Towards a Methodology for Semantic Business Process Modeling and Configuration

Ingo Weber[1], Jörg Hoffmann[2], Jan Mendling[3], and Jörg Nitzsche[4]

[1] SAP Research Karlsruhe, Germany
ingo.weber@sap.com
[2] University of Innsbruck, DERI, Austria
joerg.hoffmann@deri.at
[3] BPM Cluster, Queensland University of Technology, Australia
j.mendling@qut.edu.au
[4] Institute of Architecture of Application Systems, University of Stuttgart, Germany
joerg.nitzsche@iaas.uni-stuttgart.de

Abstract. This paper discusses potential benefits from adding semantics to Business Process Management from a methodological point of view, with a focus on the Modeling and Configuration phases. For this purpose, in each of these phases the established activities are examined and new activities are suggested: Firstly, we suggest combining existing control flow validation techniques with semantic process validation techniques. Second, discovery and composition techniques can be used to find implementations, e.g. services (or combinations of services), for the implementation of process activities at modeling time. The discovered implementations allow for mapping the process steps to the IT infrastructure according to several strategies during process configuration, which helps clearly separating modeling from configuration concerns. Furthermore, a new way of testing executable process models is suggested. [1]

1 Introduction

Business process models are used in a variety of scenarios, among others, documentation of business requirement and specification of implementation aspects. In order to serve the latter purpose, which is in the focus of this paper, an executable business process model has to specify not only the control flow, but also data flow, interfaces to the services that implement the activities, and run-time binding to service implementations. For each of these specification steps there are different quality and correctness considerations to be taken into account such that the resulting implementation meets the overall goals of the stakeholders.

In traditional Business Process Management (BPM) many of these specification steps cannot be handled automatically or facilitated by appropriate tools,

[1] This work has in part been funded through the European Union's 6th Framework Programme, within Information Society Technologies (IST) priority under the SUPER project (FP6- 026850, http://www.ip-super.org).

E. Di Nitto and M. Ripeanu (Eds.): ICSOC 2007 Workshops, LNCS 4907, pp. 176–187, 2009.

since the meaning of process artifacts is not captured in the process model. Furthermore, design artifacts such as existing process models or existing service interfaces are hardly reused in practice. The Semantic BPM (SBPM) approach[2] addresses these challenges by annotating semantic descriptions to process artifacts, yielding the opportunity to ease a business analysts job by providing tools that facilitate various parts of the BPM lifecycle.

In this paper we describe a methodology, i.e. "a documented approach for performing activities in a coherent, consistent, accountable, and repeatable manner" [2], which structures the usage of the new semantics-enabled techniques that can be used when creating an executable process model. The aim is not to give an complete discussion containing all aspects that may ever arise in relation to a methodology for SPBM, but to provide methodological guidance with respect to a number of the new opportunities. In particular, after an overview of our approach in Section 2, Section 3 discusses the modeling phase and four individual activities that we identified, three of which are important from correctness considerations. Then, Section 4 details the configuration phase and shows how four more activities should be carried out to arrive at an executable process specification. Finally, Section 5 discusses the contributions and points to future work.

2 Guidance for the Semantic BPM Life Cycle

Several textbooks on BPM such as [3,4] discuss the business process management life cycle on a very abstract level such that it provides only limited methodological guidance for the implementation of executable process models. Other approaches such as [5,6] focus on partial aspects of BPM design and do not cover implementation details. This section takes the semantic BPM life cycle as a starting point and describes our proposal of a methodology for semantic business process modeling and configuration. The traditional BPM life cycle has shortly been introduced in [4], and in [7] it has been extended into the Semantic Business Process life cycle (Fig. 1). While the life cycle of [7] is followed here on a high level, the content of the modeling and configuration phases as described in this paper is quite different. To mention just one difference: composition per process task is described in [7] as an activity of the configuration or execution phase, while we argue below that this activity should be part of modeling.

The overall semantic BPM life cycle depicted in Fig. 1 identifies four principal phases: semantic business process modeling, configuration, execution, and analysis - typically performed in this order. In short, the *modeling phase* comprises the creation or adaptation of a business process model. The *configuration phase* deals with refining this model into an executable form, traditionally by implementing it. The *execution phase* is concerned with how process instances are executed in the organization, often through its IT systems. Logs and other perceptions

[2] Cf. SUPER IP(EU-funded Integrated Project, http://www.ip-super.org), Semantic Business Process Management Working Group (http://www.sbpm.org), SBPM workshop [1]

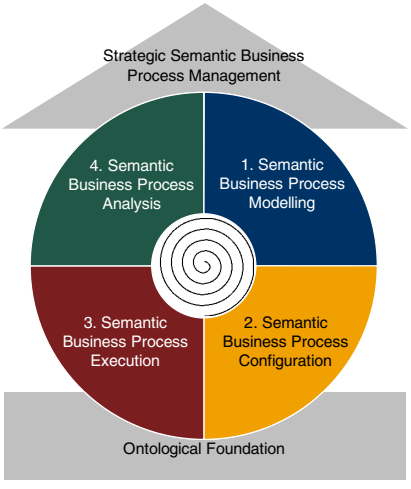

Fig. 1. Semantic BPM life cycle, adapted from [7]

from the execution phase serve as input to the *analysis phase*, where the execution behavior of processes is inspected and recommendations regarding changes and optimization are made. The strategic and the foundational layer define the context for these phases. The *strategy* of an organization defines its long-term business goals, and serves as the basis for operational decisions. The foundational layer in the SBPM life cycle consists of *ontologies* which capture generic knowledge about processes, services, etc., along with domain and organization-specific knowledge. These ontologies are referenced in the specific business processes.

In this paper we focus on the modeling and configuration phases of the SBPM life cycle. We hereby assume that a conceptual business process model needs to be refined into an executable model, and deployed to an IT-based execution environment. Fig. 2 gives an overview of the activities we identify. The goal of the modeling phase is to create or adapt a process model that can be executed correctly. Therefore, beyond conceptual modeling (M.1) and discovery and composition of tasks (M.3), it is important to consider control flow verification (M.2) and semantic validation (M.4) to guarantee this level of correctness. In the configuration phase, the process model is translated to an executable language (C.1), yielding an executable model. After the service binding (C.2) the executable process is tested (C.3) and finally deployed (C.4). The process designer conducts each of these four plus four activities and is supported by respective tools. In many cases there will be two process designers with partially overlapping expertise: a business analyst being responsible for the activities of the modeling phase, and a software engineer in charge of the configuration phase.

Note that the activities related to quality assurance (M.2, M.4, C.3) are optional, and the arrows show one possible recommended order of their execution. However, if a process model is created for execution in an industry setting, model quality and correct execution are of crucial importance. Accordingly, it is

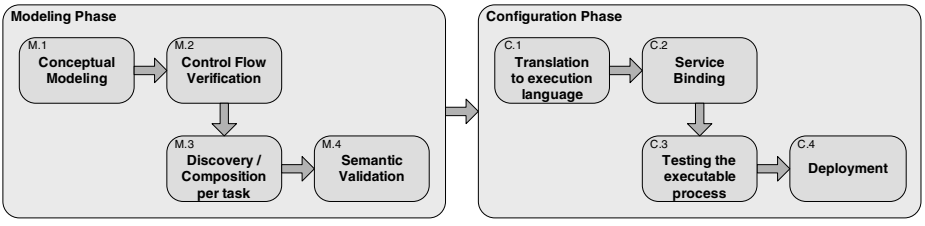

Fig. 2. Detailed view of the Modeling and Configuration phases in the BPM life cycle. The arrows depict a suggested usage of the various activities in the absence of problems.

strongly recommended to consider control flow verification (M.2), semantic validation (M.4), and testing (C.3) before deploying an executable process model. In the subsequent sections, we discuss each modeling and configuration activity in detail. To illustrate feasibility we describe several instantiations of the respective concepts, in particular with a focus on findings of the SUPER IP^2.

Since we aim at a generic methodology, a number of the suggested activities must be tailored to a specific notation or language in order to provide a thorough solution. For instance, the control flow verification depends on the execution semantics of the process modeling language at hand. Each of the activities in our methodology is subject to ongoing work on the basis of particular languages.

3 The Modeling Phase

In this section we describe the added value and functionality of the SBPM approach during modeling of a business process. The goal of this phase is to create a conceptual process model which depicts the ideal structure (to-be process) from a business analyst's point of view and at the same time is meaningful input to the creation of the executable process.

3.1 Conceptual Modeling (M.1)

Business Process Modeling can serve various purposes ranging from documentation to reorganization and workflow automation. Before starting the actual modeling task the process designer has to carefully decide for a suitable modeling language, a respective tool to facilitate the modeling, required perspectives on the process, and modeling conventions [8]. The choice for a *modeling language* can hardly be made without considering potential *tool support* since industry-scale modeling projects require sophisticated management features such as multi-user support or persistent storage in a database. Several languages are used for modeling business processes including EPCs, Petri nets, YAWL, UML Activity Diagrams, and BPMN – see e.g., [4] for an overview. *Modeling conventions* standardize the way processes are modeled in order to provide comparability and consistency across models. The *required perspectives* on the process depend on

the purpose of modeling. Since it is our goal to finally arrive at executable processes we would need not only to model the control flow, but also the data flow between the activities. Furthermore, in the case of semantic BPM the user has to annotate references to ontologies to the model. In [9] the authors present an approach towards user-friendly annotation of semantic information. SBPM offers further modeling support, e.g., content-aware process model auto-completion as in [10], which also facilitates re-use.

3.2 Control Flow Verification (M.2)

It is an important design goal for a business process model that no matter which decisions are taken at run-time the process will always complete properly without deadlocking. The *soundness* property originally defined for Workflow nets, i.e. a Petri net with one source and one sink, captures this requirement and demands that (i) for every state reachable from the source, there exists a firing sequence to the sink (option to complete); (ii) the state with a token in the sink is the only state reachable from the initial state with at least one token in it (proper completion); and (iii) there are no dead transitions [11]. The soundness property can be verified with Petri net analysis tools such as Woflan [12].

Petri-net analysis techniques are not directly applicable for other languages, but need to be adapted to the routing elements these languages provide. The OR-join that is included in EPCs, YAWL, and BPMN has been a challenge from a verification perspective for quite a while. One solution in this context is to use the notion of relaxed soundness instead of soundness. A process is *relaxed sound* if every transition in a Petri net representation of the process model is included in at least one proper execution sequence [13]. A problem though is that this relaxed soundness notion does not guarantee that the process always completes properly. As a consequence, dedicated verification techniques have recently been defined for the verification of EPCs including OR-joins [14] and for YAWL nets including cancellation regions [15]. Since these languages cover most of the control flow features of other languages, these verification techniques can easily be adapted for UML Activity Diagrams or BPMN. This way the process designer can make sure that a process model will not deadlock at run-time.

3.3 Discovery/Composition Per Task (M.3)

One of the obstacles when creating an executable process from a to-be process is finding out if the tasks in the process model may be implemented in the specified way. For a process task which should be performed by an IT system there may be an existing service, there may be services at a different level of granularity, or there may be no implementation at all. In the activity of finding suitable existing services, the process designer is supported by an intelligent software tool for automatic discovery and composition, which in term makes use reasoning techniques over domain ontologies. We will focus on semantic Web services in the following.

In discovery the tool searches, for each task, in the service infrastructure to check whether an existing service can implement that task. This check involves *matching* the services against the requirements of the task. To be able to do so, both the services and the task are annotated with a semantic description, e.g., in the WSMO language, of the provided/requested functionality. If no single service can implement the task, then the tool tries to find a composition of services that matches the task based on the semantic annotations. A key difference to other approaches here is that the tool only supports the assignment of services to tasks at design time, and that discovery and composition are performed *per task*. More ambitious approaches, like e.g. [16], try to compose the entire process. While such a technology would be nice to have, composing entire processes is essentially a form of automatic programming, which is a notoriously hard problem. Apart from the obvious severe challenges regarding the computational complexity of such a general problem, there is the practical difficulty of formulating the "goal" for such a composition. Such a goal must be a specification of what the process must fulfill in order to be suitable. Providing such a specification is often more difficult for the human that just formulating the process itself.[3] In that sense, our approach trades some generality for pragmatics: for a single task, specifying a goal is often quite easy, and it is tedious to go look for appropriate services.

The goal of a task may be refined with the semantic annotations of the found services, or in case a composition is necessary, the original task may be replaced with a container holding the tasks that relate to the services which were composed together, i.e., the tasks in the newly created container are annotated with semantic goals that correspond to the composed services and are ordered as specified by the composition. This way, the granularity of the process tasks can be adapted to the granularity of the available services, which is usually a challenge in process modeling.

Importantly, the discovered/composed "services" are identified with their semantic descriptions. Having such a semantic description in a task means that we can, later on in the Configuration Phase, replace this description with any service that satisfies it. In other words, we can separate between discovery of services, and *binding* of services – links to the found services may be kept, but during modeling no particular implementation of a service is chosen. This is advantageous because in this way we do not force a premature decision on what the actual implementation technology will be: the semantic task description can be mapped onto all sorts of technologies whichever is available and/or suits best the requirements of the particular deployment scenario.

If discovery/composition reports that there is no implementation, then according action must be taken by the process designer, e.g., process model might be changed, or the particular task may be marked, such that it is clear that an implementation for it must be created during the configuration phase. In contrast, if discovery/composition succeeds, the results should be displayed to the

[3] The unconvinced reader is invited to think about the last computer program she wrote, and then think about how she would have explained to the computer what that program is supposed to do.

process designer for checking, if only for safety reasons (the semantic descriptions might be flawed, or not precise enough to obtain a fully implemented process).

In the sense outlined here, discovery/composition can be seen as a further quality assurance activity, in that it can provide feedback if a process model is implementable or not – even to a modeler who is not trained in IT or the system landscape: if discovery/composition succeeds, the modeler knows that there is at least one service for each task to implement it.

3.4 Semantic Validation (M.4)

After the Discovery/Composition phase each task is associated with one or more semantic representations of services. If the Discovery/Composition is not done on the process level, conflicts may arise between *different* tasks. As an example consider a computer hardware procurement process for replacing the old equipment of a manager. This process includes the tasks *Buy Computer* and *Buy Printer*. Since the company pursues a double-sourcing strategy in procurement, computers and printer have to be bought from different vendors. This constraint cannot be checked unless one examines the behavior of different tasks across the process. Other issues beyond control flow could be that two services that may be executed in parallel and that have conflicting effects.

Control Flow Verification (cf. M.2, section 3.2) ignores such issues, and takes care only of checking whether the process model as such is sound. In Semantic Business Process Management, our aim instead is to have a formal model of what happens in the business, and to validate the processes against that model. In such a setting, checking constraints on the combined behavior of different tasks is essentially a model checking problem. This leads to the possible application of techniques such as linear temporal logic (LTL) [17] or access control policies for processes [18].

The basic novelty in model checking for Semantic Validation as done in (M.4) lies in the use of semantic techniques, i.e., of ontologies. Hence, to be applicable, traditional model checking techniques must be adapted in this regard. Further, in business process management, different special cases may be of interest than in other model checking scenarios, and hence such special cases – where validation may be doable in polynomial time – need to be investigated.

4 The Configuration Phase

The Configuration phase aims at mapping a semantically described conceptional process model to an executable model that is bound to a particular IT infrastructure. That is, the process model has to be translated from the formalism used by business experts to a formalism that can be executed for instance by an execution engine and the semantic descriptions of tasks have to be mapped to concrete implementations, e.g. services. Given the conceptual model is sufficiently well described and no errors occur, the configuration phase could be performed mostly automated.

We use the conceptual model of WSMO to describe the functionality a (set of) task(s) of a process requires and the functionality services provide. As a process execution language we use BPEL4SWS (BPEL for Semantic Web Services) [19], an extension of BPEL 2.0 [20] that allows using both WSDL (Web Service Description Language) [21] and Semantic Web Service frameworks like WSMO to describe activity implementations. We use particular technologies in describing the configuration phase for reasons of increased tangibility. Nevertheless, most of the concepts can be easily ported to other concrete technology. In this manner, the configuration phase starts with a conceptual process model in (s)BPMM and translates and refines this model into an executable process model in BPEL4SWS.

4.1 Translation to Executable Process (C.1)

The modeling phase leads to a process model that is both sound regarding its control flow and validated against domain constraints using reasoning techniques. In the translation phase this process model is mapped to an executable language such as BPEL. There are several challenges for the transformation exercise since executable languages are often more restrictive in the way they represent control flow [22]. Still, there are some standard solutions to automate the transformation if there are no unstructured loops in the conceptual models [23]. For the transformation of BPMN to BPEL there are several implementations of a transformation, among others the one defined in [24].

4.2 Service Binding (C.2)

During discovery and composition per task (M.3) it is ensured that there is at least one implementation that implements a given task. When binding to the infrastructure the discovery is used to discover the most appropriate implementations out of the available ones. When using the WSMO framework, the semantic task descriptions are transformed to full WSMO goals and subsequently corresponding WSMO Web services are discovered. Given this setting there are several strategies for binding implementations to executable process models during configuration:

1. WSDL services as activity implementations
   WSMO provides a grounding mechanism to WSDL for both, goals and web services. In case the WSMO goal and the WSMO Web service are grounded to WSDL these WSDL interfaces can be used to build a partner link type which can be used within the process model. Note that a grounding in the goal is only required in case of asynchronous communication [25]. Interaction activities representing the tasks of the conceptual process model reference a corresponding partner link and the WSDL operations the WSMO descriptions are grounded to. This configuration strategy results in a conventional BPEL process that runs on a conventional BPEL engine which invokes conventional Web services.

However, as a result, the flexibility of the executable process model is limited because using WSDL for describing activity implementations hampers using functionally equal services that implement different WSDL interfaces. The actual implementation, i.e. endpoint, can be either extracted from the discovered WSMO service and determined during deployment (design time binding) or discovered during runtime (runtime binding).

2. WSMO Goals as activity implementations
Using WSMO goals as activity implementations implies the existence and usage of a WSMO enabled middleware (e.g. WSMX (Web Service Modeling eXecution environment) [26]). Goals can be used with and without a restriction on Services that might be used. The restriction might be a single service (which corresponds to design time binding of WSDL services with respect to flexibility) or a (ranked) list of functionally equal services that were discovered during design time. Even more flexibility is achieved by using a goal without any restrictions on services that might be used. In this case any WSMO Web Service that meets the functional requirements can be discovered and invoked during runtime.

4.3 Testing the Executable Process (C.3)

Creating correctly executing processes is a cumbersome task, due to the fact that the precision of the process then is the same as the precision of a programming language. Practical experience shows that producing correctly executing BPEL even with state of the art tools requires several attempts and hours. Therefore, even though control flow and semantic validity have been checked, it is necessary to test an executable process before deployment.

For this purpose, [27] presents an approach to BPEL testing through "unit tests", which is related to traditional white box tests from software engineering. While this work is focused on BPEL exclusively, the methodology presented in this paper is independent of a specific format for executable processes. While the approach from [27] should be adaptable to other languages and is certainly worthwhile considering, two additional ways for testing an executable process are mentioned below. Both of these testing scenarios require some infrastructure.

The first additional option is inspired by traditional software engineering, where a replication of the productive systems is made available as a test infrastructure. Within a single organization, this approach may be extended to executable processes in the obvious way. For cross-organizational processes, a dummy replication of the remote systems within the own testing infrastructure may be beneficial.

The second option requires an abstract communication layer, e.g. in form of an Enterprise Service Bus (ESB)[28]. This bus can then be extended with a test mode, such that it reacts to messages with a "test" flag with one of a set of predefined test replies, e.g., chosen by random. The test reply messages for a particular service are stored in the ESB when this service is deployed, and may be updated over time. With this mechanism, the test message never reaches the productive application systems, and the tested executable service may only make

use of the services which are available through the ESB. However, there are a few pitfalls to this approach, e.g., when a process relies on instance-related replies the predefined test answers may not be sufficient. Although this approach aims at a lean and elegant testing infrastructure with minimal overhead, the practical applicability remains yet to be shown.

Regardless of the particular approach chosen, as soon as a process is critical and non-trivial to the least extent, testing an executable process before deploying it to productive systems is a must.

4.4 Deployment (C.4)

Deployment is the step of making processes productive and available. Processes are deployed to the infrastructure and registered, e.g. as a service. In case a service binding strategy was chosen that is different from a static binding (in our scenario different from WSDL design time binding and WSMO Goal with restriction that only one concrete service might be used) additional parameters like Quality of Service can be applied to make the (runtime) discovery of services more precise.

5 Discussion

In this paper, we described a structured methodological approach to business process modeling and configuration. This methodology easily facilitates reuse of existing process artifacts, and by doing so, aims to bridge the gap between conceptual modeling of business requirement and information system implementation. Our methodology introduces several activities for quality assurance that benefit from semantic annotations.

Let us briefly discuss our proposed order of activities as per Fig. 2 and the motivations behind it. A process model with sound control flow is not invalidated by discovery and composition, but potentially the model grows through these activities. Therefore, control flow verification is best done before discovery/composition, because it is easier for a modeler to correct a smaller model. Moreover, semantic process validation can only be done on the basis of a sound control flow, and of course semantic validation should happen after discovery/composition. In the configuration phase, the language translation takes place before service binding, because binding mechanisms are language-specific. Testing the executable process is done after service binding, because it of course requires concrete services. Deployment finalizes the configuration phase, and thus is performed last.

The utilization of semantic technologies in this methodology differs from previous approaches in several important aspects. For example, performing composition per task, rather than for an entire process, has a much better chance to yield useful results at a low modeling cost. Semantic validation is an entirely new step. Our arrangement takes the appropriate measures to deal with the subtle interplay between discovery and binding. Composition is best viewed as part of

the modeling phase, since getting to the desired IT-level model may require an iteration of human and computer-supported activities; at least, a final approval by the human user is needed before a composed process is deployed.

In future research, we aim to provide a complete instantiation of our methodology based on sBPMN [29] and BPEL4SWS. This implementation will provide us with practical insight into the applicability of the methodology. In particular, it offers the opportunity for a validation by case studies and an in-depth comparison with current BPM practice.

References

1. Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.): Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007), Innsbruck, Austria (2007)
2. US Dept. of the Treasury, CIO Council: Treasury Enterprise Architecture Framework Version 1.0 (2000) [Retrieved September 4, 2007], http://www.eaframeworks.com/TEAF/teaf.doc
3. Leymann, F., Roller, D.: Production Workflow - Concepts and Techniques. Prentice-Hall, Englewood Cliffs (2000)
4. Dumas, M., ter Hofstede, A., van der Aalst, W.M.P. (eds.): Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley, Chichester (2005)
5. van Hee, K., Sidorova, N., Somers, L., Voorhoeve, M.: Consistency in model integration. Data & Knowledge Engineering 56, 4–22 (2006)
6. Frederiks, P.J.M., van der Weide, T.P.: Information modeling: The process and the required competencies of its participants. Data & Knowledge Engineering 58, 4–20 (2006)
7. Fantini, P., Savoldelli, A., Milanesi, M., Carizzoni, G., Koehler, J., Stein, S., Angeli, R., Hepp, M., Roman, D., Brelage, C., Born, M.: SUPER Deliverable D2.2: Semantic Business Process Life Cycle (August 6, 2007) (2007), http://www.ip-super.org/res/Deliverables/M12/D2.2.pdf
8. Becker, J., Kugeler, M., Rosemann, M. (eds.): Preparation of Process Modeling. Process Management: A Guide for the Design of Business Processes, pp. 41–78. Springer, Heidelberg (2003)
9. Born, M., Dörr, F., Weber, I.: User-friendly Semantic Annotation in Business Process Modeling. In: Hf-SDDM 2007: Workshop on Human-friendly Service Description, Discovery and Matchmaking, at WISE 2007, Nancy, France (to appear, 2007)
10. Markovic, I., Pereira, A.C.: A formal framework for reuse in business process modeling. In: Workshop on Advances in Semantics for Web services (semantics4ws), at BPM 2007, Brisbane, Australia (to appear, 2007)
11. van der Aalst, W.M.P.: Verification of Workflow Nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
12. Verbeek, H.M.W., Basten, T., van der Aalst, W.M.P.: Diagnosing Workflow Processes using Woflan. The Computer Journal 44, 246–279 (2001)
13. Dehnert, J., van der Aalst, W.M.P.: Bridging The Gap Between Business Models And Workflow Specifications. International J. Cooperative Inf. Syst. 13, 289–332 (2004)

14. Mendling, J., van der Aalst, W.M.P.: Formalization and Verification of EPCs with OR-Joins Based on State and Context. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 439–453. Springer, Heidelberg (2007)
15. Wynn, M.T., Edmond, D., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Achieving a General, Formal and Decidable Approach to the OR-Join in Workflow Using Reset Nets. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 423–443. Springer, Heidelberg (2005)
16. Pistore, M., Traverso, P., Bertoli, P.: Automated composition of web services by planning in asynchronous domains. In: 15th International Conference on Automated Planning and Scheduling (ICAPS 2005) (2005)
17. Pnueli, A.: The Temporal Logic of Programs. In: Proceedings of the 18th IEEE Annual Symposium on the Foundations of Computer Science, pp. 46–57. IEEE Computer Society Press, Providence (1977)
18. Bertino, E., Ferrari, E., Atluri, V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security (TISSEC) 2 (1999)
19. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: BPEL for Semantic Web Services (submitted, 2007)
20. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language version 2.0. Committee specification, OASIS (2007)
21. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1 (2001)
22. Recker, J., Mendling, J.: On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. In: Latour, T., Petit, M. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 521–532. Springer, Heidelberg (2006)
23. Mendling, J., Lassen, K., Zdun, U.: On the transformation of control flow between block-oriented and graph-oriented process modeling languages. International Journal of Business Process Integration and Management 2 (2007)
24. Ouyang, C., Dumas, M., Breutel, S., ter Hofstede, A.: Translating standard process models to BPEL. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 417–432. Springer, Heidelberg (2006)
25. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: WSMO/X in the Context of Business Processes: Improvement Recommendations. International Journal of Web Information Systems (2007) ISSN: 1744-0084
26. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX – a semantic service-oriented architecture. In: Proceedings of the International Conference on Web Services (ICWS 2005), Orlando, USA (2005)
27. Li, Z., Sun, W., Du, B.: Bpel4ws unit testing: Framework and implementation. International Journal of Business Process Integration and Management 2 (2007)
28. Chappell, D.A.: Enterprise Service Bus. O'Reilly, Sebastopol (2004)
29. Abramowicz, W., Filipowska, A., Kaczmarek, M., Kaczmarek, T.: Semantically enhanced Business Process Modelling Notation. In: [1]