car Multimedia Portugal to assess the technological skills of the development team, concerning the business process implementation. The aim was to establish a language comparison which can be considered as the most subjective part of our work. In our case, we have based the structure of the survey on a collection of key characteristics of the business process language. We have questioned the development team on their confidence on the following characteristics: workflow modeling, graph-based modeling, XML, Petri nets, pi-calculus, business process modeling languages and notations, service-oriented architecture (SOA), web services, protocols, brokers, ESBs, and threading. Additionally, the team was questioned about BPM issues. This complementary study was helpful to characterize the team regarding its knowledge on the business process to be implemented. Thus, the survey has included questions related to the knowledge and confidence of the team on:

- business processes, activities, key performance indicators, and strategic goals;
- BPM-based tools (e.g., BPM systems, EAI, ERP, CRM);
- business quality management (Total Quality Management, Six Sigma, Balanced ScoreCards);

The answers were given with a level of knowledge about the presented standards, valued from a minimum knowledge of 1 and ending with a maximum of 5. The presented values are relative results obtained by each of the languages in each of the surveys, resulting then the addition of all the surveys classification for each language. The results of the surveys are represented in Table 2 and allow to represent the confidence of the user on using each language (for instance, survey #1 is 76% confident on using BPML, 76% on using BPEL, etc.).

Table 2. Results of the conducted surveys

Survey	BPML	BPEL	XPDL	YAWL	CPN
#1	0.76	0.76	0.76	0.65	0.84
#2	0.70	0.80	0.79	0.94	0.99
#3	0.64	0.74	0.79	0.83	0.78
#4	0.59	0.60	0.79	0.57	0.59
#5	0.62	0.72	0.79	0.70	0.76
#6	0.67	0.88	0.87	0.90	0.95
#7	0.15	0.38	0.35	0.34	0.30
#8	0.34	0.53	0.59	0.49	0.48
#9	0.78	0.79	0.92	0.69	0.75
Total	**5.26**	**6.20**	**6.65**	**6.11**	**6.43**

3.3 Business Strategy

The last comparison relates to the specific aspects of the business environment, in this case referring to the software development industry. Some of these aspects were suggested by [33] as a basis for language supporting tools comparison. Other

aspects are generally relevant concepts for using a language in a business process implementation projects (e.g., language maturity and implementation costs) with the goal of determining the added-value in using one of these languages in the organization. In Table 3 it is represented a set of characteristics, namely: language maturity, usability, existing language implementation tools, online tutorials available, if the language is translatable to another one, the language learning effort, transformability in object-oriented code, implementation costs, portability, interoperability, security, efficiency and data management, and the integration of the language in a ERP (e.g., SAP). In Table 3, for each language a set of business aspects was graded in a scale from 1 to 5. The value for each business aspect was given based on technical specifications and discussion forums. The classification was totally based on our subjective judgement after analyzing the information for each language.

Table 3. Relevant Business Aspects Considered for Comparison

Nr	Business Aspects	BPML	BPEL	XPDL	YAWL	CPN
1	Maturity	4	4	4	3	5
2	Usability	4	4	4	3	3
3	Existing Tools	4	5	3	2	5
4	Online Tutorials	5	5	3	3	5
5	Translatability	5	5	3	4	4
6	Learning Effort	4	4	4	3	3
7	Transformation to OO-code	2	5	3	2	2
8	Implementation costs	5	5	3	5	4
9	Portability	3	5	3	5	5
10	Interoperability	5	5	4	5	3
11	Security	5	5	3	5	3
12	Efficiency	4	5	2	5	5
13	Data Management	4	4	5	4	5
14	Integration in ERP SAP	5	5	3	4	3

After the comparisons of the three dimensions of the information system strategic triangle, the final results were collected in Table 4, where it shows the ordered level of suitability obtained by the languages, and the final result of each language is an overall value of all the executed comparisons.

The language with the best overall result was BPEL, because it was considered the most adequate in the business strategy and also with good classifications in information systems and organization strategies. For the #1 ranking of BPEL should be kept in mind the good result obtained for the particular software development team answering the survey. With different software development teams, the order of business strategies may vary.

Table 4. Final Comparison of the Business Process Languages

Strategy	BPML	BPEL	XPDL	YAWL	CPN
Information System	4	2	5	1	2
Organization	5	3	1	4	2
Business	2	1	5	4	3
Final	4	1	4	3	2

4 Transformation of Business Process Models

4.1 Correlation between BIM States and MDA Models

During a BIM-based business implementation project, it is possible to establish a correlation between the four states of BIM and the states of the PIM and the PSM models. The main characteristics of a business process model that is in the 'Generic', 'Instantiated' or 'Runnable' state are similar to the characteristics of a PIM, because the PF in these states do not include any reference to any platform. During the first three BIM phases ('Selection', 'Definition' and 'Concretisation'), it is not yet decided if the process will be software-executed or not. In fact, BIM suggests that, at the end of the 'Concretisation' phase, a process should be defined to be software implemented in the next phase of the methodology. However, it is also advisable to consider other alternatives, and at least one of the business processes may not require any software implementation. The 'Software-implemented' state corresponds to the PSM, since the process is executed using software, so it must obey specifications of the technological platform.

4.2 Business Process Model at the PIM Level

The third phase of BIM, 'Concretisation', defines a set of requirements the PF must respond in order to conclude that phase. The state of the PF at the end of the 'Concretisation' phase assures that the modeled PIM is ready to be transformed into a PSM. To reach that final state, we first adopt a business process reference model, as proposed in the 'Selection' phase. The use of process reference models in organizations assures adequately modeled business process, responding this way to concerns about business improvement and the quality of the resulting software system.

To exemplify the transformations across the BIM phases, we adopted a business process at the lowest representation level contained in UBK-RM (Unternehmensbereich Kraftfahrzeugausrüstung - Reference Model). UBK-RM is a reference model of the automotive divisions of the Bosch Group. One of the several second-level business processes that are part of the supply-chain process is chosen for the example. This second level process is decomposed hierarchically into some third-level business processes, before the representation at a more detailed level containing activities. We choose the product stock quantities record business process to exemplify our transformation technique due to its simplicity.

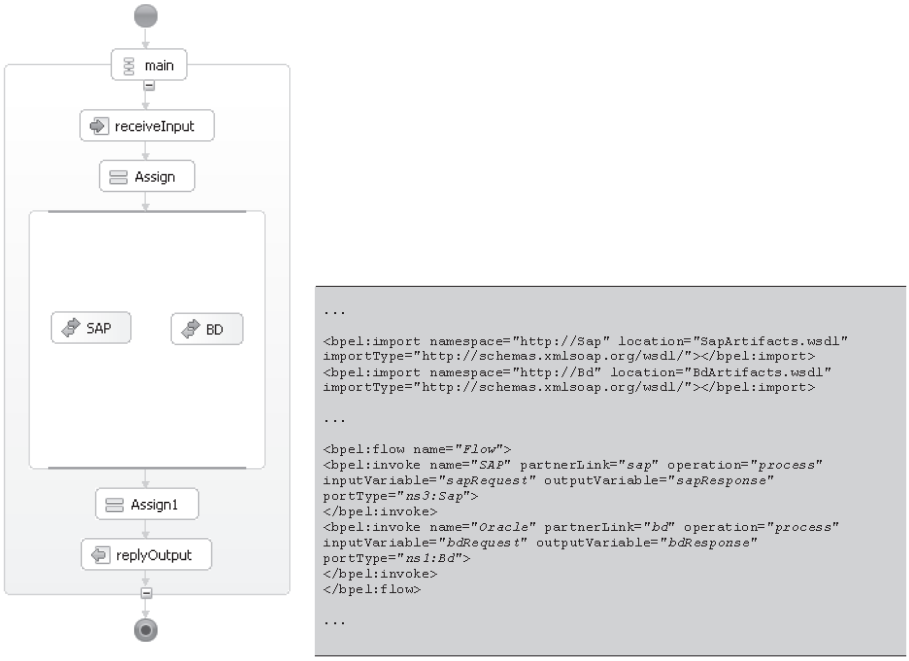

Fig. 6. The runnable Process Framework in BPEL

UBK-RM describes this process with the task that updates the product quantities in stock.

In the 'Definition' phase of BIM, the client describes his/her requirements for this process that has to update the stock of the new product and the stock of the consumed materials for the production of the new product. Thus, the business process has two activities. In the 'Concretisation' phase, the business process model is designed to fit in the information system of the organization. In this particular case, the business process inserts data into an ERP and into a Database Management System (DBMS). The business process was modeled in BPEL, as represented in Fig. 6.

The platform-independent business process illustrated in Fig 6 already embodies the characteristics described in BIM to allow their transformation into a PSM. In our modeled business process, the data received from the client is sent to two components of the information system in a parallel flow: to the ERP system and to the DBMS. The BPEL representation of the business process requires a corresponding WSDL code (Fig. 7) in order to execute correctly.

For this kind of approach, the relevant parts from the WSDL code are the data referring to the service ("lancQuantService"), the port ("LancQuantPort"), the port type ("LancQuant"), the operations ("process"), the input elements ("LancQuantRequest"), and the output elements ("LancQuantResponse").

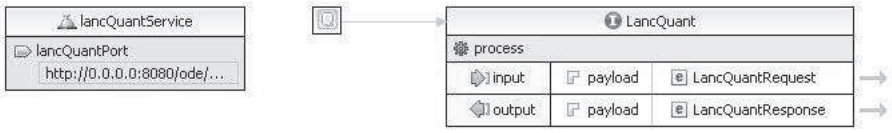

Fig. 7. WSDL representation of the business process

Fig. 8. WSDL representation of the invoked services

The model is completed with the WSDL files of the invoked services, namely the data insertion in the ERP system and in the DBMS (Fig. 8).

4.3 Description of the Platform

One of the required elements for a model transformation is the description of the platform. To define the required functionalities of the platform, BIM proposes the use of Orchestrated Business Objects (OBOs) [5]. For our business process, four OBOs are identified:

- the client component, which gives the command to initiate the process through its invocation;
- the BPEL component, with the process orchestration role, which defines the sequence of service invocations;
- the ERP component, which interfaces with the ERP to execute transactions;
- the DBMS component, which executes the record of the stock quantities.

Based on the ServiceMix JBI-based behavior, three BCs and three SEs are needed to execute the considered business process (Fig. 9). The need for the BCs is justified in order to have connections to the ERP system, to the DBMS, and the request from and the response to the client.

An adequate BC is one that allows the use of Web Services, because the characteristics of the SOAP protocol are more appropriate to send requests to and receive responses from a client. Regarding the connections to the ERP system and the DBMS, the implementation choice is based on the nonexistence of a SE with ERP functions and providing Java Database Connectivity (JDBC) [34]. For the latter two, the execution of the Web Service is made through SEs, a

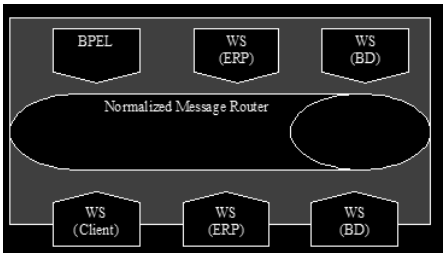

Fig. 9. Platform components to execute the process

"SAP SE" and a "DB SE". It is also required a BPEL execution engine SE. For that, we use the Apache Orchestration Director Engine (ODE) to execute the process workflow and to orchestrate the services. Apache ODE can be used as a SE inside ServiceMix.

4.4 PIM-to-PSM Mapping

In this subsection, we present the required mappings to achieve the PSM in the case study presented in the previous section. We show a set of simple transformation mappings, which can be implemented using a transformation language (e.g., MOF QVT or ATL) or a general-purpose language (e.g., Java or XSLT). For the composition of the transformation mappings, we use elements from the BPEL and the WSDL, the identified OBOs, and the typical JBI-based behavior of the ESB.

In what concerns the required transformations, the WSDL file that composes the PIM is the most used for the transformation, because in the BPEL file only few elements are marked for being transformed. In the BPEL file, the invoked activities, the imported namespaces, and the WSDL files are the elements to be transformed. Regarding the WSDL file, the elements to be transformed are the namespaces, portType, services, and ports. These elements, both from the BPEL and the WSDL files, derive the elements required by the JBI-based model, i.e., they are mapped into elements related to BCs, SEs, BPEL engine (Apache ODE), POJO classes, Java and BPEL and WSDL files.

The invoked BPEL activities expect the response of a service to a request. Therefore, the ESB component that provides this characteristic is the Service Engine. The namespaces, as well as the service data from the WSDL code - portType, service and port -, correspond to the identification data of the created SUs. The PIM-to-PSM mapping is accomplished by relating (SAP) invocations with the "SAP SE" OBO, and "BD" invocations with "BD SE" OBO. For better understanding, the relations are shown and described in Table 5, where each element of the PIM, related with the BPEL and the WSDL files, gives origin to at least one PSM element. In this architecture, PSM elements are a set of JBI-based ESB files (e.g., Java, XML, BPEL, WSDL). The marking of the elements is made from the relationships identified in the mapping (Fig. 10). Its purpose is to assist in the transformation task.

Table 5. Relations of the PIM to PSM Mapping

PIM Elements	PSM Elements
WSDL - Elements from the Request type	Entry parameters of the POJO class of the SU (CXF SEs)
WSDL - Service name ("LancQuantService", "SapService" e "BdService")	Name of the Web Service in the Java file belonging to the SU (CXF SEs)
	targetService of the SU (CXF BCs)
	Name of the service of the respective partnerLink in the SU (BPEL SE "ODE")
WSDL - Port type of the service ("LancQuant", "Sap" e "Bd")	targetInterface of the SU (CXF BCs)
WSDL - Port name of the service ("LancQuantPort", "SapPort" e "BdPort")	targetEndpoint of the SU (CXF BCs)
	Port name of the service of the respective partnerLink in the SU (BPEL SE "ODE")
BPEL - namespaces of the imported services	namespaces in the xbean.xml files and targetNamespace in the JAVA file of the SU (CXF SEs)
	namespaces of the WSDL files which are PartnerLinks in the BPEL file
BPEL - Imported WSDL files	wsdl of the SU (CXF BC), generated by the CXF SE
BPEL - "input" variable and WSDL - Request element	Entry parameters of the client Web Service
	"input" element from the Request of the WSDL
	"input" variable of the BPEL
BPEL - Invoke activity "SAP"	BPEL invoke activity "SAP SE"
BPEL - WSDL file which is "Sap" PartnerLink	Generated WSDL file from the CXF SE
BPEL - Invoke activity "BD"	BPEL invoke activity "BD SE"
BPEL - WSDL file which is "Bd" PartnerLink	Generated WSDL file from the CXF SE
BPEL - "output" variable and WSDL - Response element	Return parameters of the client Web Service
	"output" element from the Response of the WSDL
	"output" variable of the BPEL

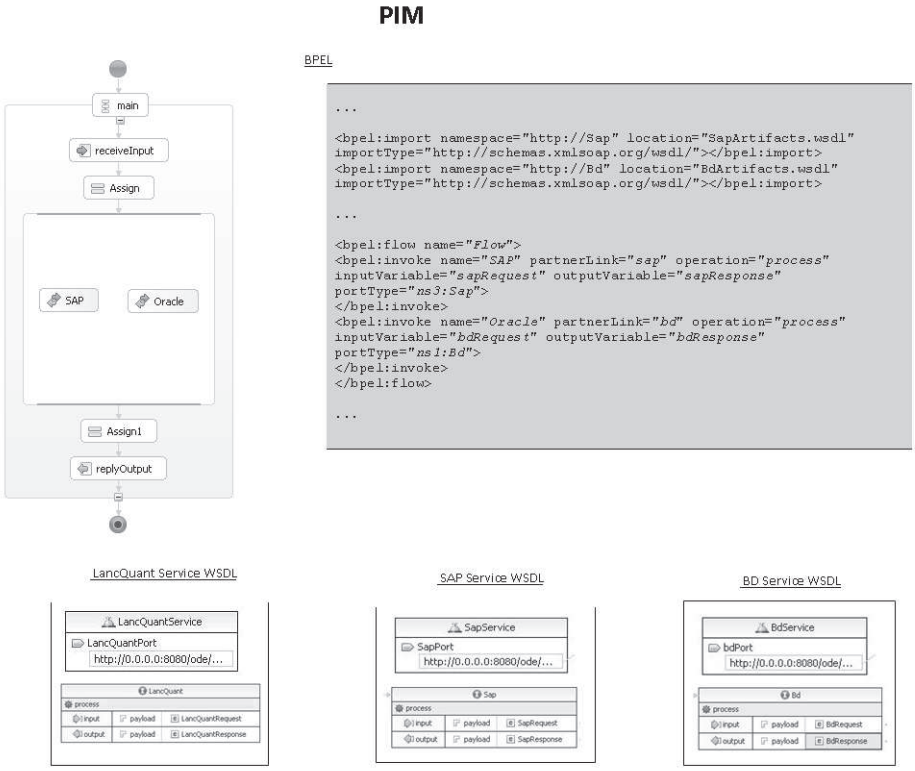

Fig. 10. Representation of the marked PIM elements to be transformed

5 Software-Executable Models at PSM Level

This section presents the technological implementation in the ServiceMix ESB of the model mapping described in the Section 4. We now detail the component deployment to assure that the ESB executes correctly.

5.1 Service Engines

The first step is to define SUs that, after being deployed into ServiceMix, are responsible for creating SEs. The choosen SE type is Apache CXF [35]. It is worthwhile to mention that this choice is related with the absence of a SE directly supporting the ERP functions. Our choice to overcome the absence is to use an interface based on Web Services as a solution. The behavior of the SU is described in a file, called 'xbean.xml', generated from a Maven[2] specific archetype for ServiceMix. The definition of the SU as a SE component is made in the first line of the file, where the namespace is defined (xmlns:cxfse=...), while the rest

[2] `http://maven.apache.org`

```
...
<bean class="pt.bosch.com.teste_cxfse_su.SapService" />
```

```
...
<bean class="pt.bosch.com.teste_cxfse_su.BdService" />
```

Fig. 11. Excerpts of the CXF SE (SAP and DB) SU code

of the file contents describe the necessary elements for the body of the SU. The content of the file is straightforward because it just indicates the location of the Plain Old Java Object (POJO) [36] class relevant for the Service Engine.

In our example, the POJO class 'SapService' (Fig. 11) is exposed via Web Service by the CXF SE.

The SU also contains a Java file, in this case 'SapService.java', which is the Web Service executed by the CXF SE. The 'SapService.java' file uses the SAP Java Connector (JCo) [37] to be able to execute the insertion and the reception of data between the CXF-exposed Web Service and the ERP SAP, obeying correctly to the requirements of a connection to an ERP.

After building the project with Maven 2, the Java class 'SapService' will create a new WSDL file, in our case renamed to 'SapArtifacts' (just to facilitate a simple use by the Eclipse BPEL Design tool). The 'SapArtifacts' WSDL file will be the one who will be invoked in the future by the BPEL PSM file (presented in Fig. 13). Inside that BPEL file, the chosen namespace will be the same as provided by the mapping.

Similarly to the implementation of the interface to the ERP recurring to a CXF SE with a Web Services, the definition of the SU containing the interface to the DB is accomplished in the same manner. The main difference of this SU to the ERP SU is just the definition of the POJO class, which will refer now to the 'BdService' class, the Web Service exposed by the CXF SE. The file 'BdService.java' will contain the code for the JDBC connection, and thus allowing the communication with the database. Also, in this case the WSDL file is generated by a Maven 2 build.

5.2 Binding Components

The configuration of a SU, originating a BC, is similar to the one originating an SE (see Section 5.1). The type of component is defined in the first line of the file 'xbean.xml' (xmlns:cfxbc=...). In this case, the component is a CXF BC, which communicates with the CXF SE, sending and receiving values from a Web Service. The element data that defines the BC must be filled in with the data from the PIMs BPEL and WSDL files, according to Table 5. This correct identification of the component endpoints required by the ESB is then the basis

```
...
<beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       ...
       xmlns:lq="http://LancQuant"
       ...
  <cxfbc:consumer wsdl="LancQuantArtifacts.wsdl"
                  targetEndpoint="lq:LancQuantPort"
                  targetService="lq:LancQuantService"
                  targetInterface="lq:LancQuant"/>
</beans>
```

```
...
<beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       ...
       xmlns:sap="http://Sap"
       ...
  <cxfbc:consumer wsdl="classpath:SapArtifacts.wsdl"
                  targetEndpoint="sap:SapPort"
                  targetService="sap:SapService"
                  targetInterface="sap:Sap"/>
</beans>
```

```
...
<beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       ...
       xmlns:bd="http://Bd"
       ...
  <cxfbc:consumer wsdl="classpath:BdArtifacts.wsdl"
                  targetEndpoint="bd:BdPort"
                  targetService="bd:BdService"
                  targetInterface="bd:Bd"/>
</beans>
```

Fig. 12. Excerpts of the CXF BC (Client, ERP, and DB) SU code

for a proper data routing inside the ESB. In Fig. 12 excerpts of the SU code for the client and the ERP BCs are presented.

Now that the transformation is completed, we present the implemented PSM. Fig. 13 shows the PSM representation in BPEL. In terms of visual notation, it does not suffer modifications related with the PIM represented in Fig. 6, due to the fact that the mapping 'SAP" to 'SAP SE' and 'BD' to 'BD SE' does not require any addition or removal of BPEL activities. An excerpt of the BPEL code containing the transformations suggested in Table 5 is also presented in Fig. 13.

The BPEL business process, to be interpreted by ODE, requires a XML file that describes the connections to the process 'PartnerLinks' so the process can be executable after deployment. In opposition to what happens with the other SUs, in which the 'xbean.xml' file defines its execution, the behavior of ODE is configured by a 'deploy.xml' file. After being correctly defined, the ODE SU is ready to be implemented in the framework. When all SUs are created, they are ready to be packaged in a SA and deployed into the ESB. Each SUs must be compiled and each one originates a compiled file. The definition of the SUs that are part of an SA is described in a 'pom.xml' file. The POM file, generated by a Maven 2 archetype, contains the general information of the project, like its name and location, as well as the identification of the components which will be used in the ESB. After being deployed into ServiceMix, the SUs contained in the


```
<bpel:process name="LancQuant"
        targetNamespace="http://LancQuant"
        xmlns:ns1="http://Sap" xmlns:ns2="http://Bd">
...

<bpel:import namespace="http://Sap" location="SapArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
<bpel:import namespace="http://Bd" location="BdArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>

...

<bpel:invoke name="SAP SE" partnerLink="sap" operation="process" ...
<bpel:invoke name="BD SE" partnerLink="bd" operation="process" ...

...
```

Fig. 13. Platform-specific business process in BPEL

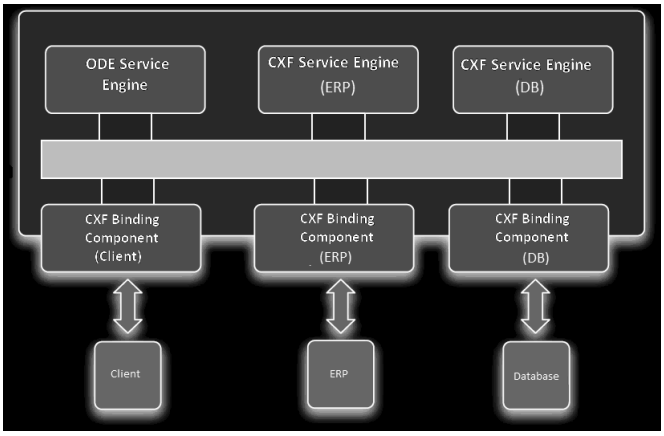

Fig. 14. PSM Final Model in ServiceMix according to JBI

SA will be used as JBI components. Depending of each 'xbean.xml' file, each SU originates either a SE or a BC. ServiceMix ESB is capable of identifying, by using the 'xbean.xml' files, what kind of component is defined in the SU. So, after their deployment, ODE SU becomes ODE SE, CXFSE SU becomes CXF SE and CXFBC SU becomes CXF BC.

The whole PSM, correctly deployed in ServiceMix, is presented in Fig. 14.

6 Conclusions and Future Work

In this paper we showed a set of techniques to use the OMG Model Driven Architecture approach in order to map business process descriptions into software systems. We proposed a mapping between the MDA platform-independent and platform-specific models, and business process models expressed by BPEL. The MDA Service Engines and Binding Components data are effectively obtained from the model transformation.

BPEL is directly executable in software, and can be used during all phases of a software development process for process-oriented organizations. These characteristics can reduce the time to implement a project, as well as the value that an organization receives from using BPEL due to the reduction of functional requirements misunderstandings and losses occurring during normal software development projects. We used a case study to better clarify these perceptions.

We also proposed an holistic technique to properly choose a business process modeling language supporting technology, business, and information systems strategies of an organization. As a result from our work, we defined also a mapping of the business process model states in order to define a correct passage from the third to the last phase of BIM.

The implementation of business process models recurring to ESBs software architectures provides an easy and sound composition of business process that require applications external to the ESB, in a standardized way.

A limitation of BPEL is that it only allows designing fully automatic business processes, i.e., business processes where all activity is executed in the computing domain. In many organizations, most of the business processes - eventually some core business processes - are not fully automatic, requiring human intervention to proceed with its flow of activities. BPEL4People [38], yet with little tool support, allows the addition of the representation of human tasks in BPEL process providing a basis to further develop the proposed techniques, now also interfacing humans. Additionally, we intend to further use the technique to choose an adequate business process modeling language in different organizations and project contexts in order to have a broader quantitative evaluation of the adequateness of each business process modeling language to the current organizations developing software.

References

1. Smith, H., Fingar, P.: Business Process Management – The Third Wave. Meghan-Kiffer Press (2002)
2. Fernandes, J., Duarte, F.: A reference framework for process-oriented software development organizations. Software and Systems Modeling 4(1), 94–105 (2005), doi:10.1007/s10270-004-0063-0
3. Fernandes, J.M., Duarte, F.J.: Using RUP for Process-Oriented Organisations. In: Bomarius, F., Iida, H. (eds.) PROFES 2004. LNCS, vol. 3009, pp. 348–362. Springer, Heidelberg (2004)
4. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture – Practice and Promise. Addison-Wesley (2003)

5. Duarte, F.J., Machado, R.J., Fernandes, J.M.: BIM: A Methodology to Transform Business Processes into Software Systems. In: Biffl, S., Winkler, D., Bergsmann, J. (eds.) SWQD 2012. LNBIP, vol. 94, pp. 39–58. Springer, Heidelberg (2012)
6. OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, version 1.1. OMG Document Number, formal/2011-01-01 (2011)
7. Zhao, W., Hauser, R., Battacharya, K., Bryant, B., Cao, F.: Compiling business processes: untangling unstructured loops in irreducible flow graphs. International Journal on Web and Grid Services 2(1), 68–91 (2006), doi:10.1504/IJWGS.2006.008880
8. Bezivin, J., Hammoudi, S., Lopes, D., Jouault, F.: Applying MDA approach to B2B applications: a road map. In: Workshop on Model Driven Development (WMDD 2004), within the 18th European Conference on Object-Oriented Programming, ECOOP 2004 (2004)
9. Bezivin, J., Dupe, G., Jouault, F., Pitette, G., Rougui, J.: First experiments with the ATL model transformation language: Transforming XSLT into XQuery. In: 2nd OOPSLA Workshop on Generative Techniques in the Context of Model Driven Architecture (2003)
10. Koehler, J., Hauser, R., Sendall, S., Wahler, M.: Declarative techniques for model-driven business process integration. IBM Systems Journal 44(1), 47–65 (2005)
11. Lezoche, M., Missikoff, M., Tininini, L.: Business process evolution: a rule-based approach. In: 9th Workshop on Business Process Modeling, Development and Support, BPMDS 2008 (2008)
12. Rungworawut, W., Senivongse, T.: Using ontology search in the design of class diagram from business process model. In: Proc. International Conference on Computer Science (ICCS 2006), Vienna, Austria, pp. 165–170 (2006)
13. MDA Guide Version 1.0.1, OMG Std.
14. Johnson, R., Hoeller, J., Arendsen, A., Risberg, T., Sampaleanu, C.: Professional Java Development using the Spring Framework. John Wiley & Sons (2005)
15. Rademakers, T., Dirksen, J.: Open-Source ESBs in Action. Manning (2008)
16. Schmidt, M.-T., Hutchison, B., Lambros, P., Phippen, R.: The enterprise service bus: making service-oriented architecture real. IBM Systems Journal 44(4), 781–797 (2005), doi:10.1147/sj.444.0781
17. Alliance, T.O.: OSGi Service Platform Core Specification 4.2, The OSGi Alliance Std. 4, Rev. 4.2 (June 2009), `http://www.osgi.org`
18. Ten-Hove, R., Walker, P.: Java Business Integration (JBI) 1.0, Final release, Technical report, JSR 208 (2005)
19. Web Service Description Language (WSDL), W3C Std., `http://www.w3.org/TR/wsdl`
20. OMG, Business Process Modeling Notation (BPMN) 1.2, Object Management Group Std. OMG Document Number: formal/2009-01-03, Rev. 1.2 (January 2009), `http://www.omg.org/spec/BPMN/1.2`
21. Juric, M., Mathew, B., Sarang, P.: Business Process Execution Language for Web Services, 2nd edn. Packt Publishing (2006)
22. Shapiro, R.: XPDL 2.0: Integrating process interchange and BPMN. In: Workflow Handbook, pp. 183–194 (2006)
23. van der Aalst, W., ter Hofstede, A.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005), doi:10.1016/j.is.2004.02.002
24. Jensen, K., Kristensen, L.: Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer (2009)

25. Ko, R., Lee, S., Lee, E.: Business process management (BPM) standards: a survey. Business Process Management Journal 15(5), 744–791 (2009), doi:10.1108/14637150910987937
26. Recker, J., Indulska, M., Rosemann, M., Green, P.: Business process modeling - a comparative analysis. Journal of the Association of Information Systems 10(4) (2009)
27. Wand, Y., Weber, R.: An ontological model of an information system. IEEE Transaction on Software Engineering 16(11), 1282–1292 (1990), doi:10.1109/32.60316
28. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. QUT Technical report, FIT-TR-2002-02 (2002)
29. Pearlson, K., Saunders, C.: Managing and Using Information Systems, 4th edn. Wiley Publishing (2009)
30. van der Aalst, W., Dumas, M., ter Hofstede, A., Wohed, P.: Pattern-based analysis of BPML (and WSCI). QUT Technical report, FIT-TR-2002-05 (2002)
31. van der Aalst, W.: Patterns and XPDL: A critical evaluation of the XML process definition language. QUT Technical report FIT-TR-2003-06 (2003)
32. Mendling, J., Moser, M., Neumann, G.: Transformation of yEPC business process models to YAWL. In: ACM Symposium on Applied Computing (SAC 2006), pp. 1262–1266. ACM (2006), doi:10.1145/1141277.1141572
33. Helkiö, P., Seppälä, A., Syd, O.: Evaluation of Intalio BPM tool. Special Course in Information System Integration (2006)
34. van Haecke, B.: JDBC: Java Database Connectivity. John Wiley & Sons (1997)
35. A. CXF, Apache CXF: An open-source services framework (2012), `http://cxf.apache.org`
36. Fowler, M., Parsons, R., MacKenzie, J.: Pojo, an acronym for: Plain old java object (2000), `http://www.martinfowler.com/bliki/POJO.html`
37. Schuessler, T.: Developing applications with the SAP Java Connector (JCo). AraSoft, vol. 1 (2002), `http://ARAsoft.de/`
38. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: WS-BPEL extension for people – BPEL4people, Joint white paper, IBM and SAP (2005)