Rule-based Autocompletion of Business Process Models

Thomas Hornung[1], Agnes Koschmider[2], and Andreas Oberweis[2]

[1] Institute of Computer Science, Albert-Ludwigs University Freiburg, Germany
hornungt@informatik.uni-freiburg.de
[2] Institute of Applied Informatics and Formal Description Methods
Universität Karlsruhe (TH), Germany
koschmider|oberweis@aifb.uni-karlsruhe.de

Abstract Several methods based upon textual programming languages or graphical notations have been proposed for manual modeling of business process models. But since manual process modeling is time-consuming and increases the amount of structural modeling errors, we aim at supporting the user during the process modeling phase. In this paper we present an initial idea for an automatic approach for completion of business process models that recommends appropriate completions to initial process fragments based on business rules and structural constraints.

1 Introduction

Manual process modeling is a time-consuming task and thus increases the total amount of modeling time. Typos and structural modeling errors make it particularly error-prone to model business processes manually. Therefore it would be useful to assist the user in modeling business processes by providing an autocompletion mechanism during process modeling. Usually, business process models are modeled according to specific business rules such as *all high loss estimations must include an expert's inspection* or *if more than two persons travel together, the third pays only half price*. Therefore to support automatic completion of such business process models, a data format with support for rule inferencing is required. By using Petri nets as the theoretical framework for this task, syntactic composition problems of initial process fragments and recommended process fragments can be solved.
To be able to resolve conflicts on a semantical level, we introduce a description of rule-enhanced Petri nets with the ontology language OWL [1] and the rule language SWRL [2] in Section 2. Finally Section 3 sums up our approach.

2 Autocompletion Process

This section gives a high-level introduction in our proposal for an autocompletion mechanism for business process models as depicted in Figure 1. The idea is that the user first selects a position in the process model and the recommendation system shows a list of meaningful process fragments that she can enter at this position.

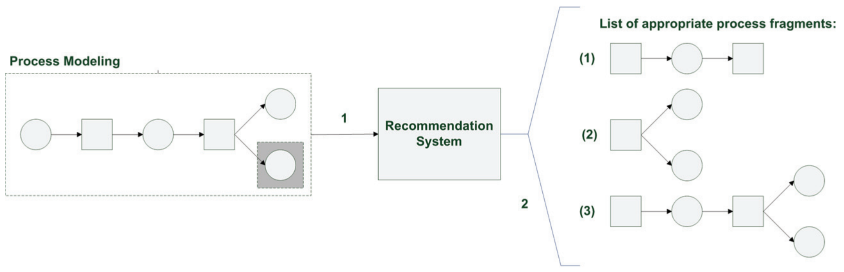

Figure 1. Rule-based autocompletion process (rectangles depict transitions and circles depict places)

2.1 Semantic business process models

A missing semantic representation of Petri net components hampers to utilize reasoning techniques that make it possible to (semi-)automatically reason about Petri net data. Therefore we describe traditional Petri nets with the Ontology Language OWL [3]. These so-called semantic business process models (SBPM) combine process modeling methods with semantic technologies to support automatic processing of process components and they make it possible to implement an efficient algorithm for (semi-)automatic similarity computation between process model variants [4]. Each Petri net element has a corresponding element in the SBPM as described in [1] with two extensions: first we introduce a property **isSelected** with the domain **Place** or **Transition** and the range **boolean** (true or false) that ensures that appropriate recommendations are made only for the selected process element. The property **initialElement** with the domain **PetriNet** and the range **Node** (**Place** or **Transition**) indicates the start element of a process as depicted in Figure 2.

2.2 Classification of rule types

Since ontologies are not good at modeling the dynamic state of Petri nets we additionally use the Semantic Web Rule Language (SWRL) to capture the dynamic nature of Petri nets and to model business rules that a process model needs to satisfy. Rule-based modeling in general ranges from Event–Condition–Action rules for databases [5] to logic-based languages like Prolog [6], and rule engine approaches like Jess [7]. We distinguish the following three types of rules in our autocompletion scenario:

1. Constraint rules: They concern the integrity of semantic business process models. These constraints are automatically pre-specified for every business process model by our autocompletion system, since they express restrictions on the OWL-based Petri net description, i.e. *successors of places are transitions and vice versa.*

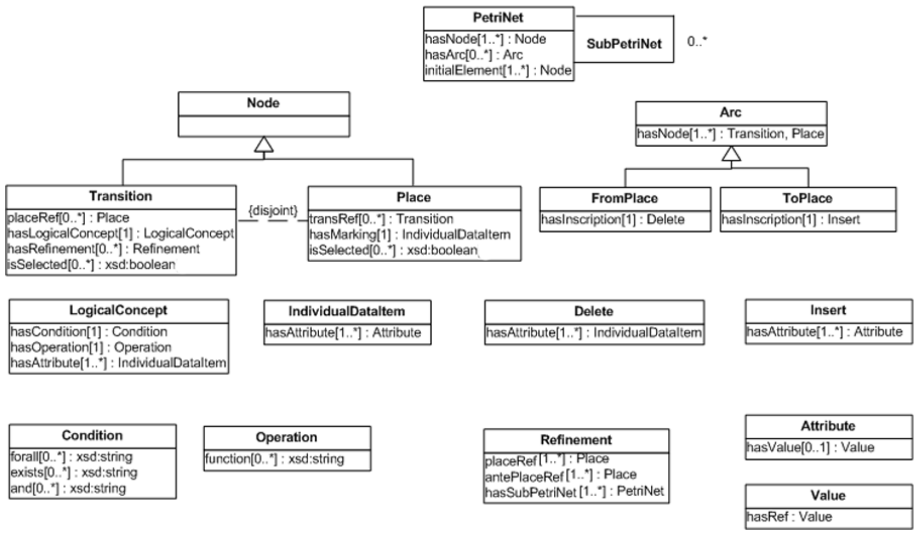
PetriNet
hasNode[1..*] : Node
hasArc[0..*] : Arc
initialElement[1..*] : Node
SubPetriNet
0..*
Node
Arc
hasNode[1..*] : Transition, Place
Transition
placeRef[0..*] : Place
hasLogicalConcept[1] : LogicalConcept
hasRefinement[0..*] : Refinement
isSelected[0..*] : xsd:boolean
{disjoint}
Place
transRef[0..*] : Transition
hasMarking[1] : IndividualDataItem
isSelected[0..*] : xsd:boolean
FromPlace
hasInscription[1] : Delete
ToPlace
hasInscription[1] : Insert
LogicalConcept
hasCondition[1] : Condition
hasOperation[1] : Operation
hasAttribute[1..*] : IndividualDataItem
IndividualDataItem
hasAttribute[1..*] : Attribute
Delete
hasAttribute[1..*] : IndividualDataItem
Insert
hasAttribute[1..*] : Attribute
Condition
forall[0..*] : xsd:string
exists[0..*] : xsd:string
and[0..*] : xsd:string
Operation
function[0..*] : xsd:string
Refinement
placeRef [1..*] : Place
antePlaceRef [1..*] : Place
hasSubPetriNet [1..*] : PetriNet
Attribute
hasValue[0..1] : Value
Value
hasRef : Value

Figure 2. Extension of the Petri net Ontology introduced in [1]

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

2.3 Formalization of rules

The next step of the autocompletion process is to formalize the rules which were classified in the previous section. To do this we first introduce the notion of order, i.e. which action is performed before or after the other. This is formalized with the SWRL predicates $before(action_1,\ action_2)$ and $after(action_1,\ action_2)$, which evaluate to true, when $action_1$ appears before $action_2$ in the business process model or after respectively. These predicates can be specified on the semantic information provided by the OWL serialization of the process model ontology and allow for trivial checking of the constraint rules via an additional SWRL predicate $arcPossible(action,\ net)$, that is satisfied if the selected action is a transition and the first element in the process fragment under consideration is a place or vice versa.

To infer Event-Condition-Action style rules we compare the OWL version of the currently modeled process with given serializations of business process fragments in a repository. This is best illustrated with an example: given the business rule “IF request is checked THEN forward order” we would check the following conditions with the help of the beforementioned 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.

3 Conclusion

Often business processes are modeled according to specific business rules. In this paper we have described our idea of realizing an autocompletion of business process models. The main elements are that we use semantic business process models based on an OWL representation of Petri nets that allows us to efficiently compute the semantic similarity between process model variants. Additionally we use the Semantic Web Rule Language, which is based upon a combination of OWL DL with Unary/Binary Datalog RuleML [8], to model additional constraints imposed by business rules. Although the implementation of the recommendation system is not finished yet a study which has been conducted seems to confirm our approach.

References

1. Koschmider, A., Oberweis, A.: Modeling semantic business process models. In Rittgen, P., ed.: Handbook of Ontologies for Business Interaction. Idea group publishing edn. (2007) (to appear).
2. 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)
3. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C) (2003) Internet: http://www.w3.org/TR/owl-features/.
4. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: Proc. of the Fourth Asia-Pacific Conference on Conceptual Modelling, Australia (2007) (to appear).
5. Gatziu, S., Dittrich, K.R.: Events in an Active Object-oriented Database System. In: Rules in Database Systems. (1993) 23–39
6. Clocksin, W., Mellish, C.: Programming in Prolog. 3 edn. Springer (1987)
7. Friedman-Hill, E.: Jess: the Java Expert System Shell, sandia national laboratories. http://herzberg.ca.sandia.gov/jess/ (2001)
8. Boley, H., Tabet, S., Wagner, G.: Design Rationale for RuleML: A Markup Language for Semantic Web Rules. In: SWWS. (2001) 381–401