

# Towards a BPMN Semantics using UML models

Oana Nicolae<sup>1</sup> Mirel Cosulschi<sup>2</sup> Adrian Giurca<sup>1</sup> and Gerd Wagner<sup>1</sup>

<sup>1</sup> Department of Internet Technology

Institute of Informatics

Brandenburg Technical University at Cottbus, Germany

{nicolae, giurca, G.Wagner}@tu-cottbus.de

<sup>2</sup> Department of Computer Science, University at Craiova, Romania

mirelc@central.ucv.ro

**Abstract.** Standardization initiatives on the field claim to be a consequence of the long expected maturity of Business Process Management (i.e. BPM) market and they try to impose a precise semantics for a unique and common understanding. Standard specifications provide usually less or more informal description of the involved languages. Open-source/proprietary tools often do not succeed to generate an executable code, therefore they could not make a clear distinction between the specification technology they implement and their own requirements. In order to suit these needs and for the sake of a common understanding of the concepts, a high-level modelling of the languages using UML is needed. The aim of this paper is to discuss Service Interaction Patterns directly supported by BPMN 1.1, and to use UML models in order to describe the abstract syntax of the involved concepts. An informal correspondence with WS-BPEL 2.0 elements is provided, together with their associated UML models description. As a consequence of the limited support for Service Interaction Patterns, another purpose of the paper is to discuss the directions in which the BPMN Standard seems to evolve, in order to suit the industry demands.

## 1 Introduction

In the context of distributed computing architecture, Web Services provide a loosely-coupled integration of business processes, while composition languages are dealing with their modelling and enactment. Therefore, languages such as DSLs (i.e. WS-BPEL [4]) or high-level modelling ones (i.e. BPMN [5]), play an important role in systems development area by trying to suit development challenges (complexity, diversity and change).

The subject of language integration is actually a prime topic in the field of business processes collaboration. The most encountered type of language integration is the language transformation that semantically splits its meaning into Model Driven Architecture<sup>3</sup> (MDA) which deals with transformations from PIMs to PSMs and Language Driven Development (i.e. LDD) that comprises

---

<sup>3</sup> MDA - <http://www.omg.org/mda/>

transformations from high-level modelling languages to DSLs (e.g. BPMN to WS-BPEL [19], [15]).

The tightly relation between models and execution languages brought with it the usage of term PAIS (i.e. Process-Aware Information Systems) [12] which designates the concept of execution based on a business process model. Usually, we understand the modelling and execution of business processes as completing technologies and not competing ones. This symbiotic relation, also shows the weaknesses of the languages, the fact that the industry do not use nor accept them individually but in this relation, are taking advantage from their both forms of concrete syntax: graphical/visual and textual.

The Object Management Group (i.e. OMG<sup>4</sup>) standardized and released BPMN (now at version 1.1) and also succeeded to impose it as the de-facto standard in the area of Business Process Modelling. BPMN proves to be an evolving language (i.e. two Specifications for BPMN 2.0 are competing for approval [6] [7]), and OMG consortium presents BPMN as a powerful tool for documenting business processes insisting on control and message flows. The current Specification still lacks precision (no metamodel defined, only an UML static diagram) in specifying a particular language for the conditional expressions, in the modelling of data flow and in the interpretation of specific structures such multiple choice (i.e. inclusive split gateway). Also the mapping to WS-BPEL proves to be difficult and needs further refinement for the enactment phase. This is one of the difficulties a mapping between a graphical concrete syntax and a textual concrete syntax is usually dealing with. A graphical concrete syntax is intuitive, but there is a certain limit of details which can be expressed, and over this limit the graphical syntax proves to be unable to face their complexity.

WS-BPEL language also lacks the formalization based on a well-defined UML metamodel. Being an XML-based language, an XML Schema is provided instead, as a form of abstract syntax.

In the context of Model Driven Architecture (e.g. OMG's MDA) MOF represents the metamodeling language. As ([8] - page 20) sustains, the actual literature about metamodeling induces a confusion about its real meaning. Standards such as UML<sup>5</sup> and OMG's MetaObject Facility<sup>6</sup> (MOF) 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. We also use UML models to describe the abstract syntax for BPMN 1.1 and WS-BPEL 2.0 specifications, and to further provide a common understanding of their specific concepts which involves business process interactions.

The remainder of this paper is organized as follows: Section 2 provides a review on the Service Interaction Patterns directly supported by actual BPMN standard, makes an analogy with corresponding WS-BPEL 2.0 elements and

---

<sup>4</sup> OMG - <http://www.omg.org>

<sup>5</sup> UML - <http://www.uml.org/>

<sup>6</sup> MOF - <http://www.omg.org/mof/>

describes both of them using UML models. Section 3 discusses the approaches of other research works and the directions towards BPMN standard seems to evolve. The last section, discusses conclusions and future works.

## 2 BPMN 1.1. - Service Interaction Patterns

The introduction of (Workflow/Interaction Service) Patterns in BPM area was done in order to capture the BPM languages suitability and their levels of expressiveness pertinent to business processes modelling, management and B2B interactions, respectively. As in [16] we understand a pattern as *the abstraction from a concrete form which keeps recurring in specific, non-arbitrary contexts.*

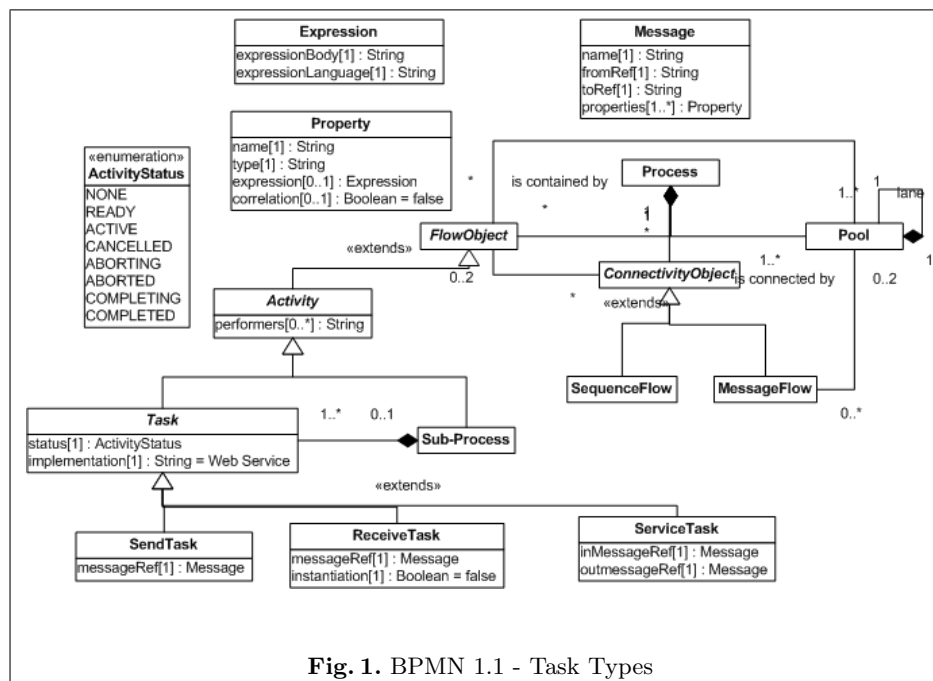


Fig. 1. BPMN 1.1 - Task Types

First introduced by Barros et al. in [2] and [3], the Service Interaction Patterns are used to describe collaboration scenarios which involve two or more parties. Literature evaluates business processes languages regarding their direct/partial or lack of support for these patterns in order to discover their suitability in this context.

We rely on these patterns as they capture the Web Service functionality concerning the composition layer (i.e. orchestration/choreography) inside a business process and they focus on more realistically, multi-participants aspects of the business. We follow the division of interaction patterns made by Barros et al. in

[3], which comprises explanations of these patterns that rely on WS-BPEL code examples, because WS-BPEL can be considered an extension of an imperative programming language with constructs specific to Web Service implementations. The specific WS-BPEL 2.0 elements corresponding to the analysed patterns are visualized using UML class diagrams and an informal mapping from BPMN 1.1 concepts to WS-BPEL 2.0 elements is provided.

BPMN 1.1 uses Business Process Diagrams (i.e. BPDs) to encapsulate its specific constructs: **Pools** (compartmented, if the logic imposes, in many **lanes**) which map the partners involved in a business process. A Pool comprises, at its turn, other 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, which can have the following values: send, receive, service, user, reference, script and manual.

In the context of Service Interaction Patterns, we focus on the concepts of: **swimlanes** (Pools and Lanes - that represents participants at the interaction and simulates their behavior), **message flows** that distinguish from **control flow**, and corresponding tasks which involve the message exchanges: **send**, **receive** and **service** task types.

Both BPMN and WS-BPEL Specifications do not provide much detail on the semantics of data flow (i.e it is unclear what it means if different activities write on the same data object).

When discussing about BPMN Service Interaction Patterns, we expose only those details pertinent to the communication-related aspects (i.e. reliability), as BPMN is a non-executable language. Therefore, all patterns related to control-related aspects (i.e. which involve a runtime determination) can not be directly supported by BPMN.

## 2.1 Single-transmission Bilateral Interaction Patterns

This group of interactions expresses the simplest form of sending/replying/invoking a message between the participants of a business process. The message can be: one-way (i.e. we do not expect an answer) or round-trip (i.e. we expect an answer from the receiver of the message).

**Send Pattern** BPMN captures this pattern using an atomic activity (i.e. Task) which has the value of its **TaskType** attribute set to **Send** (see BPMN Specification terminology page 66). Moreover, this type of Task is identified by **SendTask** name and its mandatory attribute **MessageRef** of type **Message**, which denotes the message sent to an external partner of the business process. Once the message has been sent, the Task is completed.

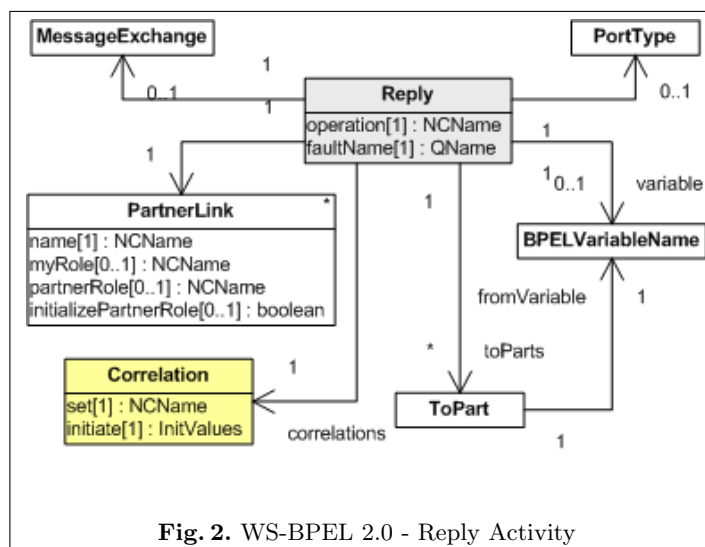
We consider that modeling the behavioral character of BPMN task types using the **TaskType** attribute is not appropriate as it restricts the extensibility of the model. Therefore, our UML class diagram models the **TaskType** attribute as classes specialisations of the main, abstract type **Task** class, which, at its turn is a specialization of the abstract **Activity** class.

By default, the value of `implementation` property of `Task` class describes a Web Service (see Figure 1).

Unpredictable events may affect the usual flow/sequence of activities inside a business process. For example, in response to a sent message may result a fault message. In BPMN 1.1, this situation is represented by attaching a catching intermediate event to the boundary of the task (e.g. of Boundary Events Only - Error, Cancel, Compensation).

WS-BPEL uses the `Reply` activity (which is the analogous concept of an asynchronous `Invoke` WS-BPEL activity) to encode this pattern. A `Reply` activity always supposes the existence of a previously declared `Receive` activity which receives the message request with the same values for the properties `portType` and `operation`. An activity is performed by a participant to the process i.e. a `partnerLink` which implicitly tight the activity with a performer and indirectly an end-point at runtime (see Figure 2).

The `implementation` property of the `Task` has the default value `WebService` and this value implies the mapping of its `Participant`, `Interface` and `Operation` properties to WS-BPEL corresponding elements: `partnerLink`, `portType` and `operation`, respectively.



A BPMN `SendTask` with an `ErrorEvent` attached to its boundary will be mapped into a WS-BPEL `invoke` or `reply` activity that results in a defined operation fault message that is sent to the initiator of the communication.

**Receive Pattern** BPMN captures this pattern using a `Task` with its `taskType` value `Receive`, which means that the task is waiting for receiving a message. Once the message has been received, the `Task` is completed. An incoming flow,

bearing the specified `MessageRef` of type `Message` is waited, for a single instance of the process. The `ReceiveTask` is again a specialization class in our model and it can employ an *instantiation mechanism* for the Process using the mandatory `instantiate` property which has the default value `false`. This attribute may be set to `true` value if the Task is the first activity initiated after the Start Event or a starting Task if there is no Start Event in a business process.

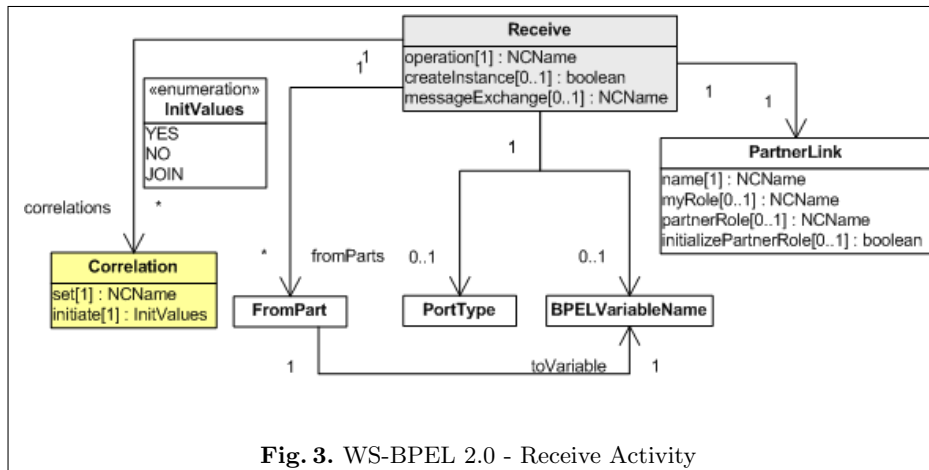


Fig. 3. WS-BPEL 2.0 - Receive Activity

**Send/Receive** This pattern is captured by BPMN using the `Service` value for the `typeTask` property. A BPMN `Service` type of task is used to embed the functionality of a Web Service. Its functionality can be easily mapped to a WS-BPEL synchronous `invoke` element. Both, an incoming message (i.e. `inMessageRef`) and an outgoing message (i.e. `outMessageRef`) need to be specified.

## 2.2 Single-transmission multilateral interaction patterns

These types of interactions comprise messages that are sent by a part to its partners (which can be more than two) and messages received by a party from more than two partners involved in the business process.

**Racing Incoming Messages** This pattern implies the existence of multiple interactions (i.e. send/receive messages) with the same party. It is directly supported by BPMN using the `Exclusive Event-based Gateway`. The term **gateway** expresses a mechanism for controlling the flow of activities inside a business process. It is not an activity. An `Exclusive Event-based Gateway` enables a participant to receive, at a point in the activity flow, different, concurrent types of messages from its business partners. The receipt of messages can be modeled





**Multi-responses** BPMN supports this type of pattern using the loop type of task, so it can create a *repeat until* procedure that stops its mechanism when a trigger event happens. As usual, this structure implies a boolean condition.

A loop activity is modeled also as a specialization class of **Task** class i.e. **LoopActivity**. The **loopCondition** property captures the boolean condition (i.e. **Expression** type) that must apply to the loop task. The **loopCounter** property is used only at runtime to count the number of loops and is automatically updated by the process engine. The **loopCounter** property is incremented at the start of a loop activity and the boolean condition can depend on it.

In BPMN, activities have a start and an end, that are in fact events (i.e. BPMN is taking into consideration only event types that change the flow of activities or involve time triggers). Therefore, the conditions on loop activities can be evaluated at the start or at the end of the activity i.e. **testTime** property of **LoopTime** enumeration.

This type of pattern is captured in WS-BPEL using again the **pick** activity and its **onMessage** handler for each type of message. Anyway, more than one message can be accepted, so a **while** or a **repeatUntil** activity must enclose the **pick** activity, and must specify appropriate boolean condition.

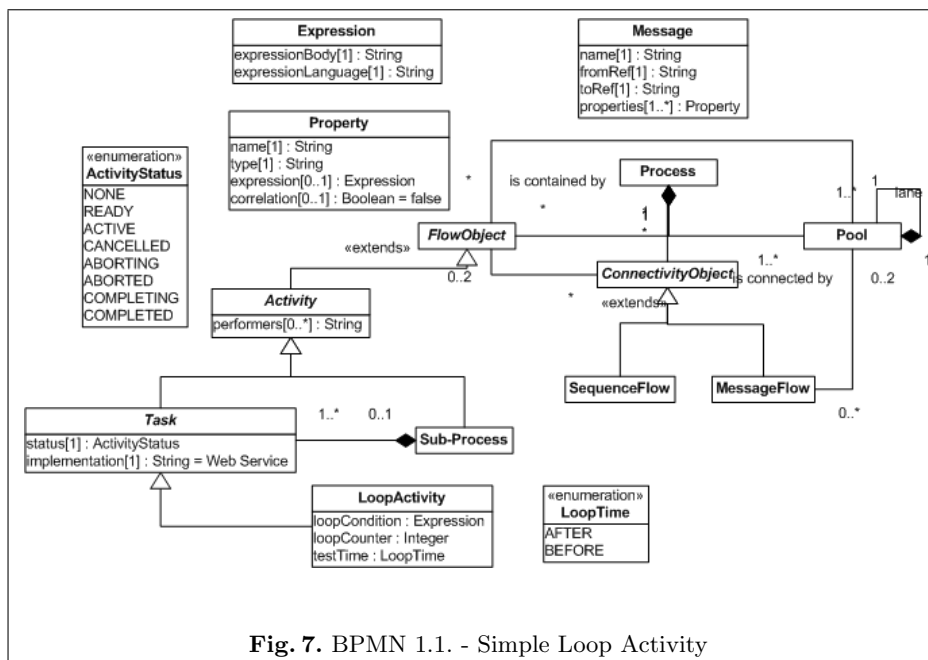


Fig. 7. BPMN 1.1. - Simple Loop Activity

### 3 Discussions

BPMN 1.1 Standard proves to be well accepted by industry<sup>7</sup> as it hides its complexity behind a simple and intuitive visual language, enhanced with an extensibility methodology. The reversed aspect is that BPMN Specification is considered only a documentation for business processes as it lacks a well structured semantics for enactment and also an XML-interchange language for its models.

For the purpose of designing new languages in a common understanding manner, and therefore as an answer to the diversity challenge, T. Clark et al. argue in [8] on the suitability of a language engineering approach based on metamodelling (i.e. the process of developing platform agnostic models). A metamodel for a language is also a model, but more than that, it is capable to capture the language abstract syntax, concrete syntax and its semantics. In this context, Standard Proposals (for version 2.0) intend to accomplish these requirements by providing a metamodel for BPMN (see [6] [7]).

#### 3.1 Related Works on the field

Conceptual modelling languages, such as BPMN, can be evaluated analytically and empirically, as they are two methods that complement each other. Another distinction can be made between assessments of single languages and comparative analysis of several languages. A lot of research works are already focusing on the area of pattern-based analysis of business process modelling (see [22], [23]). The *Workflow Patterns* were introduced in the literature by van der Aalst et al. in [1].

The lack of a well defined BPMN semantic generated attempts to define a formal semantics for a subset of BPMN concepts using Petri nets, but the resulting semantics does not properly model multiple instances, exception handling and message flows. Another attempt to provide a semantics for BPMN was done by Wong et al. in [21], where the authors provide an evaluation according to the semiotic framework (i.e. assessment of quality on a relatively general level) and the BWW ontology (i.e. oriented on finding ontological discrepancies).

A significant amount of work has been done towards the mapping from BPMN to WS-BPEL (see [5], [19], [15]). Due to complicated mappings between a graph-oriented notation language and a block-structured one enhanced with a formal semantics, the translations did not succeed to accomplish the following requirements: completeness, automation and code readability.

Besides BPMN, other languages and standards including WS-CDL, WS-BPEL, UML AD (see [9], [24], [25]) were objects of assessment, with the purpose to test their ability to capture workflow/interaction pattern structures.

---

<sup>7</sup> BPMN Industry Supporters - <http://www.bpmn.org/BPMN-Supporters.htm>

### 3.2 Towards an essential BPMN

Despite the fact that BPMN succeeded to impose itself as a de-facto standard on BPM field, other languages as **Let's Dance**<sup>8</sup> (i.e. choreography field) or **YAWL**<sup>9</sup> (i.e. orchestration area) try to improve and complete BPMN abilities, especially on Service Interaction Patterns assessments, where Let's Dance provides a directly support for most interaction patterns that lack to actual BPMN Specification (i.e. Routing Patterns, Request with Referral, and all Multi-Transmission Interaction Patterns). Let's Dance also comes with its own graphical notation.

On the literature, there are approaches that have mainly focused on the compatibility problem of BPMN for modeling Web Services choreographies described in XML-based languages such as **WS-BPEL** (i.e. abstract processes) or **WS-CDL** (see [15], [9]).

Again, being an evolving language, the need for further refinements (i.e. to capture business process collaborations) has conduced to the appearance of new ramifications from the initial BPMN standard. BPMN 1.1 provides support for choreography modelling by means of interconnected interface behaviour models, which, as Decker et al. described in [11] are susceptible of redundancies (i.e. deadlock situations) and of incompatible behaviour. As an alternative, the authors introduce interaction modeling using BPMN i.e. a set of extensions from the main standard - **iBPMN**. The main improvements brought by Decker et al. concerning process choreography were first described in [10] i.e. concepts of **Participant Sets**, **References** and **Reference Passing**. The suitability of **iBPMN** for choreography modeling was further tested in [11].

M. Siadaty et al. propose a different extension to standard BPMN in the context of modelling business processes collaborations (orchestration/choreography) using business rules, namely **rBPMN**<sup>10</sup>. The purpose of this work is to obtain an executable business process by encode it in a specific rule-based language (i.e. **R2ML**<sup>11</sup> [18]) and further to be capable to deploy it in a particular business rule-engine (i.e. **Drools**<sup>12</sup>). **R2ML** supports the event-drive situations by means of Reaction Rules.

As literature proposes new extensions for BPMN, so that the analysts can better understand the models and the process collaborations by means of choreographies, it also investigates the necessity of such a complex and multiple control flow situations, where some of BPMN concepts are overlapping in their provided functionality (see [22], [23]). In this context, the need for a simplified language was sustained by [13] that tried to provide a simplified visual notation for BPMN. Another aspect of the BPMN workflow concepts simplification can be provided through the use of just a core part of the entire BPMN set of concepts. This aim is not an easy one, as it must rely on multiple and significant Use-Cases,

<sup>8</sup> Lets Dance - <http://sky.fit.qut.edu.au/dumas/LetsDance/LetsDanceEval.pdf>

<sup>9</sup> YAWL - <http://www.yawl-system.com/>

<sup>10</sup> rBPMN - <http://milan.milanovic.org/papers/i2Lor20poster.pdf>

<sup>11</sup> R2ML- <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=R2ML>

<sup>12</sup> Drools - <http://www.jboss.org/drools/documentation.html>

and workflow patterns assessments that should prove the suitability of that core concepts (i.e. **essential** BPMN) for process modelling.

## 4 Conclusions and Future Works

Implementation and ongoing management of inter and cross-organizational business processes are an actual theme on the business market, and existing literature is almost exhaustive in this direction. We use UML models to describe the abstract syntax of BPMN 1.1 and WS-BPEL 2.0 concepts involved in the assessment of Service Interaction Patterns directly supported by BPMN 1.1 together with an analogy with corresponding elements from WS-BPEL 2.0 language. The abstract syntax we use provides only a structure of concepts and terms in a language without given any consideration to their presentation or meaning. Though, the real aim of a language is to provide a specific semantics i.e. the proper form of what the language represents and means. Our future work focuses on developing a metamodel for an essential BPMN. The semantics of the essential BPMN is to be derived from BPDM<sup>13</sup> metamodel (i.e. metamodel for BPMN language).

## References

1. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. *Workflow Patterns. Distributed and Parallel Databases*, 14(1), pages 5-51, 2003.
2. A. Barros, M. Dumas and A. H. M. ter Hofstede, *Service Interaction Patterns* in BPM 2005, pages 302-318.
3. A. Barros, M. Dumas, and A. H. M. ter Hofstede, *Service Interaction Patterns: Towards a Reference Framework for Service-based Business Process Interconnection* Technical Report FIT-TR-2005-02, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia, March 2005.
4. (OASIS) *Business Process Execution Language (BPEL 2.0)*, 2006, <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>
5. (OMG) *Business Process Modeling Notation 1.1*, (BPMN 1.1), 2007, <http://www.omg.org/cgi-bin/doc?dtc/2007-06-03>
6. (BPDM Team) Adaptive, Axway Software, EDS, Lombardi Software, MEGA International, Troux Technologies, Unisys, *BPMN 2.0 Specification Proposal*, 2008, <http://www.omg.org/cgi-bin/doc?bmi/08-02-03>
7. (BEA, IBM, Oracle, SAP), *BPMN 2.0 Specification Proposal*, 2008, <http://www.omg.org/cgi-bin/doc?bmi/08-02-06>
8. T. Clark, P. Sammut, J. Willans, *Applied Metamodeling. A Foundation for language driven development. (Second Edition)*, Ceteva 2008.
9. G. Decker, H. Overdick, J. M. Zaha, *On the Suitability of WS-CDL for Choreography Modeling (Extended Version)*, In EMISA 2006, volume 95, LNI, pages 21-33.
10. G. Decker and A. Barros. *Interaction Modeling using BPMN*. Proceedings of the 1st International Workshop on Collaborative Business Processes (CBP), pages 208-219, 2007.
11. G. Decker and F. Puhmann *Extending BPMN for Modeling Complex Choreographies* Proceedings of the 15th International Conference on Cooperative Information Systems (CoopIS), LNCS 4803, pages 24-40, Vilamoura, Portugal, November 2007. Springer Verlag.
12. M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede *Process-aware information systems: bridging people and software through process technology*, John Wiley and Sons, 2005, ISBN:0-47166-360-9.
13. M. Havey, *Keeping BPM Simple for Business Users: Power Users Beware*, BPTrends January 2006.
14. H. Meyer, D. Kuroпка *Requirements for Automated Service Composition*, In Business Process Management Workshops, LNCS, Vol. 4103, pages 439-450, 2006.

<sup>13</sup> BPDM - <http://www.omg.org/docs/dtc/07-07-01.pdf>

15. C. Ouyang, Dumas, A. H. M. ter Hofstede and W. M. P. van der Aalst. (2007) *Pattern-based translation of BPMN process models to BPEL web services*, International Journal of Web Services Research (JWSR).
16. D. Riehle and H. Zlighoven, *Understanding and Using Patterns in Software Development*, Theory and Practice of Object Systems (TAPOS), Vol. 2, Nr.1, 1996, pages 3-13.
17. T. Wahl, G. Sindre *An Analytical Evaluation of BPMN Using a Semiotic Quality Framework*, In EMMSAD'05, 2005.
18. G. Wagner, A. Giurca and S. Lukichev, *R2ML: A General Approach for Marking up Rules*, Dagstuhl Seminar Proceedings 05371, In F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, ISSN:1862-4405.
19. S. A. White. *Using BPMN to Model a BPEL Process*, BPTrends, March 2005.
20. S. A. White, *Process Modelling notations and Workflow Patterns*, BPTrends, March 2004.
21. P. Y. H. Wong and J. Gibbons, *A Process Semantics for BPMN*, Submitted for publication, 2008, <http://www.comlab.ox.ac.uk/peter.wong/pub/bpmsem.pdf>
22. P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. *Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives (revised version)*, BPM Center Report BPM-06-17, BPMcenter.org, 2006.
23. P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell *On the Suitability of BPMN for Business Process Modelling*, In BPM 2006, pages 161-176.
24. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. *Analysis of Web Services Composition Languages: The Case of BPEL4WS*. In ER 2003, pages 200-215.
25. P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell, *Pattern-based Analysis of the Control-Flow Perspective of UML Activity Diagrams* In ER 2005, pages 63-78.