Semantic Annotation and Composition of Business Processes with Maestro*

Matthias Born[1], Joerg 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 | joerg.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. One of the main problems when creating execution-level process models is finding implementations for process activities. Carrying out this activity manually can be time consuming, since it involves searching in large service repositories. We present Maestro for BPMN, a tool that allows to annotate and automatically compose activities within business processes. We explain the main assumptions and algorithms underlying the tool, and we overview what will be demonstrated at ESWC.

1 Introduction

One of the biggest challenges within Service Oriented Architectures (SOA) is the composition of different Web services, achieving a higher utility. While in the realm of Web services such a combination is usually presented as a new service, we focus on how such compositions can be used as partial implementations of business processes, which is one of the key aspects of the SUPER project.

Web services need to be formally annotated in order for tools to automatically compose them into an orchestration (defining the control flow between them). We tackle the composition problem as a key part of the question on how to implement a business process with a set of given Web services. Business processes are often modeled as a set of activities (or tasks) together with their control flow. In order to make a process executable, e.g., in a workflow execution engine, all tasks in the process have to be carried out manually or automatically by Web services. Our aim is to semantically annotate such tasks and to automatically discover or compose (if needed) the services which collectively implement the required functionality. A business process modeling tool serves as a main user interface for performing these activities. This article focuses on the extensions made to *Maestro for BPMN*, a modeling tool from SAP Research. These extensions enable semantic annotation of the business process and automatic discovery and composition of Web services for this process.

* This work is partly funded by the EU 6th Framework Programme, within Information Society Technologies (IST) under the SUPER project (http://www.ip-super.org). Thanks also to Alina Dima, Florian Dörr, and Mario Karrenbrock for their coding support and Christian Drumm and Christian Brelage for their advice.

In the last years, Business Process Modeling Notation (BPMN) has received wide attention as a graphical representation of business process models. Within SUPER, business processes are stored as ontology descriptions. The *sBPMN* ontology[1] [1] serves as a meta model for BPMN process models, featuring the concepts, relations and attributes for standard BPMN. This ontology has been extended, mainly featuring the ability to define a state of the process before and after execution of successive activities. With these extensions we can derive semantic goal descriptions for activities – i.e., formal descriptions of the functionality which an implementation of a particular task needs to perform. This is in line with most popular approaches to Semantic Web Service description[2] where Web services can be annotated with pre and postconditions.

Handling semantic descriptions for activities was one of the features added to Maestro for BPMN. We also equipped it with the ability to call a composition tool with the annotated tasks as input[3] and integrated its output back to the user interface provided by the modeling tool.

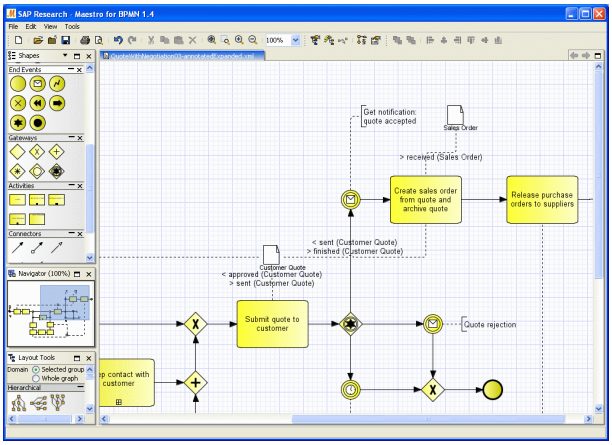

Fig. 1. A fragment of a process model represented in Maestro for BPMN.

Within the composer, we tackle some of the interesting opening issues in the Semantic Web Service Composition (WSC) area. We define a formal framework for WSC, inspired by A.I. Planning methodologies [9, 3]. We consider plug-in matches, where services do not have to match exactly, but have to be able to connect in all possible situations. In particular, we take the background ontology into account during the composition process; in contrast, many existing works assume exact matches (of concept names). The distinguishing feature of our work on composition is that we explore restrictions on the background ontology in order to find a solution (i.e., a composition) efficiently.

[1] sBPMN is written in WSML (http://www.wsmo.org/TR/d16/d16.1/v0.21/)
[2] Followed for example in WSMO (http://www.wsmo.org)
[3] Note that the annotated tasks will in effect be equivalent to a WSMO Goal

2 Process Modeling

From the graphical point of view, Maestro for BPMN follows BPMN. 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 pane, an instance of the concept *Task* is created in the in-memory working ontology. This enables supportive reasoning over the working ontology. The underlying conceptual work on the ontology and design choices are documented in [2].

The main goal of the tool extensions is to allow a user-friendly semantic annotation of process models. This is achieved by allowing to link semantically expressed process activities to a domain ontology. We focus on how process activities manipulate business objects in terms of their life cycles. E.g., a task "Send offer" sets the status of the object "Offer" to the state "sent". For this purpose, the domain ontology needs to specify the business objects of interest together with their life cycles[2]. This technique enables the user to define formal pre and postconditions of tasks in a human-friendly way.

For creating such links, we implemented matchmaking methods that filter the domain ontology based on the process context and rank the concepts to include the pre/postconditions. The textual descriptions of tasks (or other elements) are matched against the entities of interest in the domain ontology using linguistic methods, such as the edit distance between strings. E.g. if a task has label "Send offers", then the object "offer" from the domain ontology may be suggested as a top match. Another way to restrict the set of matches is by employing the process structure, e.g., by not suggesting the same activity twice or by comparing the process control flow to the object life cycle. The extensions made are conceptually independent of the tool chosen, and could be ported to other modeling notations.

3 Task Discovery and Composition

As a first step in finding process task implementations, we try to discover a single Semantic Web Service (SWS) for each annotated task. To achieve this, we check if the concept from the domain ontology describing a SWS matches the concept used for annotating the task. We follow a matching technique proposed in [8], analysing intersection of ontological elements in service descriptions and rating two descriptions as relevant whenever they specify an overlapping functionality. For that, we use standard reasoning task of *concept satisfiability* of a conjunction between the concepts taken from the task and Web service descriptions [7].

If a Web service cannot be found, WSC is performed. This 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 address (ii) by trading off expressivity of the background ontology against efficiency, i.e., we investigate restricted classes of ontologies allowing reasoning to be performed in polynomial time. Problem (ii) is closely related to the notion of "belief updates" in A.I. We define a clear formal model that combines

s_0 := *reasoning-startstate*(); (h, H) := *heuristic-function*(s_0); open-list := $\langle(s_0, h, H)\rangle$;
while TRUE **do**
    (s, h, H) := remove-front(open-list);
    if *is-solution*(s) **then return** path leading to s;
    for all applicable calls a of SWS in H **do**
        s' := *reasoning-resultstate*(s, a);
        (h', H') := *heuristic-function*(s');
        insert-ordered-by-increasing-h(open-list,s',h',H');

Fig. 2. The main loop of our WSC algorithm.

these notions and those from planning techniques, following recent formalisations of WSC [6, 4] and use heuristic methods for efficient searching [5]. One of our results is a polynomial time reasoning over background ontologies with Binary Clauses. For Horn Ontologies, we use an approximate update-reasoning technique that still runs efficiently but sacrifices some precision, preserving either soundness or completeness. The restricted ontologies allow to describe (amongst others) subsumption hierarchies, cardinality bounds and image type restrictions. Other features (such as QoS) are part of our ongoing work.

The main algorithm of the composer is shown in Fig. 2. The algorithm performs a forward search in a space of states s corresponding to different situations during the execution of the various possible compositions. The key elements are the *reasoning-startstate*, *reasoning-resultstate*, and *is-solution* procedures - maintaining the search states and detecting solutions - and the *heuristic-function* procedure - taking a state and returning a solution distance estimate h as well as a set H of promising Web services by solving a relaxed version of the problem. The states are ordered by increasing h, which is a standard method called "best-first-search". The set H is used for *filtering* the explored SWS calls. Filtering is widely perceived to be essential in WSC since it "forces" to check the most promising services first, leading to a considerable speed-up of the search.

The performance of our *WSC* tool was tested on two testbeds: the Telekomunikacja Polska (TPSA) which defines how a service (e.g. VoIP) is created for a new customer and the Virtual Traveling Agency (VTA) whereby the user specifies the kind of services that she/he would like for a trip (such as flight and hotel). The composer was set up in different configurations: *Blind* uses neither h nor H; *Heuristic* uses only h; *Filtering* uses only H; *Full* uses both. The results are plotted in Fig. 3, showing how runtime scales over the number of available services N; N was increased by generating additional services through randomized modifications of the original services.

4 Demo Scope

The demo will show an example of a realistic Business Process (Fig. 1). We will first demonstrate how data objects are associated with tasks (annotation) and how the states of these objects can be attached to the pre and postconditions of the task. Discovery is then used to find a Web service that fulfils that particular task. We will also show a task for which composition is required (rather than discovery). The annotated task will serve as the input goal to the composer. The

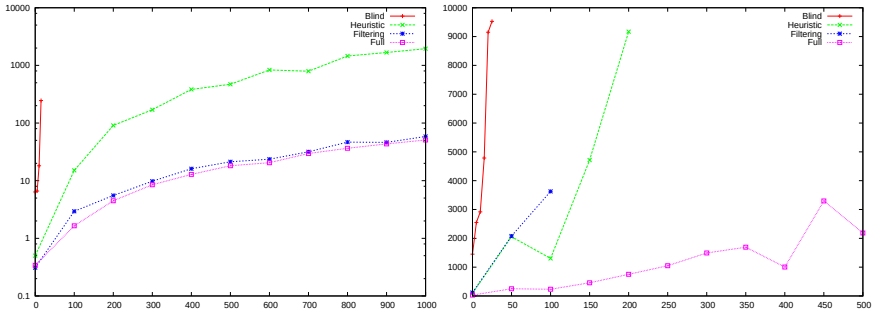

Fig. 3. Results for TPSA (left) and VTA (right), plotted as Seconds (y-axis) against N (x-axis)

component is run in the background and once a solution is found, the task in the process is replaced with the sequence of Web services found by the composer. The search for a solution will be performed within a large number of Web services such that the audience can clearly see the scalability of our approach.

5 Conclusion

We have presented extensions to Maestro for BPMN, which demonstrate tool support for semantic annotation of the process models, as well as task discovery and composition. This enables more agile business process development and deployment. We showed how business analysts can easily annotate process elements (in particular—tasks) and automatically find Web services that fulfil them using discovery. If the latter fails, an efficient composer tool that we developed can be used to find a chain of Web services that can adequately fulfil the task.

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, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A logic programming approach to knowledge-state planning: Semantics and complexity. *Transactions on Computational Logic*, 5(2):206–263, 2004.
4. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the approximation of instance level update and erasure in description logics. In *AAAI*, 2007.
5. J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. AI Research*, 14:253–302, 2001.
6. C. Lutz and U. Sattler. A proposal for describing services with DLs. In *DL*, 2002.
7. I. Markovic and M. Karrenbrock. Semantic web service discovery for business process models. In *Hf-SDDM Workshop*, December 2007.
8. D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW*, pages 89–98, 2002.
9. Marianne Winslett. Reasoning about action using a possible models approach. In *AAAI*, pages 89–93, 1988.