# Supporting Execution-Level Business Process Modeling with Semantic Technologies

Matthias Born[1], Jörg Hoffmann[1], Tomasz Kaczmarek[3], Marek Kowalkiewicz[1], Ivan Markovic[1], James Scicluna[2], Ingo Weber[1], and Xuan Zhou[1]

[1] SAP Research, Karlsruhe, Germany, {mat.born | joe.hoffmann | marek.kowalkiewicz | ivan.markovic | ingo.weber | xuan.zhou}@sap.com
[2] STI Innsbruck, Austria, james.scicluna@sti2.at
[3] Poznan University of Economics, Poland, t.kaczmarek@kie.ae.poznan.pl

**Abstract.** When creating execution-level process models from conceptual to-be process models, challenges are to find implementations for process activities and to use these implementations correctly. Carrying out these activities manually can be time consuming, since it involves searching in large service repositories and cycles of testing and re-designing. We present *Maestro for BPMN*, a tool that allows to annotate and automatically compose activities within business processes, and to verify the consistency of an annotated process.

## 1 Introduction

One of the big promises of Semantic Business Process Managament is to help bridge the gap between the business level, where the processes are modeled, and the IT level, where they are implemented. This is one of the key issues addressed in the SUPER project,[1] based on Service-Oriented Architectures, and on semantic annotations. For that purpose, we have made extensions to *Maestro for BPMN*, a modeling tool developed at SAP Research.

In our framework, business processes are modeled as a set of *tasks* together with their control flow. In order to make a process executable, a user has to associate every task in the process with an implementation in terms of services. To support this implementation activity, we propose a combination of the following steps: (1) semantically annotate the individual tasks; (2) check if the annotation and the control flow are consistent; (3) discover and compose services which implement the functionality required for each task. In our work, we developed techniques supporting the user in step (1) (published in [2]); we developed fully automated techniques for step (2) (published in [5]) and step (3) (published in [8]). The demonstration will show the Maestro tool, which implements all these techniques in a convenient GUI. Figure 1 shows a screenshot.

Within SUPER, business process models are represented in terms of the *sBPMN* ontology [1],[2] serving as a meta model for BPMN process models. The ontology features the concepts, relations and attributes for standard BPMN. We

[1] http://www.ip-super.org
[2] sBPMN is written in WSML (http://www.wsmo.org/TR/d16/d16.1/v0.21/)

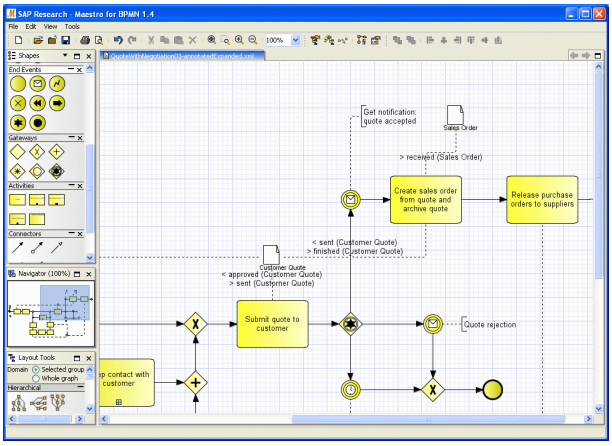


**Fig. 1.** A fragment of a process model represented in Maestro for BPMN.

extended it with the ability to define a state of the process before and after execution of successive tasks. With these extensions we can derive semantic goal descriptions for tasks. This is in line with most popular approaches to Semantic Web Service description,[3] where Web services can be annotated with preconditions and postconditions.

As indicated, we developed support for the convenient creation of semantic annotations. That support is based on business object life cycles. Further, we extended Maestro with automatic consistency checking – verifying the interaction of annotations and control flow – as well as automatic discovery and composition – finding service-based implementations for individual tasks.

We define a formal framework for service composition, following A.I. methodologies [9]. We consider plug-in matches, where services do not have to match exactly, but must be able to connect in all possible situations. In particular, we take the background ontology, i.e. the domain axioms it specifies, into account. This is in contrast to many existing works that assume exact matches (of concept names). We explore restricted classes of axioms, to find solutions efficiently.

The consistency checking also deals with ontological domain axiomatizations, and it also exploits restricted classes of axioms for the sake of computational efficiency. The check determines whether any preconditions may be violated, and whether any parallel tasks are in a logical conflict.

Section 2 explains how we support semantic annotations. Sections 3 and 4 cover consistency checking and discovery/composition. Section 5 concludes.

## 2 Process Modeling and Semantic Annotation

From the graphical point of view, *Maestro for BPMN* implements BPMN 1.1. However, it makes use of the sBPMN ontology, by creating on-the-fly a set of instances for sBPMN classes. If a new BPMN task is created on the drawing

[3] Followed for example in WSMO (http://www.wsmo.org).

pane, an instance of the concept *Task* is created in the in-memory working ontology. This enables supportive reasoning over the working ontology.

The tool allows a user-friendly semantic annotation of process models. The annotations link process tasks to a domain ontology. We focus on how process activities manipulate business objects in terms of their life cycles. E.g., a task "Send quote" sets the status of the object "Quote" to the state "sent". For this purpose, the domain ontology needs to specify the business objects of interest together with their life cycles. The textual descriptions of tasks (or other elements) are matched against the entities of interest in the domain ontology using linguistic methods for term similarity, synonyms, etc. Another way to restrict the matches is by employing the process structure, e.g., not suggesting the same activity twice, or comparing the process control flow to the object life cycle.

## 3 Consistency Checking

The consistency checking is an extended form of process verification: it deals with inconsistencies between control flow and semantic annotations. Specifically, we check two properties: Are the semantic preconditions guaranteed to be true whenever a task is activated? Are there conflicting tasks that may be executed in parallel? The basic steps taken for answering these questions are the following:

- **Compute the parallelism relation.** For every task, determine the set of tasks that are potentially executed in parallel. This is done by propagating a matrix through the process graph, where matrix entry $(i, j)$ is true iff tasks $(T_i, T_j)$ may be executed in parallel.
- **Detect conflicting parallel tasks.** Two parallel tasks $T_i$ and $T_j$ are in *precondition conflict* if $pre_i$ (the precondition of task $T_i$) contradicts $post_j$, and they are in *effect conflict* if $post_i$ contradicts $post_j$.
- **Detect non-supported preconditions.** We designed a propagation algorithm that associates every edge $e$ with the *intersection*, $I(e)$, of all logical states that may arise while $e$ is activated (carries a token). In other words, $I(e)$ contains exactly those literals which are always true when $e$ is activated. Hence, if task $T_i$ has $e$ as its incoming edge, then we can test $T_i$'s precondition for inclusion in $I(e)$. If the precondition is not included, then we know that there exists a process execution in which $T_i$ is activated from a workflow point of view, but is not executable according to its semantic annotation.

## 4 Task Discovery and Composition

Our first step in finding process task implementations is to discover a single semantic Web service for each annotated task. For each task, we check whether the domain ontology concept annotating the service matches the concept annotating the task. We follow a matching technique proposed in [7], analysing intersection of ontological elements in service descriptions and rating two descriptions as relevant whenever they overlap. For that, we use the standard reasoning task of *concept satisfiability* of conjunctions of concepts.

If a Web service cannot be found for a task, composition is performed. In the spirit of [6], we formalize the semantics of composed services based on the

notion of "belief updates" from A.I. [9]. Composition is computationally hard and has two main sources of complexity: (i) combinatorial explosion of possible compositions, and (ii) worst-case exponential reasoning. We tackle (i) using heuristic search, a well known technique for dealing with combinatorial search spaces. We adapt techniques originating in A.I. Planning [4]; ours is the first heuristic of this kind that deals with ontological axiomatizations. We address (ii) by trading off expressivity of the background ontology against efficiency: we investigate *tractable classes* of ontologies, i.e., classes where the required reasoning can be performed in polynomial time. We show that a combination of binary clauses (such as subsumption relations and attribute domain/image typing) and attribute cardinality bounds is tractable. It follows from [3] that Horn clauses (corresponding to Logic Programming rules) are not tractable; we design tractable approximate update-reasoning techniques that preserve either soundness or completeness. Other features (such as QoS) are part of our ongoing work. Our experiments show that the tool can create non-trivial composed services, from repositories containing hundreds of services, within a few seconds.

## 5 Conclusion

We present extensions to Maestro for BPMN. The demo shows how business analysts can easily annotate process elements (in particular tasks). Maestro checks whether the annotated process model is consistent in terms of its workflow and the semantics of its tasks. Maestro automatically finds and composes Web services that can realize individual tasks. Overall, this provides significant support for the service-based implementation of process models.

## References


1. W. Abramowicz, A. Filipowska, M. Kaczmarek, and T. Kaczmarek. Semantically enhanced business process modelling notation. In *SBPM Workshop*, 2007.
2. M. Born, F. Dörr, and I. Weber. User-friendly semantic annotation in business process modeling. In *Hf-SDDM Workshop*, December 2007.
3. T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.
4. J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. AI Research*, 14:253–302, 2001.
5. Jörg Hoffmann, Ingo Weber, James Scicluna, Tomasz Kacmarek, and Anupriya Ankolekar. Combining scalability and expressivity in the automatic composition of semantic web services. In *ICWE*, 2008.
6. C. Lutz and U. Sattler. A proposal for describing services with DLs. In *DL*, 2002.
7. D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW*, pages 89–98, 2002.
8. Ingo Weber, Jörg Hoffmann, and Jan Mendling. On the semantic consistency of executable process models. In *ECOWS*, 2008.
9. Marianne Winslett. Reasoning about action using a possible models approach. In *AAAI*, pages 89–93, 1988.