Towards Standards for Modelling and Executing Collaborative Business Processes

Oana NICOLAE [1], Mirel COŞULSCHI[2], Adrian GIURCĂ[1] and Gerd WAGNER[1]

[1],Department of Internet Technology
Institute of Informatics
Brandenburg Technical University at Cottbus, Germany
nicolae, giurca, G.Wagner@tu-cottbus.de,
[2]Faculty of Mathematics and Computer Science,
Department of Computer Science,
University of Craiova, Romania
mirelc@central.ucv.ro

Abstract. Business Processes are becoming some valuable assets for the actual industry landscape. Web Services based on the Service Oriented Architecture (SOA) framework provide an appropriate, technical foundation for making Business Processes accessible within and across enterprises and assure the independence of tools and infrastructure. Business Process Modelling (BPM) and SOA facilitate the next phase of Business Process evolution by enabling dynamic processes. BPM solutions simplify the modelling, design, monitoring and enactment of complex processes based on decision making. The paper intends to be a review on BPM techniques: BPDM/BPMN and BPEL/WS-CDL by describing them using UML models, as the missing link between the Web Services and Business Processes. Our approach also argues for the need of standardization on BPM field.
Keywords: business process modelling, UML, BPM, MDA, BPDM, BPMN, BPEL, WS-CDL, standardization, orchestration, choreography.
Math. Subject Classification 2000: 68T30; 68T35

1 Introduction

The need of flexible and scalar business process environments corroborated with the growing demand for dynamic integration with other services are the main factors that motivate the continuous development of Web Services and Service Oriented Architecture (SOA). These technologies represent a pivot point in joining intelligent software infrastructure with the purpose to introduce and enable Web Services and SOA based inter-operability protocols with modeled business processes.

Well accepted Web Service standards like UDDI[1] (OASIS[2]), WSDL[3] and SOAP[4] (W3C[5]) represent the first step to solve the integration problem. But these technologies are not further capable to provide an appropriate environment where business processes can be developed, shared and managed.

Here is the point where BPM technologies are used. They simply model, compose and treat Web Services as business processes of an enterprise. BPM suite is process-oriented, theoretically with no need for programming and therefore suitable for business analysts. The business model is not only a business process documentation, but a design created as a source for executable process implementations.

As an answer to the necessities of inter-operability and standardization (i.e. an unique way of understanding the description of business processes both at the design and at the implementation/enactment levels), Business Process Modelling Notation (BPMN [3]), Business Process Definition Metamodel (BPDM [1]), Business Process Execution Language (BPEL [2]) and Web Service Choreography Description Language (WS-CDL [11]) specifications were developed by OMG[6], OASIS and W3C, respectively, having as target the actual business market.

Our paper is structured as follows: Section 2 presents the motivation for the chosen subject in the context of BPM market and Model Driven Architecture (MDA[7]) approach and its levels of abstraction. Section 3 introduces the OMG's BPDM and BPMN standards and provides brief explanations of the involved concepts by means of a BPMN metamodel. In Section 4 we present the main languages from the actual literature which deal with business process collaborations i.e. BPEL and WS-CDL. For a better understanding of their basic concepts, we provide extracts from their UML metamodels. Section 5 discusses and outlines the main conclusions vis-a-vis our approach.

2 Motivation

The main objective of modelling the business processes is ensuring them *consistency* and *rigor*. As BPM market becomes more mature, there is a visible, growing interest in the literature on subjects surrounding BPM based Web Services and Web Services Composition techniques: *orchestration* and *choreography*.

In orchestration, a central process takes control over the involved Web Services and coordinates the execution of different operations on the Web Services

[1] UDDI - http://www.oasis-open.org/committees/uddi-spec/doc/
[2] OASIS - http://www.oasis-open.org/
[3] WSDL - http://www.w3.org/TR/wsdl/
[4] SOAP - http://www.w3.org/TR/soap/
[5] W3C - http://www.w3.org/
[6] OMG - http://www.omg.org/
[7] MDA - http://www.omg.org/mda/

involved in the operation. The involved Web Services do not know, and do not need to know that they are involved into a composition and that they are a part of a higher business process. Only the central coordinator of the orchestration (i.e. BPEL process) knows these details, so the orchestration is centralized with explicit definitions of operations and the order of invocation of Web Services.

On the other side, choreography is a collaborative effort focused on exchange of messages. All participants of the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges.

MDA approach introduces a set of basic concepts such as: *model*, *metamodel*, *modelling language* and *transformation*. It also divides the known business process modelling languages into three distinct levels. Abstract business process modelling belongs to the Computational-Independent Model (i.e. CIM - domain model or business model) e.g. business process metamodel provided by BPDM specification. Platform-Independent Model (i.e. PIM) is also used to describe a system: it is more specific and lower-level e.g. BPMN (graphical notation) or WS-CDL (textual notation). Platform-Specific Model (i.e. PSM) deals with the enactment and includes software implementation details e.g. BPEL specification.

OMG's MDA standards such as UML and Meta Object Facility (MOF[8]) provide metamodels that claim to define the standard, but they only focus on the abstract syntax (i.e. the vocabulary). Even it can not provide an executable model, UML is used to define a rigorous and precise specification to which any executable program language must conform ([7]). Further in this paper, we use UML models to briefly describe the core concepts of BPMN 1.1, WS-CDL 1.0 and BPEL 2.0 specifications for collaborative business process modelling and enactment, respectively.

3 OMG's specifications for business process modelling: BPDM and BPMN

OMG's Business Process Definition Metamodel (BPDM) was started in 2003 as a metamodel for general purpose business processes, initially without a notation, and later aligned with BPMN 1.1 Specification. Its purpose is to allow tools inter-operability through a common process serialization mechanism. It also claims to provide support for SOA and to enable business rules integration within business processes. One of its useful characteristics is the ability to enable a serialization mechanism for BPMN and to provide a MOF-based metamodel (i.e. a precise semantics) for BPMN.

Business Process Modelling Notation (BPMN) was proposed as the graphical modelling component of the original set of specifications introduced by

[8] MOF - http://www.omg.org/technology/documents/formal/mof.htm

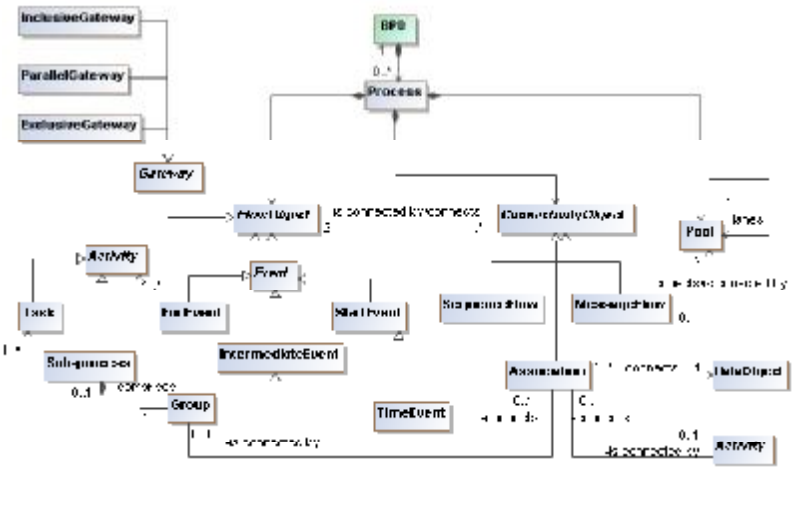

Fig. 1. BPMN 1.1 - Core Concepts Metamodel.

the former Business Process Management Initiative (BPMI[9]), now part of the OMG.

The aim of BPMN is to provide a notation that is easily understandable by all business users, from the business analysts which create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor them.

The output of a BPMN model was initial intended to be BPML[10] (Business Process Modelling Language), whose purpose was to translate the graphical model into a machine-readable format, in order to allow the interchange of the business processes definitions. Very similar with BPEL, this metalanguage developed by BPMI was abandoned, as its description is not bound to WSDL and the communication protocol is left to a BPML compliant implementation.

BPMN 1.1 uses Business Process Diagrams (i.e. BPDs) to encapsulate its specific constructs. In the followings, we will describe a BPD from the perspective of the BPMN metamodel constructs (see Figure 1). A BPD can contain many BPMN `processes`. We focused only on the representation of private BPMN business processes i.e. processes that allow the visualization of the entire workflow. Each process comprises, at its turn, many `Pools` (compartmented, if the logic imposes, in many `lanes`) which map the business partners involved in

[9] BPMI - http://www.bpmi.org/
[10] BPML - http://www.ebpml.org/bpml.htm

a business process. A process also contains BPMN specific constructs, such as: `flow objects` (events, tasks or gateways), `connectivity objects` (sequence flows, message flows, associations) and `artifacts` (e.g. data objects). A task representing the atomic activity in BPMN 1.1 is identified by the `taskType` attribute ([9]). Many tasks can be grouped into sub-processes, for enabling the reusing of some task groups.

`Pools` are connected through message flows that carry the information and provide means for business partners communication. BPMN proves to be an evolving language (i.e. two Specifications for BPMN 2.0 are competing for approval [5] [6]). OMG consortium presents BPMN as a powerful tool for representing business processes collaborations. Despite their claiming, the current specification still lacks precision (i.e. no metamodel defined) also in representing business process choreographies. It only allow a well defined representation of business processes orchestrations and through a complicated mapping to BPEL language provides means for executing the obtained business process collaboration. Moreover, the choreography specification is a first target for BPMN 2.0 specification proposals.

4 BPEL and WS-CDL - standards for collaborative business processes

Business Process Execution Language (i.e. BPEL 2.0) is a portable execution format that relies exclusively on Web Services (i.e. WSs) resources and on XML data. It specifies business processes as a set of interactions between WSs. Nowadays, BPEL is known as the dominant orchestration language on the business market. BPEL involves only automating processes which invoke WSs and are defined using Web Services Definition Language (i.e. WSDL). The BPEL Specification, sustained by IBM, Microsoft, BEA, SAP, and Seibel Systems, defines a model and grammar for describing the behavior of a business process based on interactions between the process and its partners. The use of WSs is essential to BPEL, which allows the modelling of executable and abstract processes, and is layered on top of several XML specifications: WSDL 1.1, XML Schema 1.0, XPath 1.0, XSLT 1.0, WS-Addressing.

A BPEL file is an XML document that conforms to the BPEL XML Schema. The BPEL file is interpreted at runtime by a BPEL processor (i.e. BPEL engine) that identifies keywords or activities and executes them as defined in the BPEL file. Our metamodel has at its basis the XML Schema provided by OASIS [11].

The BPEL industry standard is now maintained by OASIS organization, but at its origins, BPEL represents a joined effort from IBM's Web Service Flow Language (WSFL) and Microsoft's XLANG initiatives. It was developed during years (now it is at 2.0 version) as a complex and expressive business process

[11] OASIS XML Schema - `http://docs.oasis-open.org/wsbpel/2.0/OS/process/executable/ws-bpel_executable.xsd`

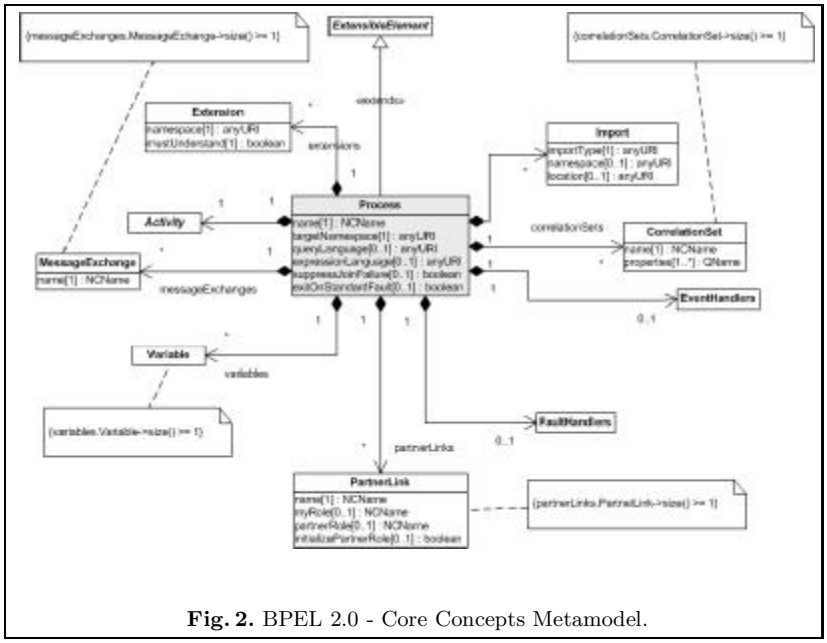

Fig. 2. BPEL 2.0 - Core Concepts Metamodel.

execution language providing an unique, rich support for communication and workflow patterns.

However, this complexity also conducts to many overlapping BPEL constructs that makes the semantics of the language to be sometimes unclear and leads to misunderstandings. A simplification of the language is then needed. Still, no actual tools offer a BPMN model execution, but only a BPMN to BPEL translation and then the execution of the resulting business process.

BPEL supports two different types of business processes:

- Executable processes allow the specification of the exact details of business processes. They can be executed by an orchestration engine.
- Abstract business protocols allow the specification of the public message exchange between participating parties only. They do not include the internal details of process flows and are not executable.

An executable BPEL process describes the complete, internal workflow of a business process and can express conditional behavior, for example, a Web Service invocation can depend on the value of a previous invocation. It can also construct `loops`, declare `variables`, `copy` and `assign` values, define `fault handlers`. By combining all these constructs, it can define complex business processes in an algorithmic manner. Figure 2 represents an overview of the entities that are comprised by a BPEL executable business process (i.e. BPEL program). We

used composition oriented arrows to express the compositions for a `Process` type class. The only mandatory element from the `Process` class composition structure is the `Activity` class that contains the entire logic of the BPEL program, revealing this way the procedural programming aspect of the language.

A particular concept belonging to BPEL is `PartnerLink`. A partner link is used to express the interactions between an executable process and its partners (i.e. a WS provider/consumer or a two-way long running asynchronous conversation, P2P, where both the BPEL process and its partner are providers and also consumers).

Interactions are defined by means of Web Service `activities` such as: `receive`, `invoke` or `reply`. The syntax of these activities implies the presence of a `partner link`, its `port type` and `operation`. The port type comes from the associated `partner link type`. meaning that the process can accept incoming message data from the service playing the partner role defined in the partner link. Inside a partner link specification there must be at least one role declaration.

```
 1.<process name="TravelInfo" targetNamespace="http://maps.org/bpel/travelinfo"
 2.   xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
 3.   xmlns:map="http://maps.org/wsdl/mapinfo"
 4.   xmlns:wea="http://weather.org/wsdl/weatherinfo"
 5.   xmlns:rest="http://restaurant.org/wsdl/restaurantinfo"
 6.   xmlns:trav="http://maps.org/wsdl/travelinfo/">
 7.  <partnerLinks>
 8.    <partnerLink name="TravelInfoPL" partnerLinkType="trav:TravelInfoPLT"
 9.        myRole="TravelInfoServiceProvider"/>
10.    <partnerLink name="MapInfoPL" partnerLinkType="map:MapInfoServicePLT"
11.        partnerRole="MapInfoServiceProvider"/>
12.          ...
13. </partnerLinks>
```

Usually, a BPEL business process has at least one invoked partner link, because it will most likely invoke at least one web service, as can be seen in the above example.

BPEL uses the `variable` concept in order to receives/manipulates/sends data. BPEL relies exclusively on XML data, so the variables must be defined as WSDL message types, XML schema types or XML schema elements. They can be declared globally (as a child of process element) or locally (inside a `scope` activity). Variables in `receive`, `onMessage`, `onEvent`, and inbound `invoke` activities are automatically initialized. The initial value of a variable is validated against the schema or WSDL definition during process execution. A variable can also be initialized when it is declared, or later using an `assign` activity. A variable can be validated using a `validate` activity.

```
14.<variables>
15. <variable name="inboundTravelInfoRequest" messageType="trav:getTravelInfo"/>
16. <variable name="outboundTravelInfoResponse"
17.     messageType="trav:getTravelInfoResponse"/>
18. <variable name="outboundMapInfoRequest" messageType="map:getMapInfo"/>
19. <variable name="inboundMapInfoResponse"
20.         messageType="map:getMapInfoResponse"/>
21.</variables>
```

Regarding the internal structure of any BPEL process (i.e. procedural programming language with specific Web Service constructs) the reader can see it is composed from basic elements called `activities`.

There are two kinds of activities: `basic activities` and `structured activities`. Basic activities represent primitive types of activities such as: `invoke/receive/ reply` activities - that we have distinctly grouped into WS-Activity class - together with `assign` (i.e. copy data from one data container to another one; a data container can be represented by a variable or by a data literal; an assign activity can update any number of variables), `wait` (i.e. keep the business process idle for some specified time interval; when the specific time is reached, the business process continues its execution), `throw/rethrow` (i.e. expose the errors that can appear during a business process execution: faults that should be catch by fault handlers), `exit` (i.e. abandon the execution of the business process instance and all in use activities must terminate without any fault handling or compensation behavior) and `empty` (i.e. this activity has no effect; it is used for synchronizing concurrent activities).

Using the WSActivity concept inside our UML class diagram, we classify the types of BPEL activities that implies the communication between the business process partners (i.e. exchanged messages) in synchronous and asynchronous types of communication. The WS activities are: `invoke`, `receive`, `reply`.

```
22.<invoke name="invokeMapInfo" partnerLink="MapInfoPL"
23.    portType="map:MapInfoInterface" operation="getMapInfo"
24.    inputVariable="outboundMapInfoRequest"
25.    outputVariable="inboundMapInfoResponse"/>
26.<reply name="replyTravelInfoResponse" partnerLink="TravelInfoPL"
27.    portType="trav:ClientInterface" operation="getTravelInfo"
28.    variable="outboundTravelInfoResponse"/>
```

`Invoke` activity calls an operation of a partner. A BPEL process immediately continues its execution after invoking a one-way operation. When a BPEL process invokes a request-response operation, the activity blocks the execution of the BPEL process until a response message or a fault message is received.

`Reply` activity is used in combination with a receive activity to provide a request-response operation. This activity sends a reply message or a fault message as a response to a message received by the receive activity.

The code samples presented above represents a common way to obtain a SOA-based composition (i.e. BPEL orchestration) of Web Services. Its basic architecture stack is SOAP and WSDL compliant (i.e. any BPEL program is in fact a Web Service and uses WSDL for describing the services orchestration, therefore the transport protocol is SOAP) and comprises components such as: the registry for Web Services publication, the security layer that maintain some authentication of the involved partners, the reliable messaging layer that guarantees the exchange of information between corresponding partners, the context-coordination-transaction layer that enable a global agreement protocol, the business process languages layer with the purpose of describing the business process inner, execution logic. A further layer in the stack of business process composition the actual literature focus on, is represented by the concept of business processes choreography (see Figure 3). In this context, WS-CDL is the promoted language, developed and sustained by W3C community. It is described as a solution for automating P2P collaborations, within or across

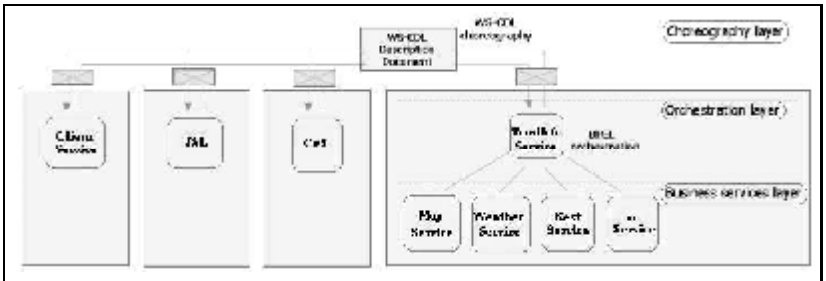

Fig. 3. WS-CDL Choreography Architecture.

organizations trusted domains, by clearly defining the rules of participation that are jointly agreed. As a language it has the following characteristics:

- It is not an executable language, therefore not depending on any specific business process implementation.
- A WS-CDL compliant choreography is represented by an XML-based document describing a multi-participant collaboration engagement. The `roleTypes`, `relationshipTypes`, `channelTypes` are WS-CDL concepts that depict the participants involved in the choreography and their coupling.
- Each participant to the choreography can be implemented independent of the supporting platform and using different mechanisms such as business process languages (i.e. BPEL, BPML) or programming languages (i.e. Java) or human controlled software agents.
- WS-CDL is also layered on top of the following XML specifications: XML 1.0 (Namespaces, Schema), XPath 1.0, WSDL 1.1/WSDL 2.0.

For a better understanding of the concepts we construct a WS-CDL 1.0 metamodel based on the informal description that WS-CDL Specification provide. The root element is the `package` construct (see Figure 4.). Its authoring properties are defined by the attributes: `name` (mandatory), `author` and `version` (optional). The `targetNamespace` provides the namespace associated with all WS-CDL type definitions contained in the choreography package. Inside a choreography package we can define many choreographies and also we can include other WS-CDL type definitions that are defined in other choreography packages i.e. *xi: include href="otherChoreography.xml"*.

The participant of a WS-CDL choreography is modelled by a `participantType` construct. It groups the behavior that is to be implemented by the same logical entity or abstract organization. In this context, `roleType` concept is used in order to enumerate the potential `Behavior` a participant can exhibits in order to interact. A `Behavior` element may define an optional `Interface` element that represents an analogous WSDL interface type. `relationshipType` represents the jointly agreed commitments between the participants. A `relationshipType`

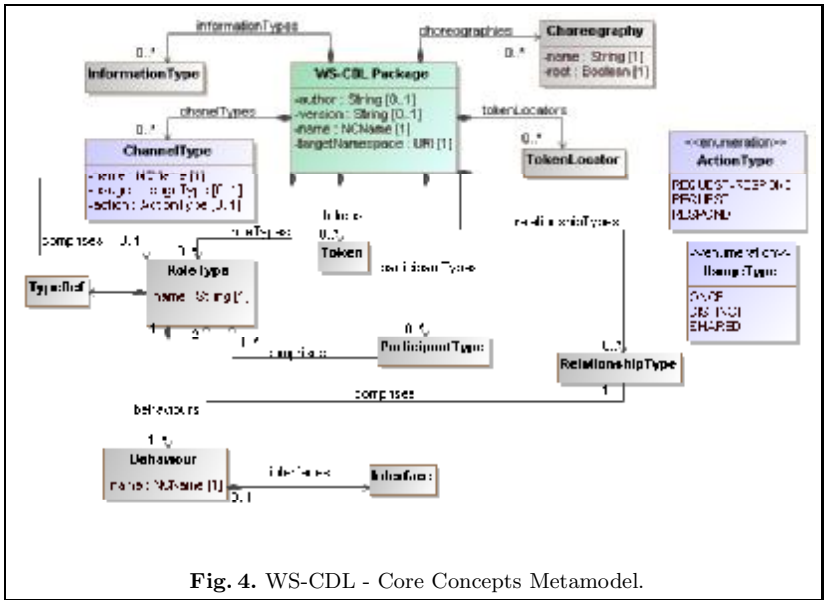

Fig. 4. WS-CDL - Core Concepts Metamodel.

must have exactly two `roleTypes` defined, each specified by `TypeRef` construct which must reference the name of the appropriate role type. Inside of a `participantType`, one or more `roleType`(s) may be defined corresponding to the roles that the participant must implement. The `ChannelType` construct specifies how and where the message information (both static and dynamic) is exchanged between `ParticipantTypes`.

5 Conclusion and Future Works

In this paper we have discussed the importance of BPM techniques for the actual business market by outlining the relevance of the relationship between Web Services and Business Processes. The paper also sustains the need of standardization on BPM and Web Service related fields. Briefly explanations about BPMN 1.1, WS-CDL 1.0 and BPEL 2.0 core concepts were given by means of UML meta-models. The importance of UML for modelling software system is also mentioned and argued. Future works on this area covers a technical report on BPEL 2.0 Specification and a complete meta-model for WS-CDL 1.0.

References

1. ***, (BPDM 1.0) Business Process Definition Metamodel (OMG), `www.omg.org/cgi-bin/doc?bei/03-01-06`, 2006.

2. ***, (BPEL 2.0) Business Process Execution Language (OASIS), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html, 2006.
3. ***, (BPMN 1.1) Business Process Modeling Notation 1.1, (OMG), http://www.omg.org/cgi-bin/doc?dtc/2007-06-03, 2007.
4. **R. Khalaf, A. Keller, F. Leymann**, Business Processes for Web Services: Principles and Applications, IBM Systems Journal, Celebrating 10 Years of XML Volume 45, Number 2, 2006.
5. ***, (BPDM Team) Adaptive, Axway Software, EDS, Lombardi Software, MEGA International, Troux Technologies, Unisys, BPMN 2.0 Specification Proposal, http://www.omg.org/cgi-bin/doc?bmi/08-02-03, 2008.
6. ***, (BEA, IBM, Oracle, SAP), BPMN 2.0 Specification Proposal, http://www.omg.org/cgi-bin/doc?bmi/08-02-06, 2008.
7. **T. Clark, P. Sammut, J. Willans**, Applied Metamodeling. A Foundation for language driven development. (Second Edition), Ceteva 2008.
8. **F. Leymann, D. Roller, M.T. Schmidt**, Web Services and Business Process Management, IBM System Journal, Volume 41, Number 2, 2002.
9. **O. Nicolae, M.Cosulschi, A. Giurca, G. Wagner**, Towards a BPMN Semantics using UML models, CBP 2008 Workshop, Milano, Italy, 2008.
10. **J. Mendling, G. Neumann, M. Nüttgens**, A Comparison of XML Interchange Formats for Business Process Modelling, EMISA 2004.
11. ***, (WS-CDL 1.0) Web Services Choreography Description Language (W3C), http://www.w3.org/TR/ws-cdl-10/, 2005.