How to ensure correct process models? A semantic approach to deal with resource problems

Michael Fellmann[1], Frank Hogrebe[2], Oliver Thomas[1], Markus Nüttgens[2]

[1]Universität Osnabrück,
Institut für Informationsmanagement und Unternehmensführung,
Fachgebiet Informationsmanagement und Wirtschaftsinformatik,
Katharinenstraße 3, 49069 Osnabrück
{michael.fellmann|oliver.thomas}@uni-osnabrueck.de

[2]Universität Hamburg,
Fakultät Wirtschafts- und Sozialwissenschaften,
Lehrstuhl für Wirtschaftsinformatik, Von-Melle-Park 5, 20146 Hamburg
{frank.hogrebe|markus.nuettgens}@wiso.uni-hamburg.de

Abstract: Models are important to manage complexity. They provide a means for understanding processes, and understanding already is a benefit. Process models support the optimization, reengineering, and implementation of supporting IT systems. In this context, the correctness of process models is significant for both, research and practice. The paper presents an ontology-driven approach that aims at supporting semantic verification of semi-formal process models. We apply our approach using real-life administrative process models taken from a capital city.

1 Introduction

A major problem regarding resource verification is how to automate it. Model creators and readers do not necessarily share the same understanding as the concepts they use are usually not documented and mix both discipline-specific terminology and informal, ordinary language. Therefore, it is hard for humans to judge if a model is semantically correct and almost impossible for machines (apart from using heuristics) because the model element labels are not backed with machine processable semantics. The result of that is that the machine cannot interpret the contents of model elements. Our solution approach is to encode the model element semantics in a precise, machine readable form using ontologies. Based on this, we use rules to encode constraints used to verify aspects of process correctness regarding resource problems.

The paper is organized as follows. In section 2, we provide an overview of approaches in the state-of-the-art of model verification. In section 3, we present a case study that motivates our approach. We illustrate our approach by presenting rules to tackle real-life resource problems in process models in section 4. In section 5, we describe the limitations of our approach and look at future research.

2 State-of-the-Art

The analysis of the correctness of models can fundamentally be divided into verification and validation. In literature, verification is often mentioned if the criterion is the internal, syntactic and semantic constitution of a model. In contrast to that, validation means the eligibility of a model in respect to its intended use [De02, p. 24] or in respect to the correctness of the representation of the underlying object – in other words: if the criteria is something outside the model [ChBr08], [Me09, p. 2]. Therefore, while verification is potentially highly amenable to automation, validation is bound to human judgment and expertise. In contrast to validation, much research has been done regarding verification. In the area of process modeling, formal criteria such as „soundness", „relaxed soundness" or „well-structuredness" have been developed which are used to detect shortcomings such as deadlocks, missing synchronisations and other defects regarding the formal semantics [Me09]. There are some tools supporting these verifications such as the bflow* toolbox (www.bflow.org) [GrLa09] or the EPC Tools (wwwcs.uni-paderborn.de/cs/kindler/research/EPCTools).

However, although these criteria clearly go beyond merely checking the conformance of a model to its meta model or grammar of the modeling language, the semantics of individual model elements typically expressed using natural language labels is still rarely considered (see e.g. [ThFe09] for first ideas on that topic). A major problem regarding such semantic verification approaches concerns rule dynamics, as semantic verification rules do not target the (stable) modeling language but rather the model contents and thus are influenced by constantly changing legal and economic circumstances. Some efforts addressing the problem area of rule dynamics suggest graphical modeling languages such as BPSL (Business Property Specification Language) [LMX07] or suggest to capture the required rules implicitly by providing negative examples [SiMe06] or by patterns [SPH04]. We extend the state-of-the-art by showing that ontology-based representations of process models enable the formulation of more abstract and hence stable verification rules which are then applied to concrete process models using an inference engine in order to automate semantic verification. We apply our approach to real-world problems and therefore demonstrate that semantic verification is not only feasible, but also proves to be useful for solving real-world problems.

3 Case study

The municipality we chose for our case is one of the biggest cities in our country (region capital city). It has about 580,000 inhabitants and the public administrative authorities are employing about 9,100 employees, distributed over about 440 administration buildings. The structure is decentralized and subdivided into seven departments, each with 48 assigned offices and institutes. Based on a Fat Client Server architecture, the 6,000 IT-jobs are workplace-based and completely linked with each other via a communication system throughout the city. In view of the increasing international competition, the city is requested to rearrange its product and process organization, particularly, as the support of enterprise-related activities increasingly becomes a competitive factor. In the city,

about 99% of the enterprises have less than 500 employees and can be considered as small or medium-sized enterprises. These are about 40,000 enterprises. The strategic objective of the city is to make the place even more attractive for enterprises in terms of their competitiveness with a long-lasting effect. This shall be achieved by making the enterprise-related offers and services of the city even easier for enterprises to access, in terms of a One-Stop eGovernment. To reach this goal, the city has to model about 550 enterprise-related administrative processes. The process setting is highly relevant for the capital city, because several of the procedures are used about 15,000 to 25,000 times per year by the companies. After having started the project, we detected several inconsistencies in the collected data. Subsequently, we describe the modeling problems that we encountered regarding resource usage problems. The two core modeling errors (E1, E2) in this area were:

- (E1) Usually, process activities are executed by certain organization divisions. For example, the function "check the application of business registration" can only be executed by the civil servants of the business registration office. But in 44 of the relevant process models was a wrong department modeled or the organization unit was missed completely.

So, there is lack of resource usage rules like: *If a process uses an activity X, the process must (must not) use the resource Y.*

- (E2) Companies often combine several application cases. For example, in 24% of the cases the companies combine both, the application of business registration and the application of business building permission. In these cases, two different organization units are responsible, the business registration office and the building authority. But in 13% of the cases one of the responsible organization units was missed.

So, there is lack of resource occurrence rules like: *If a process demands a resource X, then it must also contain/involve that resource X.*

4 Ontology-driven approach for semantic verification

4.1 Classification of semantic verification rules

In general, rules may be divided into deductive and normative rules based on Boley et al. [BKP07, p. 273]. Deductive rules are used to win new facts on the basis of existing facts through the use of logical implications. Normative rules are used to express conditions for the data used for an application or the logic used by it. This understanding implies that the ontology is either entirely true or contains incorrect facts. As we also want to express constraints which – when violated – result merely in warnings and thus leave it up to human judgment to decide whether a model construct is correct or not, we do not call our rules "integrity rules". Instead, we prefer the term "verification rules", and as our rules are specified using concepts of a formal ontology we call them "semantic verification rules". The rule matter specifies the subject of a rule which is either the process, i.e. the set of nodes and arcs which constitute the core process graph, or the resources which are involved in the process. In the remainder of this paper, we focus on the latter aspect. In addition, the rule focus is either the structure of a process graph involving several resource-nodes connected by edges or the occurrence of specific resource nodes

anywhere in the process graph. According to this distinction, we differentiate between *resource usage rules* and *resource occurrence rules*. The result of the execution of a semantic verification rule may be a warning or an error.

4.2 Application to the case problems

In this section, we provide practical examples for the semantic verification rule types introduced in the previous section illustrating how our approach of semantic verification can be applied to the case problems given in section 3. As a prerequisite, the process model has to be represented in the ontology and annotated with ontology instances using the `p:equivalentTo`-property (see Fig. 1, due to space limitations, we only show some annotations). We use the prefix `p` for more general ontology contents and the prefix `ex` for contents related to concrete examples. On top of this ontology-based representation, we apply our semantic verification rules. The ontology language OWL, used in our approach, only supports the formulation of rules via extensions. Such an extension is the Semantic Web Rule Language (SWRL) [HPB04] which extends OWL with IF-THEN-rules in the form of a logical implication. The rules presented in the examples are of this nature and can be formalized using SWRL. The rules have the general form of

antecedent → consequent.

If the antecedent (body) of the rule is true, then the consequent (head) must also be true. Since the consequent consists of error messages, it will not be true in a literal sense, it rather will be generated if the antecedent matches and the rule is fired. In the following, we elaborate on some of the abstractions and inferences possible by using terminological and domain knowledge. They are an important merit of our approach as they provide for the formulation of rather generic semantic verification rules applicable to concrete models by automated machine reasoning:

- *Resource usage rule:* The rule in the given example (Fig. 1) fires if any activity node assigned to an organizational node being an individual of `p:ConsultingUnit` produces a legal document as output. The example makes use of subsumption reasoning so it can be inferred that `ex:trade_licence` of type `p:LicenceDocument` is-a `p:LegalDocument`.
- *Resource occurrence rule*: The rule makes use of a property `p:contains` being the inverse of `p:occursIn`, so that it can be concluded that `ex:process` contains the application of business building `ex:app_bus_building`. Based on this, it can be inferred that this process belongs to the class `p:ProcessWithReqBuildingAuthority` which is defined precisely as all processes containing an `ex:app_bus_building`. As `p:ProcessWithReqBuildingAuthority` is subsumed by `p:ProcessWithRequirement`, the semantic verification rule can operate on this abstract level using the latter class. Requirements are specified on the respective subclasses of `p:Process WithRequirement` using the *hasValue*-restriction of OWL which allows specifying a value of the requirement. That is, an instance must be present in the process (i.e. that should be annotated to at least one of the process nodes). The rule checks if there is not a single node in the process graph being an-

notated with that instance by using the noValue-extension of the Jena rule engine (jena.sourceforge.net).

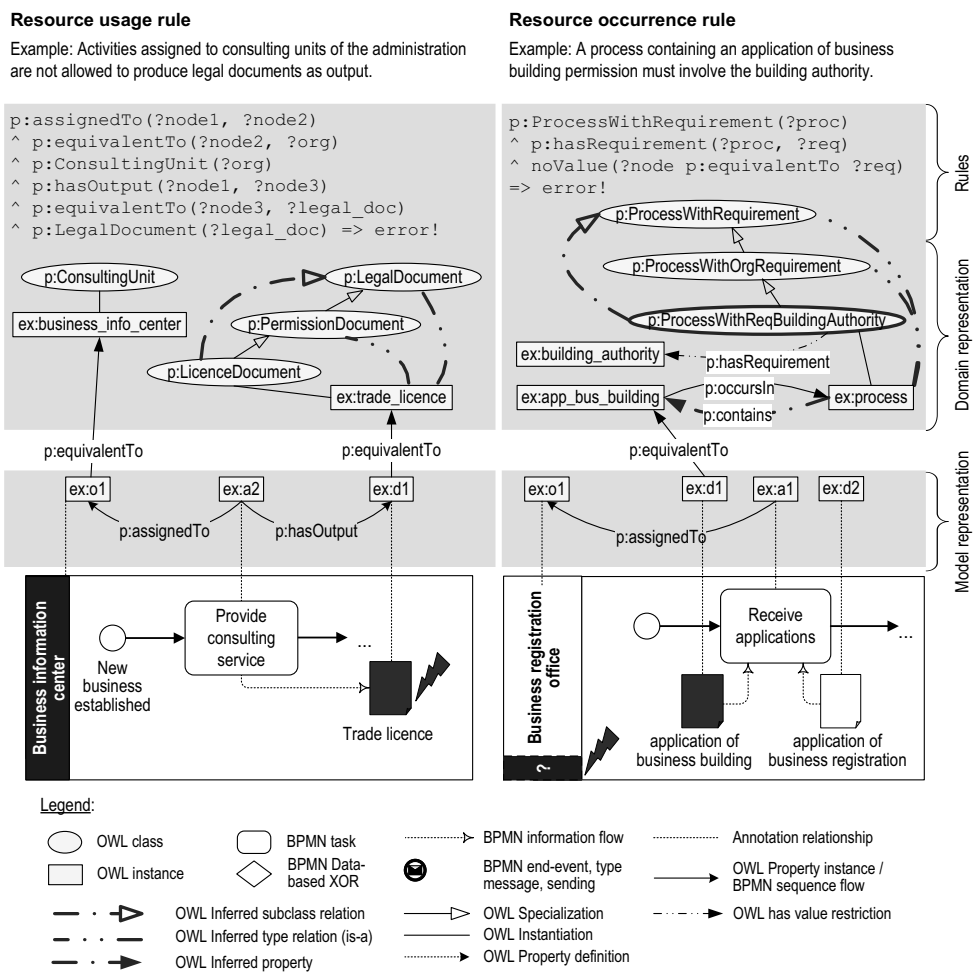

Fig. 1: Resource usage and resource occurrence rule

In general, SWRL and OWL work according to the so called “open world assumption” which is based on the assumption, that facts not present in the knowledge base are unknown or undefined. Therefore, only rules conforming to the scheme x ^ y → error are possible. By using the Jena rule engine, we can extend the range of possible rules to include also rules of the form x ^ ¬y → error. That is, if some facts x are known and some other facts y are not present in the knowledge base, the failure to derive them is treated as a form of negation (Negation as Failure, NAF). With NAF it is possible to specify rules which fire if something is missing in the process model.

5 Conclusion and further Research

The approach presented in this paper showed how to use ontologies, rules and reasoning for the semantic verification of process models. Future versions of our approach will tackle the limitation that control flow is currently not considered. As a next step, we plan to integrate a further pre-processing step which will mark the nodes in the graph according to their succession of logical connectors such as AND, XOR and OR. The capturing of information on such local contexts of parallelism or exclusivities to the ontology based representation of process models will allow advanced semantic verification rules such as “resource x must not be used in parallel branches”.

References

[BKP07] Boley, H., Kifer, M., Patranjan, P. L., Polleres, A. (2007): Rule Interchange on the Web. In: Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.-L., Tolksdorf, R. (2007) (Eds.): Reasoning Web : Third International Summer School 2007, September 3-7, Dresden, Germany. Berlin, Springer, pp. 269–309

[ChBr08] Chapurlat, V., Braesch, C. (2008): Verification, validation, qualification and certification of enterprise models: Statements and opportunities. In: Computers in Industry 59, No. 7, pp. 711–721

[De02] Desel, J. (2002): Model Validation – A Theoretical Issue? In: Esparza, J., Lakos, C. (Eds.): Proceedings of the 23rd Internat. Conference Application and Theory of Petri Nets (ICATPN 2002), June 24–30, Adelaide, Australia. Berlin, Springer, pp. 23–43

[GrLa09] Gruhn, V., Laue, R. (2009): Ein einfaches Verfahren zur Erkennung häufiger Fehler in EPKs (in german). In: Nüttgens, M., Rump, F. J., Mendling, J., Gehrke, N. (Eds.): 8. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)", November 26.–27., Berlin, p. 74

[HPB04] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL (2004): A Semantic Web Rule Language: Combining OWL and RuleML, W3C Member Submission 21 May 2004. www.w3.org/Submission/SWRL. Accessed at 04/2010

[LMX07] Liu, Y., Müller, S., Xu, K. (2007): A static compliance-checking framework for business process models. In: IBM Systems Journal 46, No. 2, pp. 335–361

[Me09] Mendling, J. (2009): Empirical Studies in Process Model Verification. In: Jensen, K., van der Aalst, W. M. P. (Eds.): Transactions on Petri Nets and Other Models of Concurrency II. Berlin, Springer (LNCS 5460), pp. 208–224

[SPH04] Speck, A., Pulvermuller, E., Heuzeroth, D. (2004): Validation of business process models. In: Proceedings of the 17th European Conference on Object-oriented Programming (ECOOP), July 21–25, Darmstadt, Germany. Berlin, Springer (LNCS 3013)

[SiMe06] Simon, C., Mendling, J. (2006): Verification of Forbidden Behavior in EPCs. In: Mayr, H. C., Breu, R. (Eds.): Proceedings of the GI Conference Modellierung (MOD2006), March, 22–24, Innsbruck, Austria, pp. 233–242

[ThFe09] Thomas, O., Fellmann, M. (2009): Semantic Process Modeling – Design and Implementation of an Ontology-Based Representation of Business Processes. In: Business & Information Systems Engineering 1, No. 6, pp. 438–451