A Recommender System for Business Process Models

Thomas Hornung
Institute of Computer Science,
Albert-Ludwigs University Freiburg,
Germany

hornungt@
informatik.uni-freiburg.de

Agnes Koschmider, Andreas Oberweis
Institute of Applied Informatics and Formal
Description Methods,
Universität Karlsruhe (TH), Germany

koschmider|oberweis@
aifb.uni-karlsruhe.de

Abstract

Process modeling facilitates understanding and restructuring of activities used to achieve business goals. However, manual process modeling is time-consuming and error-prone. In this paper, we describe a recommender system that suggests a list of correct and fitting process fragments for an edited business process model, which can be used to complete the process model being edited. The recommender system supports the user during the process modeling phase and reduces the number of structural modeling errors.

1. Introduction

Precise modeling of business processes has paved the way to realize process-aware information systems [3] that include allocations of resources, communication services, or hardware devices to users. The modeling and implementation effort of information systems could be substantially reduced if available business process models were reused. An automatic reuse of business processes requires appropriate techniques, which support the access to available business process fragments.
The aim of our research is to decrease the modeling time and to assist users during process modeling by a recommender system, which is based on business process fragment reuse. The foundations for the recommender system are available business process models, which are stored in a process model repository. “Bad” and “good” examples of available process fragments for an edited business process are given in Figure 1.

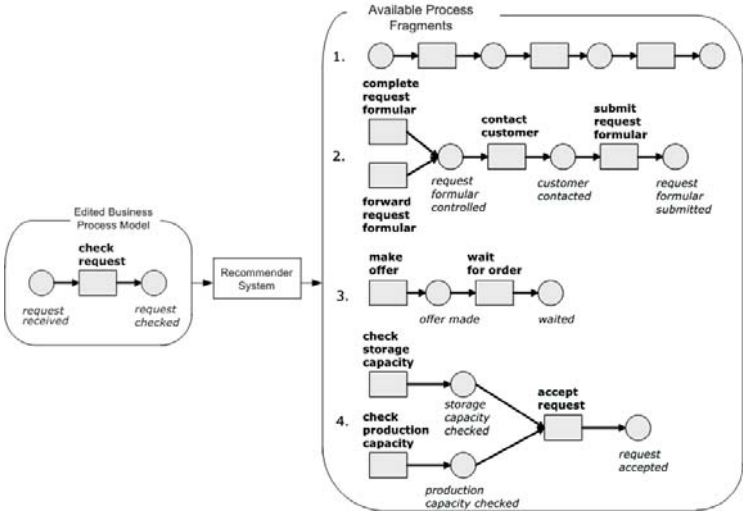

Figure 1: Pool of available process fragments

The process fragments are modeled with Petri nets [9], which support the modeling of complex business process models and the related business process objects. The formal foundation of Petri nets and thus the available analysis methods for Petri nets provide for a correct syntactical interconnection of the edited business process model and the recommended process fragments. In order to incorporate the analysis methods in our application scenario we distinguish between two correctness criteria for interconnection. The first criterion is a syntactically valid connection between the element where the recommendation system has been invoked and the first element of the proposed process fragment. A Petri net is a bipartite graph where only places can be connected with transitions. This criterion is violated for the first process fragment in Figure 1 (a place would be connected with a place). Thus, this process fragment will not be recommended for the edited business process. The second correctness criterion is the compliance of structural properties such as well-structureness, which is formalized in [10]. A Petri net is well-structured if AND-split's are complemented by AND-join's and OR-split's are complemented by OR-join's. This structural property for business processes is violated if for example a parallel flow initiated by an AND-split is synchronized by an OR-join. The benefit of this property is a "good" process modeling style, which makes understanding of the processes models easy and supports the detection of undesirable deadlocks. Well-structured models can be checked for soundness in polynomial time [10]. This second correctness criterion is violated for the fourth process fragment because an OR-split is synchronized by an AND-join. This interconnected business process will cause undesirable deadlocks and thus the fourth process fragment will not be recommended for the edited business process. After validation of the correctness criteria the second and the third process fragments remain for recommendation.

Additionally, we regard the fitness of process fragments as a criterion for process recommendation. A process fragment is regarded as a fitting one if it matches in respect of content and if it possesses the same abstraction level. The vocabulary used to name elements in the edited process model can be different from the one used to label elements stored in the process repository. The detection of different terms for the same process object names requires a significant amount of experience in the field of process engineering and may result in extra analyzing efforts. In order to automatically uncover synonyms or homonyms in business process models we adopt the semantic similarity measures that were presented in [1]. The similarity degree between process models can be determined when considering the semantics of element names with four similarity measures. To make assertions about the similarity between process models the four calculated similarity degrees are aggregated to a combined similarity. Due to the degree of the combined similarity fitting process fragments can be detected and proposed for recommendation even when a different vocabulary for the same process objects is used.

The second business process fragment fits in respect of content (it models the handling of requests) while the abstraction level of the edited business process and this second process fragment differ. Each decomposition level describes process elements from a different abstraction level. Top level process models formulate an overview of process activities and bottom models provide more detailed descriptions. In order to improve model consistency users have to maintain a homogeneous level of abstraction of process element names on the same decomposition level. The abstraction level can be identified when measuring the linguistic specificities of element names. Our measurement is based upon the hypothesis that in fine-grained processes element names are more specific than in coarse-grained ones. Specificity of names indicates their process decomposition level and thus their abstraction level. The second process fragment models in detail the handling of requests and can be considered as a refinement of the transition *check request*. Thus, this process fragment will not be recommended. The third process fragment describes how to handle offers, which fits in respect of content and abstraction to the edited business process. Only this process fragment will be recommended for the edited business process (if only the four process fragments are stored in the repository).

In this paper we will describe a novel recommender system and show how to infer correct and fitting process fragments for an edited business process model. Section 2 presents requirements for such a recommender system. Section 3 describes an extension of the Petri net model and sketches predicates to support process fragment inferencing. The paper concludes with an outlook on future work.

2. Requirements Analysis

In the following we will sketch out some required features for the recommendation system which we have defined for our application scenario.

- Transparency

Transparency of business process models comes up with a restricted number of process elements in one process model diagram. An appropriate number (practically tested by us in several projects) are 15 elements, which should be considered when proposing correct and fitting process fragments. If the user has already modeled 5 elements and invokes the recommendation system then each proposed process fragment should not exceed 10 elements. The amount of recommended process elements correlates with the choice of fitting process fragments. To decrease the modeling time by a recommender system the number of proposed fragments should not excess four or a ranking function should sort the relevance of the proposed process fragments to the edited business processes. One fragment should be composed of at least five and at most ten elements so that the modeling support is beneficial for the user. We made a face-to-face interview (in German) with 55 persons with different modeling experiences. A process fragment with four or less elements, which fits to the edited business process, has not been selected at all.

- Flexibility

When proposing fitting process fragments the different user types (unaware, beginner, advanced, expert) have to be regarded in order to satisfy an individual flexible recommendation. Due to their comprehensive modeling experiences, experts can faster understand the recommendations and thus easier choose the fitting process fragments. Beginners need more time in order to make a decision. The amount of process fragments recommended for experts can be higher than for beginners. Furthermore, the quality of business processes that were modeled by experts is higher than the quality of models created by beginners. The business process models, which are stored in the process repository, should therefore be annotated with the user type in order to allow reliable assertions about the quality of process fragments.

- Modeling technique

Business processes can be modeled forward-oriented starting at the process trigger or backward-oriented starting at the process result. For the sake of practical applicability the recommendation should be available when modeling processes successively from the left to the right hand side and vice versa.

- Properties of the process fragments

The storage of process fragments in the process repository requires a priori analysis, which should guarantee that the fragments satisfy e.g. the property of free-choiceness (it is not allowed to mix choice and synchronization). Consequently, the fragments to be reused by the recommendation system already fulfill several “correctness” criteria.

3. Realization

In order to support (semi-)automatic recommendations of correct and fitting process fragments we describe Petri nets with the Ontology Language OWL [8], resulting in so-called semantic business process models [7]. This OWL-based description of Petri nets makes it possible to (semi-) automatically handle and manipulate business process models. Each Petri net element has a corresponding concept in the OWL-based description. The set of places corresponds to the concept `Place`, the set of transitions to the concept `Transition`. To fully incorporate the above mentioned semantic business process models into the recommender system, we need a rule language that is capable of reasoning over OWL classes and individuals. A suitable format for the use of reasoning mechanisms is the Semantic Web Rule Language (SWRL) [5]. There are several implementations

available for the rule language SWRL. Additionally, the SWRL language proposal explicitly defines a list of supported built-ins, which enables users to realize predicates as functions in an optional programming language. In the SWRL specification rules are defined in the following form: *antecedent* → *consequent*, where antecedent (body) and consequent (head) consist of zero or more predicates, and multiple predicates are treated as a conjunction.
To infer correct and fitting process fragments for an edited business process model we defined 18 SWRL rules, which allow using reasoning techniques. Furthermore, we extended the OWL-based description of Petri nets with two properties. The first OWL property `isSelected` with the range boolean (true or false) ensures that appropriate recommendations are made only for the selected process element. The second OWL property `initialElement` indicates the start element of a process.
Subsequently, we will sketch SWRL predicates, which maintain the requirements during the modeling support as explained in the last section. Other rule languages would have been possible as well, e.g. we are considering Prolog's [2] OWL support at the moment as a possible alternative, since it has wide-spread support and fast reasoning engines are available.
We will introduce two SWRL predicates in order to guarantee a correct interconnection of the currently edited process and the recommended process fragments:

1. `arcPossible(?node,?net)`: Satisfied, if there is a valid syntactical connection between the selected node and the process fragment net.
2. `branching(?node,?net)`: Satisfied, if the selected node is an element of a forward split and the process fragment net contains the corresponding backward join (hence the property well-structureness is satisfied).

Additionally, we check that the number of start elements matches the number of selected nodes.

To include the semantic similarity in our reasoning process which guarantees appropriate process fragment recommendations, we introduce a new built-in predicate into SWRL denoted as

1. `semSimilarity(?node1,?node2,?arg)`: Satisfied, if the semantic similarity of node1 and node2 is greater than or equal to ?arg (?arg is a similarity degree).

We regard the abstraction level in the modeling support for business processes via the predicate:

2. `semAbstraction(?node,?net)`: Satisfied, if the selected node and the recommended process fragment are at the same abstraction level.

In [6] two algorithms are presented, which compare the abstraction level of interconnected business process models based upon so-called refinement patterns. We use the term refinement pattern as our abstraction level detection system isolates element name sequences of one process regarding abstraction homogeneity and heterogeneity and proposes refinement in case of nonuniformly specified names.

Usually, when modeling business processes, users have in mind internal business restrictions and policies. Thus, one feature of the recommendation system has to fit to specific business rules that are pertaining to the enterprise respectively the modeled real-world excerpt. Rule-based modeling in general ranges from Event–Condition–Action rules for databases to logic-based languages like Prolog, and rule engine approaches like Jess. We distinguish the following three types of rules in our business process modeling support scenario [4]:

- Constraint rules: They represent assertions that constrain the domain of business facets, e.g. "customers under 18 cannot make an online reservation".
- Action initiated rules: They concern the integrity of semantic business process models and have a syntax described by IF, THEN, AND, OR, e.g. "IF customer order is checked THEN manufacture item AND send article".

- Dynamic rules: These rules are applied during process modeling by the modeler and may vary from one context to another, e.g. “in a specific context an amount of 1000 Euro is considered as a high loss”.

The next step is to formalize the rules which were classified before. To do this we first introduce the notion of order to denote, which action is performed before or after the other. This is formalized with the SWRL predicates `before(node1,node2)` and `after(node1,node2)`, which evaluate to true, when node1 appears before node2 in the business process model or after respectively.
To infer action initiated rule style we compare the currently modeled process with given serializations of business process fragments in a repository. This is best illustrated by an example: given the business rule “IF request is checked THEN forward order” we would check the following conditions with the help of the before mentioned predicates:

1. The condition(s) of the IF part have to appear before the selected element,
   and
2. The action(s) of the THEN part need to be either the first action of the process fragment or appear after the first element in the process fragment under consideration,
   and
3. The hypothetical connection of the currently modeled process fragment and the process fragment under consideration need to satisfy the constraint rules.

Finally, dynamic rules can be realized by changing the action part of a business rule to a question for the modeler. An automatic conversion from naturally described rules to SWRL rules requires to maintain the IF . . .THEN syntax. Otherwise, a correct conversion cannot be guaranteed. Currently, our recommender system neither checks rule consistencies nor addresses conflicts between rules.

Suppose the business rule repository contains several rules that are of interest for an edited manufacturing business process. One user denotes a rule “IF purchase order was handled THEN initiate payment”. The recommender system would try to find correct and fitting process fragments dealing with manufacturing orders and payment. The according SWRL representation is as follows:

$$
\begin{aligned}
&\texttt{isSelected(?node1,true)} \wedge\ \texttt{initialElement(?net,?node2)} \wedge \\
&\texttt{arcPossible(?node1,?net)} \wedge\ \texttt{branching(?node1,?net)} \wedge \\
&\texttt{beforeOrLast(?node}_{\texttt{condition}}\texttt{,?node1)} \wedge\ \texttt{afterOrFirst(?node}_{\texttt{action}}\texttt{, ?node2)} \wedge \\
&\texttt{semSimilarity(?node}_{\texttt{condition}}\texttt{, purchase\_order\_handled, 0.4)} \wedge \\
&\texttt{semSimilarity(?node}_{\texttt{action}}\texttt{, initiate\_payment, 0.4)} \wedge \\
&\texttt{semAbstraction(?node1,?net)} \\
&\rightarrow \texttt{isPossible(?node1,?net)}
\end{aligned}
$$

First the recommender system has to be invoked at one or more elements (note that the recommender system can only be invoked for elements of same set). Then the system verifies if there exists a valid syntactical connection between the selected element and the first element of the recommended process fragments and if the well-structureness property holds. Assuming the user has specified a threshold of 0.4 the differently used terms in the process elements and business rules will be treated as being similar and the condition of the business rule is fulfilled. Then the recommender system would propose fragments, which have a similarity between its first or succeeding elements and *initiate_payment* that is greater than or equal to 0.4. When recommending process fragments the system satisfies the requirements of transparency and flexibility. One current limitation in respect of the requirements remains: the processes have to be modeled starting at the process trigger.

4. Conclusion and Future Work

In this paper we described the implementation of a recommender system for business process modeling. The benefit of such a system is to assist users during process modeling by reusing process

fragments from a process repository. We defined required features for such a recommender system, which we evaluated in a face-to-face interview. These features should be supported in order not to limit its application in practice.
The simplicity of our approach makes it possible to apply our recommender system to other process modeling languages such as WS-BPEL or Event Driven Process Chains (EPC). More research work is required in order to store process fragments in different modeling languages and to recommend process fragments to modelers in the currently used modeling language.

References

[1] Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007), volume 67, pp. 71-80. Australian Computer Science Communications, Ballarat, Australia, 2007.

[2] Clocksin, W., Mellish, C.: Programming in Prolog. 3 edn. Springer, 1987.

[3] Dumas, M., Aalst, W. van der, ter Hofstede, A. (Eds.): Process-Aware Information Systems, John Wiley & Sons, 2005.

[4] Hornung, T., Koschmider, A., Oberweis, A.: Rule-based Autocompletion Of Business Process Models. In CAiSE Forum 2007. Proceedings at the 19th Conference on Advanced Information Systems Engineering (CAiSE). Trondheim, Norway, June 2007.

[5] Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/ (2004).

[6] Koschmider, A., Blanchard, E.: User Assistance for Business Process Model Decomposition. In First IEEE International Conference on Research Challenges in Information Science, pp. 445-454. April, Ouarzazate, Maroc, April 2007.

[7] Koschmider, A., Oberweis, A.: Modeling semantic business process models. In Rittgen, P. (ed.): Handbook of Ontologies for Business Interaction. Idea Group Publishing, 2007.

[8] McGuinness, D. L., van Harmelen, F.: OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (2003): http://www.w3.org/TR/owl-features/.

[9] Reisig, W., Rozenberg, G.: Lectures on Petri Nets I: Basic Models. Springer, 1998.

[10] van der Aalst, W.: The Application of Petri Net to Workflow Management. The Journal of Circuits, Systems and Computers, Department of Mathematics and Computing Science, Eindhoven University of Technology, http://is.tm.tue.nl/staff/wvdaalst/publications/ p53.pdf.