

Auto-completion for Executable Business Process Models*

Matthias Born¹, Christian Brelage¹, Ivan Markovic¹, Daniel Pfeiffer², Ingo Weber¹

¹SAP Research Center CEC, Karlsruhe

{mat.born, christian.brelage, ivan.markovic, ingo.weber}@sap.com

²European Research Center for Information Systems (ERCIS), Münster. pfeiffer@ercis.de

Abstract. This work presents an auto-completion mechanism for supporting the creation of executable business process models. Currently, process modeling tools provide only little support to identify the relevant services that are needed to execute the process model – the selection of appropriate services is left to the skills of the modeler. A novel solution technique for this problem is proposed here as the combination of (1) a context-based analysis, (2) by taking pre and post-conditions into account, and by (3) evaluating the non-functional properties of the functionally and context-wise fitting services.

1 Introduction & Problem description

Business process modeling has evolved into an important means for the collection, documentation, and analysis of processes in companies and public administrations. Recently, the focus of business process modeling has changed from just being a means for documentation towards enabling process enactment. The constantly growing interest in service-oriented architectures (SOA) has moved the identification and composition of services in the focus of research and practice. Business process models are commonly considered as an appropriate vehicle to support this task. All major vendors of ERP software have published SOA-based strategies and have stressed the important role of process modeling in it.

However, using current process modeling approaches and tools is still very costly and error-prone since users are not guided in any sensible way. Major training investments are needed to educate people in business processes modeling. In particular the problems below hinder a more widespread usage of process modeling:

- Transition from “sense-making” models to engineering models is done manually and, thus, error-prone, costly, and slow.
- The waterfall-like modeling paradigm of continuous refinement does not reflect the interactive, “going-back-and-forth” nature of the modeling process. In particular, we claim that an early identification of technical artifacts (such as Web services) has the potential to improve the model quality significantly during the modeling activities. The approach proposed

* This work has in part been funded by the EU in the FP6 IST project SUPER, ip-super.org

herein allows matching (technical) services to conceptual modeling elements as early as possible in the modeling process.

- Current process modeling tools provide only little guidance to the user and, thus, resemble a scratchboard rather than a technical approach. If any guidance is provided at all, it is on a purely syntactical level which does not account for business semantics. The approach describe herein allows tools that factor in business semantics and, thus, make suggestions that follow business logic.
- Finally, the approach speeds up the modeling process by making sensible suggestions and by implementing trivial tasks (semi-)automatically.

In this paper we introduce a mechanism that offers a list of services whose entries represent potential successors in the model. This list is compiled based on the actual state of the process model. We call this functionality an auto-completion mechanism.

The paper proceeds as follows: we next present the technical structure of our novel solution technique; subsequently, we provide an overview of related work and close with a summary of the main result and a discussion of our contribution. For brevity, we refer the reader to more detailed descriptions of various technical contributions.

2 Solution technique

The solution technique we describe in the course of this work consists of three different sources, namely process context-based analysis, pre- and postcondition analysis, and non-functional property analysis. These three different techniques are used to identify appropriate services from a repository and the results of these three independent techniques are weighted and consolidated. Finally, the resulting list of proposed services is presented to the user within the process modeling tool. The general architecture of the proposed solution technique is shown in Fig. 1.

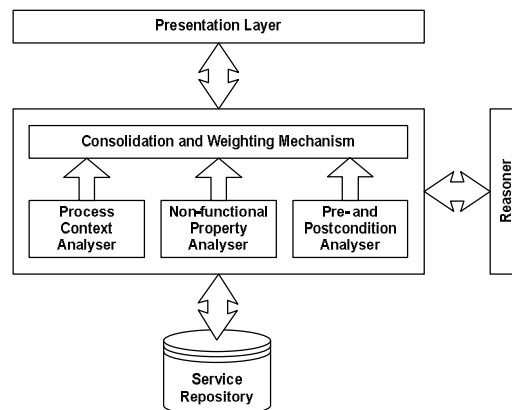


Fig. 1. Architecture of the proposed auto-completion technique.

In the following subsections we will discuss the three aspects of our technique in more detail and describe a mechanism for consolidating and weighting the individual aspects.

2.1 Process context-based auto-completion

In our opinion, the process context is composed out of the following perspectives: Business Function, Business Domain, and Business Goal, which, in our context, are modeled in respective process ontologies. These ontologies provide an unambiguous, explicit, and machine-readable categorization of processes by three basic aspects: the functional, domain, and intentional aspect.

The Business Function Ontology (BFO) provides a structural breakdown of the organization's business functions. It does so by splitting the domain in two dimensions, namely vertical and horizontal. The vertical dimension classifies processes into various process areas, such as: procurement, manufacturing, warehousing, order fulfillment, etc.; the horizontal dimension differentiates the key functional areas across process areas, describing concepts such as customer relationship management, supplier relationship management, etc. Concepts from this ontology classify processes by their functionality, independent of the business domain. The Business Domain Ontology complements BFO and describes the domain inside the organization where the process is used. Examples of business domain concepts are: product area, localization area, etc. Business Goals indicate what a process needs to achieve from the business perspective and provide answers to why processes exist in the organization. The Business Goal Ontology models a hierarchy of business goals and provides a set of relations between them. For more details on our formal process representation, we refer the reader to [5].

By annotating processes and Web services with the concepts from the aforementioned ontologies, we are able to perform context-based matching in order to find the Web services which contextually match a given process. The matchmaking mechanism that can be used here is described in detail in [6].

When performing matchmaking, we utilize the ontology structure for ranking of the matching results. Here, we measure the semantic distance between the concepts annotating a process model and those annotating services. Services annotated with concepts having the smallest semantic distance obtain the highest rank in the list.

With this approach we can ensure that the discovered services are fitting in the specific context of the process models. However, we cannot guarantee that the proposed services can eventually be executed as the current state of the process might not fulfill the requirements of the service.

2.2. Auto-completion with pre- and postconditions

There is a substantial body of work on the problem of functional discovery, i.e., matching services against a request on the basis of pre- and postconditions. The *precondition* of a service is a logical formula that describes under which circumstances the service is able to execute: if the state of the (relevant part of) the

world satisfies the logical formula when the service is invoked, then its execution is possible and defined. Similarly, the *postcondition* (also called effect) describes how the service changes the world, i.e., how the state when invoking the service is altered by the service. Given the pre- and postconditions of an anticipated service (also referred to as a *goal*) and a repository of services, these approaches can determine the degree to which each service matches the goal. In particular the best match or the set of best matches can be determined.

In principle, any such approach can be used for the purposes here, where functional discovery is used for auto-completion. This can be achieved by coming up with the suitable pre- and postconditions for the part of the process where a service is to be added. The precondition can be derived from the process as modeled so far [2] by propagating the postconditions of process activities [1] that will be executed before the current position in the process will be reached. Then, all services with a (partially) satisfied precondition can be selected for auto-completion, where the matching degree of the services can be taken into account by the ranking. Further, potential conflicts with parallel process parts can be detected upfront [2] and the auto-completion can either exclude the conflicting services or adapt the ranking accordingly.

Using the pre- and postcondition analysis we can be certain that the proposed services can be executed at runtime as the states of a service are matched against the states of the process. Nevertheless, this approach might list services that are unsuitable in a specific process context.

2.3. Auto-completion with non-functional properties

Non-functional descriptions of Web services, in particular quality parameters such as price, availability, and execution time, can also be used in the auto-completion process. This step is best performed after the process context-based and functional matchmaking provided a list of services. In this step, we evaluate the level of match between the desired non-functional properties against those offered by services obtained through pre-filtering in the previous two steps.

The user will have the possibility to specify his preferences regarding the predefined set of quality of service criteria in the modeling tool, as described in [4]. Techniques presented in [4] can be used as a basis for machine-readable representation of non-functional aspects as well as the matchmaking and ranking mechanism. By taking into account the quality of service properties when proposing follow up activities, we increase the level of precision of the suggestions.

2.4. Auto-completion based on combined criteria

This section describes how the “Consolidation and Weighting Mechanism” from Fig. 1 can be implemented and how the weights may be adjusted based on user feedback. The list of suggestions can be consolidated by making sure that each service appears only once and by ordering the combined list through a unified view on the matching degree. We suggest doing so by calculating a weighted sum of the individual

matching degrees. We hereby assume that the matching degree is expressed as a value ≥ 0 . Then the formula for the weighted sum is:

$$MD_{Sum} := MD_{Context} \times W_{Context} + MD_{Functional} \times W_{Functional} + B_{NFP} \times MD_{NFP} \times W_{NFP} \quad (1)$$

where the right-hand part of the formula comprises MD as the matching degree and W as the weight of either the Context (Section 2.1), the Functional discovery (2.2) or the Non-Functional Properties (2.3). We hereby assume that the absence of a match is signaled by a $MD=0$, and the better a service matches the higher the respective MD . B_{NFP} is a switch: if $MD_{Context} = 0$ and $MD_{Functional} = 0$ we set $B_{NFP} := 0$; otherwise we set $B_{NFP} := 1$. Thus, if a service is no match in terms of the Context or the Functional Discovery, then the combined MD_{Sum} is set to 0; otherwise it is a standard weighted sum. A non-functional match can improve the MD_{Sum} , but if the service is not of interest with respect to functionality or context, then the MD_{NFP} is of no relevance.

The goal of this calculation is to have a single unified view on the matching degree, namely MD_{Sum} . Then, the list of matches can be ordered according to MD_{Sum} and, optionally, also pruned so that, e.g., only the 15 best matches are displayed.

While the weights may be set based on experience or importance of the respective technique, they can also be learned from user behavior. Machine-learning approaches such as Reinforcement Learning [3] can learn user preferences from actions. This could be used by allowing the system to adjust the three weights on its own and by interpreting the user selection from the list as the feedback: the higher the ranked service which the user selects, the better. In particular, multiplying the position of the selected service in the list with “-1” can serve as the so-called feedback function. E.g., when the seventh service in the list is selected, then the feedback is -7. This is better than a -10, but worse than a -1. The following two points raise interesting questions:

- List places which are displayed directly, e.g., the top three or top five choices, could be set to the same feedback value, arguing that the top places in the list are all reasonably good.
- If none of the services in the list applies, the presented services may be the wrong ones due to false weights or the information on which the choices are presented may be misleading or inaccurate. Therefore, the question is whether this should result in a strong negative feedback value or not taken into account by the learning system.

A practical evaluation is necessary to give answers to these questions and to obtain meaningful weights.

3 Related Work

Our approach is related to the work of [7], which is also based on semantic business process modeling. The author describes methods and techniques to support modeling of semantically annotated business processes in the form of Petri Nets. The work presented by Koschmider focuses on measuring the similarity of process task labels in order to suggest possible process fragments whereas our approach has a clear linkage to implementation-level services.

In [8], the authors present an approach for supporting the modeling of business processes using semi-automated Web service composition techniques. The main difference to our work is that it only takes into account the functional part of service descriptions when making suggestions during modeling. We consider the process context and non-functional properties in addition to the functional properties, thus increasing the level of precision of the suggestions.

4 Conclusions

This work presents a novel approach to improve current business process modeling tools. Our solution supports the process modelers actively during their modeling tasks and thus helps them in finding appropriate services more intuitively. Eventually, this will lead to process models with higher quality and reduce the actual modeling time, as the process modeler gets active support from the process modeling tool.

Our auto-completion mechanism rates the relevance of services based on the following three different techniques: i) process context-based auto-completion, ii) auto-completion with pre- and post-conditions, iii) auto-completion with non-functional properties. Using a mechanism to weigh the relevance of the three techniques, the system will eventually suggest meaningful services to the modeler.

In contrast to existing process modeling tools we claim that our solution is a step towards more intuitive and guided process modeling tools. Some of the major benefits are i) early identification of technical artifacts which match to conceptual model elements, ii) sensible suggestions for the users based on business logic and not only on mere syntactical information, iii) improved model quality and faster modeling process by (semi-)automating certain modeling tasks.

References

1. Weber, I., Hoffmann, J., Mendling, J. Semantic business process validation. In Proc. Semantic Business Process Management (SBPM'08) Workshop at ESWC, 2008.
2. May, N. and Weber, I. Information Gathering for Semantic Service Discovery and Composition in Business Process Modeling. In Proc. CIAO! Workshop at CAiSE, 2008.
3. Sutton, R. and Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998
4. Vu, L.H., Hauswirth, M., Porto, F., Aberer, K.: A search engine for QoS-enabled discovery of semantic web services. Intl. J. Business Process Integration and Management 1, 2007
5. Markovic, I., Pereira, A.C.: Towards a formal framework for reuse in business process modeling. In Proc. Business Process Management Workshops, Brisbane, Australia, 2007.
6. Markovic, I., Pereira, A. C., de Francisco, D., and Muñoz, H. Querying in business process modeling. In Proc. of the ICSOC Workshops, Vienna, Austria, 2007.
7. Koschmider, A. Ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse. Karlsruhe University Press, University of Karlsruhe, Karlsruhe, 2007.
8. Schaffner, J., Meyer, H., Weske, M.: A Formal Model for Mixed Initiative Service Composition. In Proc. Intl. Conf. Services Computing (SCC), Salt Lake City, USA, 2007.