Transformation of Models in an MDA Approach for Collaborative Distributed Processes

Youssef Balouki, Abdessamed Balouki, Mohamed Far, Abdelouahed Kriouile

Transformation of Models in a MDA Approach for Collaborative Distributed Processes

Youssef Balouki , Abdessamed Balouki ,
Mohamed El FAR , Abdelouahed Kriouile

LAVETE, Dep of Mathematics & Computer Science,
University HASSAN 1er Morocco.
{Balouki.youssef, Balouki, El_Far, Kriouile}@gmail.com

Abstract. This paper studies the specification, mapping and the transforming of behavioral aspects of Open Distributed Processing Information Language, within the context of Model Driven Architecture. In order to specify the executable behavior of a system and to make the processes of the Information executable and controllable, the Reference Model for Open Distributed Processing can be used as a meta-model for behavioral specifications. In the Information language the behavior is specified in terms of schema dynamic, processes, actions, state and the relationships between these concepts. In this work we describe how behavior process can be generated exploiting the benefits of a MDA approach. We define the behavior models by using UML profile and their transformations into BPEL artifacts.

Keywords: RM-ODP, Information Language, Behavioral Concepts, BPEL Language, UML Profile, MDA.

1 Introduction

The Reference Model for Open Distributed Processing (RM-ODP) [1-2] provides a framework within which support of distribution, networking and portability can be integrated. It defines a framework comprising five viewpoints, viewpoint language, ODP functions and ODP transparencies. The five viewpoints, called enterprise, information, computational, engineering and technology provide a basis for the specification of ODP systems. The first three viewpoints do not take into account the distribution and heterogeneity inherent problems. This corresponds closely to the concepts of PIM (Platform Independent Model) and PSM (Platform Specific Model) models in the OMG MDA architecture.

In this context we use in this paper the BPEL (Business Process Execution Language for Web Services) (BPEL4WS or BPEL for short) to specify process behavior based on actions, time and states in the context of ODP systems. The BPEL is an XML-based standard for defining how you can combine Web services to implement business processes [8]. It builds upon the Web Services Definition

Language (WSDL) and XML Schema Definition (XSD). This article specifies the behavior processes by the activity diagrams, and generates the corresponding BPEL and computational files to implement that process. This capability is used to highlight some benefits of the Object Management Groups (OMG) Model Driven Architecture (MDA) initiative: raising the level of abstraction at which development occurs; which, in turn, will deliver greater productivity, better quality, and insulation from underlying changes in technology.

The paper is organized as follows. Section 2 introduces, both BPEL and the core behavior concepts (time, action, behavior and process). Section 3 describes and specifies the behavior by the activity diagrams. In Section 4, we define the mapping from the concepts of behavior Information language to BPEL concepts and we present the syntax and the structure of a BPEL Behavior process. We focus on behavioral constrains. A conclusion ends the paper.

2 Preliminaries

2.1 BPEL

Web services are a set of technologies allowing applications to communicate with each other across the Internet. Among the technologies used are the Extensible Markup Language (XML) [8], the Web Service Description Language (WSDL) [10] and the BPEL, also known as BPEL4WS, built on IBM's WSFL (Web Services Flow Language) and Microsoft's XLANG (Web Services for Business Process Design). It combines the features of a block structured process language (XLANG) with those of a graph-based process language (WSFL). BPEL is intended to describe a business process in two different ways: executable and abstract processes. An abstract process is a business protocol specifying the message exchange behavior between different parties without revealing the internal behavior of any of them. An executable process specifies the execution order between a number of constituent activities, the partners involved, the message exchanged between these partners and the fault and exception handling mechanisms.

A composite service in BPEL is described in terms of a process. Each element in the process is called an activity. BPEL provides two kinds of activities: primitive activities and structured activities [13].

2.2 The Behavioral Concepts in Information Language

The individual components of a distributed system must share a common understanding of the information they communicate when they interact [3-5]. Some of these items of information are handled by many of the objects in the system. To ensure that the interpretation of these items is consistent, the information language defines the semantics of information and the semantics of information processing in an ODP system in terms of a configuration of information objects, the behavior of those objects, and environment contracts for the objects in the system.

The information specification comprises a set of related schemata, namely, the invariant, static and dynamic schemata.

An invariant schema is a set of predicates on one or more information objects which must always be true. The predicates constrain the possible states and state changes of the objects to which they apply. ODP also notes that an invariant schema can describe the specification of the types of one or more information objects that will always be satisfied by whatever behaviour the objects might exhibit.

A static schema is a specification of the state of one or more information objects, at some point in time, subject to the constraints of any invariant schemata.

A dynamic schema is a specification of the allowable state changes of one or more information objects, subject to the constraints of any invariant schemata.

We consider the basic set of modeling concepts necessary for behavior specification:

- Action: a model of something that happens in the real world. An action in the information viewpoint is associated with at least one information object.
- Interactions: an action always takes place with the participation of the environment of the object. Objects can only interact at interfaces.
- Behavior of an information object: a collective behavior composed of the actions in which the objects participate in fulfilling the roles of the system, together with a set of constraints on when these actions may occur, it may be interesting to specify which actor initiates that action.
- Process: identifies an abstraction of the behavior that includes only those actions that are related to achieving some particular sub-objective within the system. Processes decompose the behavior of the system into steps.

We represent a concurrent system as a triple consisting of a set of behavior, a set of process and a set of action. Each behavior is modeled as a finite or infinite sequence of interchangeable behavior and actions. In figure 1, we define a model to be the ODP information viewpoint specification. That is, a set of information objects, and their relationships and behaviour.

The concept of time dependence is given in "Specification concepts" (RM-ODP 2.9), we define the semantics of OCL precondition and postcondition by applying the minimal sets of instances after execution of operation [14] [16].

```
context  Time inv :
forall(o:InformationObject ,t:Time | t.instant ->notEmpty implies o.state ->notEmpty)

context Precondition inv :
Forall (prec: Dynamicschema.Precondition , o : InformationObjectlexists( s : State) |
o.mappedTo = prec and o.state_start = s)

context  Postcondition inv :
forall (postc: dynamicschema.Postcondition , o :  InformationObject | exists(s : State) |
o.mappedTo = postc and a.state_end = s)
```

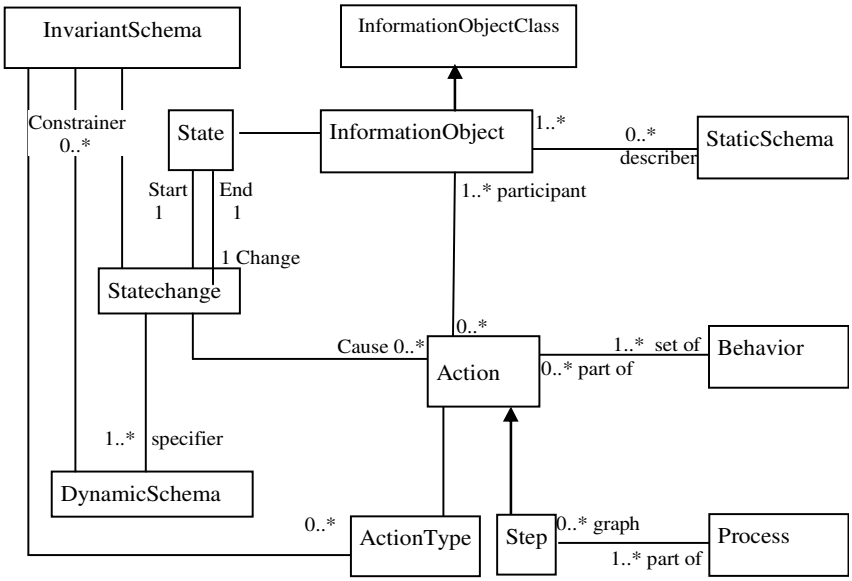

Fig. 1. Core Behavior Concepts

3 UML Profile for Behavior Process

Taking an object-oriented approach, the Unified Modeling Language (UML) is often used to model the relevant aspects of the behavior. In UML, an object is an entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships. The behavior of UML object expressing an ODP information object is expressed by state machines. The scope of this article is mainly centered on stereotypes. Stereotypes are a way of categorizing elements of a model. We can combine a set of these stereotypes in a Profile. A UML Profile is used to define a specific set of extensions to the base UML in order to represent a particular domain of interest.

This section introduces a UML Profile which supports modeling with a set of semantic constructs that correspond to those in the Business Process Execution Language for behavior in Information language (see table 1).

Table 1. Behavior concepts to UML mapping overview

Behavior Concepts	Profile Construct
Process	<< process>> class
Behavior	Activity graph on a <<process>> class
Action	<<metaclass>> signal
Role	<<partner>> class
static Schema	<< metaclass >> statemachine
Invariant Schema	<< metaclass >> constraint
dynamic Schema	<< metaclass >> package

In the UML profile, a process is represented as a class with the stereotype <<Process>>. The stereotype «IV_Process» extends the metaclass Activity with multiplicity [0..1]. It is intended to capture the semantics of a Process in the RM-ODP information language. The attributes of the class correspond to the state of the process (variables in BPEL 1.1). The UML class representing the behavior process is shown in Figure 2.

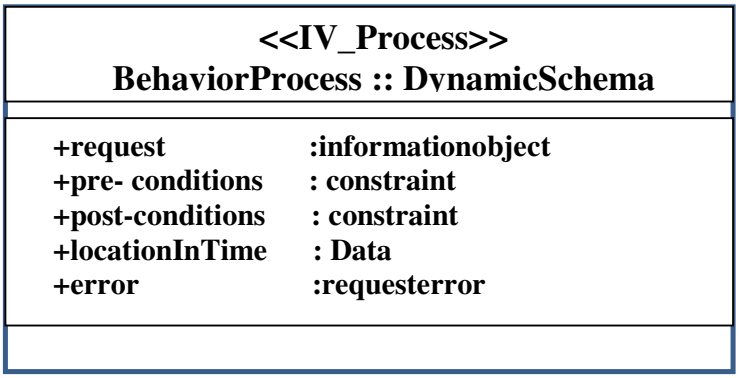

Fig. 2. UML Profile for the Behavior Process

4. Generating a BPEL Process from a UML Model

The Model Driven Architecture (MDA) [15] provides an approach for specifying a system independently of the platform that supports it; specifying platforms; choosing a particular platform for the system; and transforming the system specification into one for a particular platform.

4.1 Mappings between UML and BPEL

The UML profile for automated behavior processes expresses that complete executable BPEL artifacts can be generated from UML models. Table 2 shows an overview of mapping from the profile to BPEL covering the subset of the profile introduced in this article [15].

Table 2. UML to BPEL mapping overview

Profile Construct	**BPEL Concept**
<< process>> class	BPEL process definition
Activity graph on a <<process>> class	BPEL activity hierarchy
<<process>> class attributes	BPEL variables
Hierarchical structure and control flow	BPEL sequence and flow activities
<<receive>>,<<reply>>,<invoke>>activities	BPEL activities

BPEL is an XML representation of an executable process which can be deployed on any process motor. The atomic element of a process BPEL is an "activity", which can be the send of a message, the reception of a message, the call of an operation (sending of a message, makes an attempt of an answer), or a transformation of data.

A process BPEL defines, in XML, the activities realized by the framework of the behavior process execution. In the following we describe its structure and syntax.

< IV_behavior >
- < roles /> → definition of the actors
- <containers/> → definition of the containers of the data
- <invariant schema /> → A set of predicates which must always be true.
- <static schema /> → A configuration of information objects.
- <transitioncondition>
  - <dynamic schema /> → A state changes of one or more information objects.
- </transitioncondition>

</IV_behavior >

<IV_**process** >
- < partners /> → definition of the partners (actions)
- <containers/> → definition of the containers of the data
- <sequence />
  - <receive /> → reception of a request
  - <assign /> → transformation of the data
  - <invoke /> → call of an action
  - <reply /> → sending of an answer

</sequence>
</IV_**process**>

```
<schema>   name = "nameschema"
   <process name ="process"/>
   < action  name =  "action"/>
       <constraint type ="pre-conditions"/>
      <constraint type ="post-conditions"/>
 </schema>
```

4.2 Transforming the process Specification into BPEL

Model transformation is the process of converting between two models describing different aspects or levels of detail of the same thing: UML model files which can be opened and modified with tools [12], and XML files containing the XMI version of the UML models and which are exported by them. In figure 3, we can see that this corresponds to the UML models, or the XMI output of these tools [8-9].

Figure 3 uses a UML Activity Diagram to show the overall process of transforming the files; the information specification is related to a computation independent model (CIM); The information and computational specifications together form a (set of) platform independent model(s) (PIM).The main stages are:

1. Specifying the UML model (CIM)
2. Exporting the UML Diagrams to XMI (PIM)
3. Generating the BPEL process, Actions, and behavior files(PIM)
4. Creating a Database Information Object
5. Deploying these on the BPEL motor (PSM).

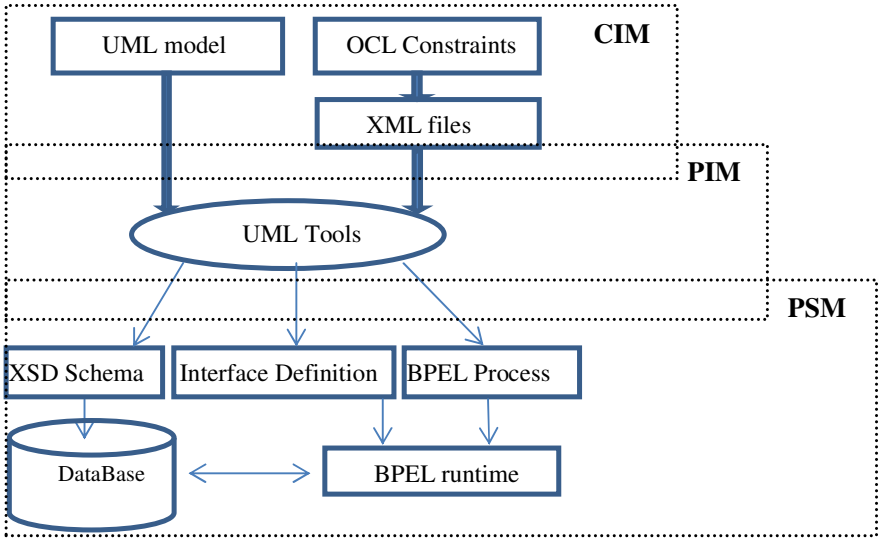

Fig. 3. Developing a process

5 Conclusion

This work introduces the modeling, mapping and transformation of behavioral aspects of Open Distributed Processing (ODP) Information Language, within the context of Model Driven Architecture (MDA). In particularly, we have demonstrated how to model a UML profile for automated behavior processes with UML to BPEL translator. The profile allows developers to use UML skills and tools to develop behavior processes using BPEL. This approach enables service-oriented BPEL components to be incorporated into an overall system design utilizing existing software engineering practices.

References

1. ISO/IEC. : Basic RM-ODP-Part1: Overview and Guide to Use. ISO/IEC CD 10746-1, 1994
2. ISO/IEC. : RM-ODP-Part2: Descriptive Model. ISO/IEC DIS 10746-2, 1994.
3. ISO/IEC. : Use of UML for ODP system specifications. ISO/IEC 19793, 2006.
4. ISO/IEC. : The ODP Trading Function. ISO/IEC JTC1/SC21, 1995.
5. ISO/IEC. : RM-ODP Enterprise Language. ISO/IEC 15414, July 2006.
6. J. Rumbaugh and all. : The Unified Modeling Language. Addison Wesley, 1999.
7. M. Bouhdadi, Y.Balouki. : Meta-modelling Semantics of Behavioral Concepts for Open Virtual Enterprises.ECC 2007, Athens 25-27 Sep, Springer Verlag.
8. Y.Balouki and M.Bouhdadi. : Using BPEL for Behavioural Concepts in ODP Enterprise Language. Virtual Enterprises and Collaborative Networks, IFIP, Vol. 283, pp. 221-232, Springer, 2008.
9. E. Evans, R. France, K. lano, B. Rumpe. : Meta-Modeling Semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, Behavioral specifications for businesses and systems, Kluwer Academic Publishers, Norwell, MA, September 1999. Chapter 4
10. B. Rumpe.: Agile Modeling with UML, '' LNCS vol. 2941, Springer, 2004, pp. 297-309.
11. L. Briand. : A UML-based Approach to System testing. LNCS vol. 2185. Springer, 2001, pp. 194-208,
12. B. Rumpe. : Executable Modeling UML. A Vision or a Nightmare?. In: Issues and Trends of Information technology management in Contemporary Associations, Seattle, Idea Group, London, pp. 697-701.
13. Dimitris Karagiannis and al.: Business-oriented IT management developing e-business applications with E-BPMS. ICEC 2007, 97-100
14. M. Broy. : Formal treatment of concurrency and time. Software Engineers's Reference Book, Oxford Butterworth-Henenmann (1991).
15. Keith Mantell: From UML to BPEL Model Driven Architecture in a Web services world. Report IT Architect, IBM 2003.
16. M. Bouhdadi, Y. Balouki. :'Meta-modelling Semantics of Behavioral Concepts for Open Virtual Enterprises. : ECC 2007, Athens 25-27 Sep, Springer Verlag.