An ontology-driven approach to support semantic verification in business process modeling

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: This paper presents an ontology-driven approach that aims at supporting semantic verification of semi-formal process models. Despite the widespread use of these models in research and practice, the verification of process model information is still a challenging issue. We suggest an ontology-driven approach making use of background knowledge encoded in formal ontologies and rules. In the first step, we develop a model for ontology-based representation of process models. In the second step, we use this model in conjunction with rules and machine reasoning for process model verification. We apply our approach using real-life administrative process models taken from a capital city.

1 Introduction

1.1 Motivation

Models are important to manage complexity. They provide a means for understanding the business process, and understanding already is a benefit. This is indicated by a study from Gartner revealing an increase in efficiency of 12 percent gained solely by documenting actions and organizational responsibilities in process models [Me05, p. 4]. Moreover, process models serve for optimization, reengineering and implementation of supporting IT systems. Due to the importance of process models, model quality and correctness is important. According to ISO 8402, quality is "the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs". Facets of quality are – amongst others – appropriateness in respect to the abstraction level of the representation (scale), detail of representation (granularity), compliance (conformance to rules

and regulations), adequate coverage of the model object, usefulness and correctness. We concentrate on correctness as the most fundamental quality aspect. Among the aspects of correctness are fundamentally: (a) syntax and formal semantics (structure and grammar), (b) linguistic aspects (labels) and (c) semantics (content). There is much research on (a) e.g. to detect deadlocks in works such as [MeAa07]. Aspects of (b) are increasingly focused in the scientific community to ensure compliance to naming conventions, see e.g. [PeWe09], but (c) has been neglected so far: semantic correctness means that the facts captured in the model about an object are (assumed to be) true. We call the latter aspect "semantic verification". A major problem regarding semantic verification is how to automate it. This problem is rooted in natural language being used for labeling model elements, thus introducing terminological problems such as ambiguity (homonyms, synonyms) and other linguistic phenomena. 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 is that the machine cannot interpret the contents of model elements, i.e. what is "inside the box" (rectangle, shape). Our solution approach is to encode the model element semantics in a precise, machine readable form using ontologies. Further, we then use rules to encode constraints used for verifying aspects of semantic correctness.

1.2 Prospects of Semantic Verification

The proposed approach of semantic verification allows performing additional checks on process models. Such checks are possible by annotating process models with instances of a formal ontology containing terminological knowledge of the domain under consideration. The ontology in conjunction with an inference engine can then be used to automatically verify several aspects of models based on the semantics of the individual model elements. This decoupling from human labor makes semantic verification scalable even in incremental approaches to model construction where a model has to be re-verified repeatedly. An important additional benefit thereby is that the semantic verification rules can be formalized on a more abstract and generic level and the inference engine interprets them with the help of both explicitly encoded and inferred knowledge from the ontology. Therefore, it is possible to formulate semantic verification rules in a more natural and understandable way that accommodates to the nature of generic rules such as guidelines, best practices, conventions, recommendations or laws being rather abstract in order to ensure broad applicability.

The paper is organized as follows. In section 2, we provide an overview of tools and approaches in the state-of-the-art of model validation and verification. In section 3, we present a case study that motivates our approach. We present our approach of semantic verification, a rule classification and examples illustrating the application of such rules to the real-world problems of the case study in section 4. In section 5, we describe the limitations of semantic verification and in section 6, we look at future research.

2 State-of-the-Art

The verification of models has focused mainly the syntax and formal semantics so far. In this sense, verification abstracts from the individual semantics of model elements which is given by natural language and concentrates on formal procedures. Such procedures partly originate from software engineering [Gr91] where they are discussed under the terms “model checking” and “theorem proving” [ChBr08]. These approaches concern dynamic aspects of model execution which are verified using finite state automata (FSM). In the area of process modeling, independent formal criteria have been developed such as „soundness“, „relaxed soundness“ or „well-structuredness“ which are used to detect shortcomings such as deadlocks, missing synchronizations and other defects regarding the formal semantics [Me09]. These criteria clearly go beyond merely checking the conformance of a model to its Meta model or grammar of the modeling language. There are some tools supporting these verifications such as the bflow* toolbox (www.bflow.org) or the EPC-Tools (wwwcs.uni-paderborn.de/cs/kindler/research/EPC Tools). Research concerning formal verification is still an active field; new approaches consider e.g. the verification of access constraints in semi formal models [WMM09], verification in the context of hierarchical models [SCA07] or workflows [TBB08].

However, a major problem still is that verification rules are exposed to frequent changes due to the dynamics of the contemporary legal and economic world. Some efforts address this problem area of rule dynamics and suggest graphical modeling languages such as BPSL (Business Property Specification Language) [LMX07] or suggest capturing the required rules implicitly by providing negative examples [SiMe06] or by patterns [SPH04]. Nonetheless, a fundamental problem is still, that most approaches require a rather fine grained specification of rules conflicting with the rather abstract nature of rules required for semantic verification in the sense of guidelines, best practices or general principles. First approaches in this direction therefore explore the use of rules together with semantic process descriptions [ThFe09] and describe frameworks for semantic verification related to compliance [ElSt08]. These approaches therefore rely on more formally defined semantics in comparison to e.g. glossaries or technical term models [KuRo98].

We extend the state-of-the-art by showing that ontology-based representations of process models enable the formulation of generic 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-life problems.

3 Case study

The municipality we took as our case is one of the biggest cities in the country we accomplished our research (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 Fat Client Server architecture, the 6,000 IT-jobs are workplace-based and completely linked to 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 becomes increasingly an essential position factor in the international competition. 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 goal 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 to access for enterprises, 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 lay open.

3.1 Terminological problems

- (T1) Due to the fact that laws and regulations are regularly made by jurists and not by IT-experts, terms and facts of cases are often differently named although the meaning of two terms is the same. For example, the terms “admission” and “permission” were found in 334 administrative process models, but the terms always had the same meaning and the same process-related consequence.
- (T2) Another terminology problem occurs concerning the fact that in the municipality we have examined no rules were arranged to allow only one preferred term for one correspondent meaning. For example, some modelers (14) used “address” and some (8) “mailing address”, or modeler used abbreviations, like “doc” instead of “document”.

So, there is a lack of terminological modeling rules. These terminology problems hindered the identification, comparison and further use of the administrative process models (e.g. in process automation) in the city we focused on.

3.2 Verification problems

The administrative process models had also several errors regarding the correct sequence processing. Subsequently, we show the core modeling errors of process sequence conflicts (V1-V4) and in process sequence conformance (V5):

- (V1) In 64 process models, the event "admission free of charge" was followed by the (wrong!) function "start payment process".
- (V2) As part of a preliminary check, which is executed in every application process at the beginning, the civil servants check the completeness of the submitted documents. In 41 of these process models, we found after the event "documents uncompleted" the (wrong!) event "preliminary check complete", although documents were still missing.
- (V3) In 32 process models we found after the event "application is not licensable" the (wrong!) function "send admission".
- (V4) The next step after the preliminary check is an in-depth check of the admission case. This row is strictly followed. But in 13 of the administrative process models, we found the two checks reversed.

So, there is a lack of element flow rules like: *After X must (must not) follow Y.*

- (V5) If a process contains the event "procedure is billable", the same process must also contain a function "calculate charge". But in 21 of the relevant process models, no such function was found.

So, there is a lack of element occurrence rules like: *If a process contains X, the process must (must not) contain Y.*

4 Ontology-driven approach for semantic verification

4.1 Ontology-based representation of process models

A first step towards semantic verification of semiformal process models is the representation of the process models using a formal ontology language such as the Web Ontology Language (OWL) standardized by the World Wide Web Consortium (w3.org/2004/OWL). We use this ontology language, as it has gained a broad acceptance both inside and outside the AI and Semantic Web community. The use of the ontology-based representation is twofold. On the one hand, it allows the connection of process models with domain knowledge in order to improve the interpretation and derive new facts not explicitly specified by the modeler but relevant for verification. On the other hand, it provides for a machine processable representation enabling the automation of such derivations and therefore using logic and reasoning to automate verification tasks. The ontology-based representation of process models consists of creating a model representation in the ontology (step 1) and the annotation of domain knowledge to that representation (step 2) (cf. Fig. 1) which are described subsequently.

The creation of a process model representation in the ontology is done by considering its graph structure. For each node, an instance is created in the ontology and for each arc, a property is created in the ontology connecting the two nodes which are at the end of the arc. This step can be executed automatically using the capabilities of a transformation language such as XSLT. The instances created in the ontology are instances of the classes shown in the left part of Fig. 1 which reflects the well-known Workflow Patterns. The properties having their domain and range on the `p:ProcessGraphNodeClass` are used to represent direct connections between model elements (property `p:connectsTo` being a sub-property of `p:flow`) as well as the set of following elements which can be reached without traversing an exclusive decision point such as an XOR-Gate (transitive property `p:followedBy`) or which can be reached by an arbitrary path along the flow in the process model (transitive property `p:flow`). We use the namespace-prefix `p:` for indicating the process space in general and `ex:` for indicating example data that is strongly intertwined with the concrete process fragment being used for illustrative purposes. Due to space limitations, we have omitted the translation of BPMN-lanes into organizational units in the ontology which can be represented by properties `p:assignedTo` which are added to each node in a lane. Currently, we also omit pools for the sake of simplicity.

The annotation of the process model representation with domain knowledge via the `p:equivalentTo`-properties shown in the right part of Fig. 1 provides for the semantic specification of the model elements with machine processable semantics. Domain ontologies can be built by leveraging existing ontologies (cf. section on the state-of-the-art), using reference models, ontologizing industrial standards or extracting structures out of IT-systems such as database schemas. Also, top-level or upper ontologies may be used as a basic backbone structure that helps bootstrap ontology development and reaching ontological commitment on how to think about the world in the sense of a shared "contract" between the different involved stakeholders. In the example of Fig. 1, we have used the SUMO-ontology as a backbone structure providing basic distinctions such as between abstract and physical entities forming the basis of the subsumption hierarchy. This hierarchy not only serves for disambiguation purposes (e.g. `Service` as subclass of `ComputerProcess` vs. subclass of `Product`). It also provides for the specification of semantic verification rules on varying levels of generality. This enables the specification of rather generic verification rules such as guidelines and best practices and letting an inference engine do the work of verifying whether a specific model is compliant or not. So, for example, a government agency could have the guideline that immediate feedback should be given on each application. If a process model starts with a citizen having filed her tax return and contains an activity "send feedback via e-mail", then the inference machine can prove that this process complies with the guideline as "tax return" is subsumed by "application" and "feedback via e-mail" is subsumed by "feedback".

Beyond such simple subsumption reasoning, an inference engine can also be used to automatically derive more complex conclusions. Automatic classification of instances for example could be achieved by using class expressions composed of intersection, union and complement which are available in OWL and which rely on propositional logic. Also, automatic classification can leverage existential restrictions of properties on classes as well as restrictions on their domain and ranges thus relying on a fragment of

first order logic. Moreover, OWL and most of the current ontology languages also provide for specific characteristics of properties such as symmetry, transitivity, reflexivity etc. leading to additional conclusions in regard to the structure of a process graph represented in the ontology. While we use ontology for both, representing a process graph and inferring new facts about it, we use rules to express constraints for verification. Before we show the application of such rules to solve the case problems described in section 3, we will introduce them in the next section.

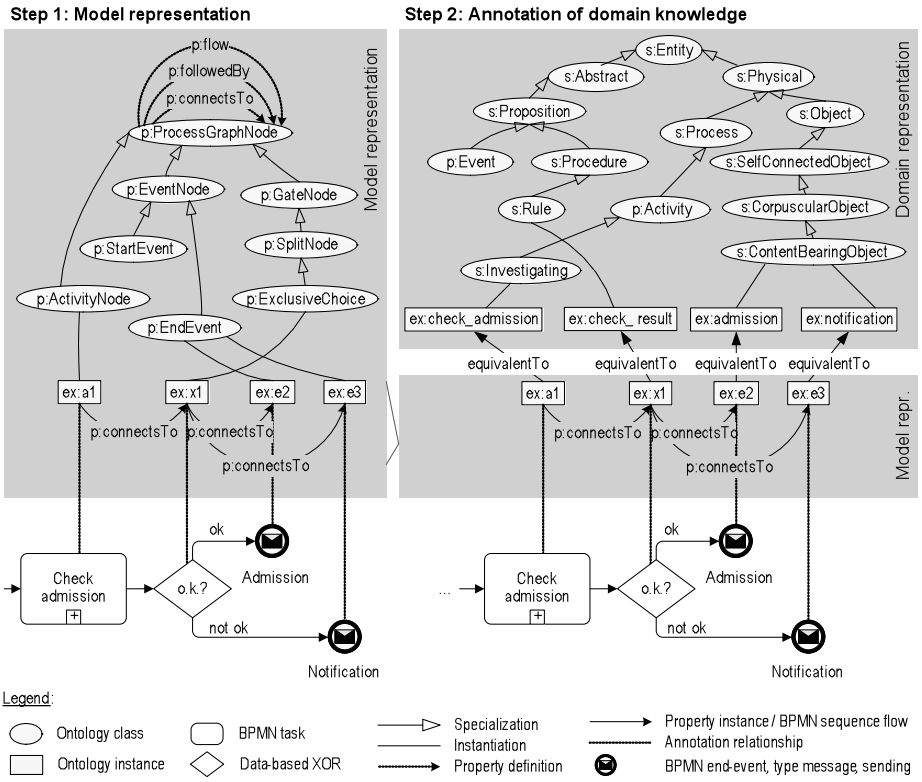

Fig. 1: Ontology-based representation of process models

4.2 Semantic verification rules

According to the IEEE 1012-1998 definition [IEE98] “verification” means to check whether an artifact and/or its creation comply to a set of given requirements hence focusing on artifact-internal aspects. This understanding of verification is in contrast to validation which means ensuring that an artifact is eligible for the intended purpose [De02] thus focusing on artifact-external aspects and human judgment and experience. Intuitively, verification of process models is more amenable to machine processing than validation, given the fact that the model elements are annotated with machine processable

semantics and the verification rules are formalized. Due to the fact that our approach suggests also using terminological and domain knowledge (encoded in ontologies) for verification, we call our verification rules "semantic verification". If the focus of a rule is the structure of a process (e.g. the sequence of actions), then we call such a rule *element flow rule*. If the focus of the rule is the occurrence of model elements in arbitrary positions in the model, then we call such a rule *element occurrence rule*. These basic rule types may be mixed in practical applications, such that any combination of two types may be combined to a single rule. For example, if an organizational unit "government representative" is present anywhere in the process (element occurrence rule), then an additional sequence of activities such as "report results to head of administration" has to be performed (element flow rule) involving at least one information system for archiving the results.

4.3 Application to the case problems

In this section, we provide practical examples for each of the basic 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. Fig. 2 illustrates the application of an element flow rule (on the left side) and an element occurrence rule (on the right side). At the bottom layer, fragments of a process described by using BPMN are displayed. Model elements targeted by the verification rules are highlighted (dark-red filling with white labels). Above the model layer, the ontology is displayed consisting of a model representation part and a domain representation part. The semantic verification rules using the classes and instances of the ontology are displayed above the ontology. The rules are displayed in an informal notation with variables prefixed by question marks, class memberships written as functions with one argument and predicates (properties in the OWL-terminology and edges in the graph-terminology) as functions with two arguments. To improve comprehensibility, the rules have additionally been paraphrased using natural language at the topmost layer.

Regarding rules, there are a number of non-web-based ontology languages, such as OCML and Ontolingua, which make it possible to formulate rules without an extension. The ontology language OWL, used in this article, only supports the formulation of rules via extensions (apart from simple property chains in OWL 2.0). 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. They have the general form of antecedent → consequent – i.e. 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 executed (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:

- *Element flow rule*: Terminological knowledge is used in stating that `ex:check_permission` is the same as `ex:check_admission`. Hence, using this terminological knowledge, the `p:equivalentTo` property between `ex:a1` and `ex:check_admission` can be inferred. Moreover, as `p:followedBy` is a transitive property, the triple `ex:a1 p:followedBy ex:a3` can be inferred. As `ex:a3` is annotated with an ontology instance that belongs to the class of `p:PreChecking` activities, the antecedent of the rule is satisfied and the rule fires.
- *Element occurrence rule*: The example makes use of a class definition by enumeration resulting in `ex:receipt_child_benefit_app` being classified as an individual of `p:UnbillableProcStartEvent`. The rule fires because there is another node in the process that is annotated with an individual belonging to the class `p:FeeCalculation`. Obviously, the rule is specified rather generic and will fire if two nodes are annotated with instances classified as members of the two classes `p:UnbillableProcStartEvent` and `p:FeeCalculation`.

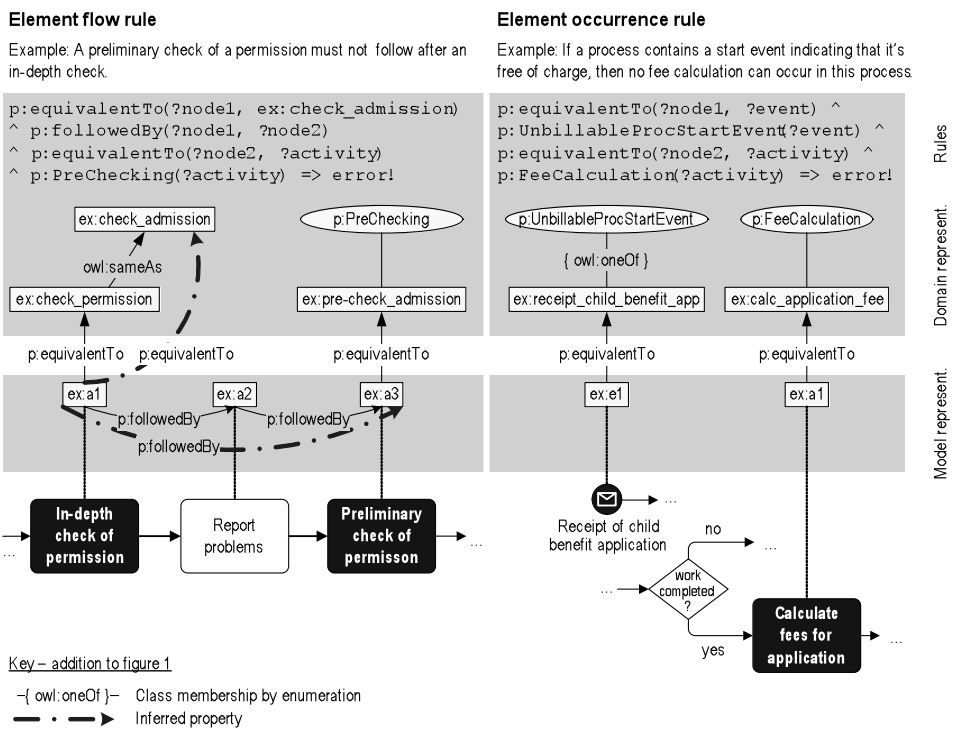

Fig. 2: Element flow rule and element occurrence rule

The examples presented to exemplify the rule types have in common, that they use facts that are explicitly known (either declared or inferred). The general pattern of this is `a ^ b -> error`. However, both rules can also be modified to the form of `a ^ ¬b -> error`, i.e.

if some facts a (fragments of a process graph) are known, some other facts b (again fragments of a process graph) should not be present in the knowledge base and the failure to derive them should be treated as a form of (weak) negation. This implies closed world reasoning (as opposed to open world reasoning) and negation as failure (NAF). Ontologies in the Sematic Web adhere to the open world assumption (OWA), which makes sense in an open and networked environment such as the web.

According to the OWA, facts that are not explicitly stated in the knowledge base are not false but instead unknown or undefined. In contrast to that, to verify process models it would be useful to at least temporarily assume to know all the facts and hence switch to closed world reasoning. This sort of reasoning requires negation as failure (NAF) and can be introduced using the Jena built-in rule engine ARQ (jena.sourceforge.net/ARQ/) which provides this feature using procedural built-in primitives which can be invoked by the rules. Each primitive is implemented by a Java object and additional primitives can be created and registered by the user. To achieve closed world reasoning using NAF, the primitive `noValue(?subject, ?predicate, ?object)` can be embedded in a rule which will cause a rule to fire if no matching triple can be found. With closed world reasoning, semantic verification rules such as the following examples would be possible:

- *Element flow rule*: If there is a preliminary check, the in-depth check always has to be performed afterwards.
- *Element occurrence rul*e: If the process starts with an event indicating that this process is billable, then somewhere in the process there must be an activity "calculate fee".

Furthermore, tools such as Jena or the SQWRL query language implemented in the Protégé-editor also provide built-ins for counting, geo-related reasoning and many other possibilities which enhance the power of semantic verification rules.

4 Limitations of semantic verification

Clearly, semantic verification rules have some limitations. To begin with, they should not be regarded as a surrogate for verifications related to the meta-model or the grammar of the used modeling language. They are rather complementary to such verifications and correct models form the basis for additional semantic verification checks. Also, aspects regarding the execution semantics of models such as soundness, relaxed soundness etc. dealing mainly with the absence of deadlocks and livelocks are not covered by our approach due to its complimentary nature.

Further limitations of semantic verification rules are that they depend on the availability of an ontology and the annotation of process models. While in other areas such as the life sciences huge ontologies have been developed and standardized, the field of administration still lacks authorities who develop and standardize ontologies. However, this problem may partly disappear if the terminology problem will be solved, e.g. by defining structured vocabularies which bootstrap the development of ontologies. Also, current tools for process model annotation are mostly in the state of research prototypes. In par-

ticular, functionalities for semi-automated annotations and annotation suggestions based e.g. on annotations previously made in the current model or the whole model repository etc. have to be developed in the future in order to enable comfortable and cost-effective semantic verification. Also, aspects of ontology management have to be considered since ontologies are not automatically a shared knowledge representation. A major limitation of the current approach is that it is agnostic to the control flow of process models. At the moment, the only exception of that is the property `p:followedBy` connecting only nodes which form a sequence when the model will be executed and so it provides for rules such as "b should not be executed after a".

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 some of the described limitations. 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" or "activity x and activity y should always be executed exclusively".

References

[ChBr08] Chapurlat, V., Braesch, C. (2008): Verification, validation, qualification and certification of enterprise models: Statements and opportunities. In: Computers in Industry 59, Nr. 7, pp. 711–721

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

[ElSt08] El Kharbili, M., Stein, S. (2008): Policy-Based Semantic Compliance Checking for Business Process Management. In: Loos, P., Nüttgens, M., Turowski, K., Werth, D. (Eds.): Modellierung betrieblicher Informationssysteme (MobIS 2008) - Modellierung zwischen SOA und Compliance Management. 420. RWTH Aachen (CEUR Workshop Proceedings), pp. 165–177

[Gr91] Gruhn, V. (1991): Validation and verification of software process models. In: Proceedings of the European Symposium on Software Development Environments and CASE Technology. Berlin, Springer

[HPB04] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M. (2004): SWRL: A Semantic Web Rule Language: Combining OWL and RuleML, W3C Member Submission 21 May 2004. W3C. http://www.w3.org/Submission/SWRL/. Accessed at 2010-05-01

[IEE98] IEEE 1012-1998. http://www.techstreet.com/standards/IEEE/1012_1998?prod-uct_id=31920. Accessed at 2010-05-01

[KuRo98] Kugeler, M.; Rosemann, M. (1998): Fachbegriffsmodellierung für betriebliche Informationssysteme und zur Unterstützung der Unternehmenskommunikation. In: Fachausschuss 5.2 der Gesellschaft für Informatik e. V. (GI) (Ed.): Informationssystem-Architekturen, 5, pp. 8–15

[LMX07] Liu, Y., Müller, S., Xu, K. (2007): A static compliance-checking framework for business process models. In: IBM Systems Journal 46, Nr. 2, pp. 335–361

[Me05] Melenovsky, M.J. (2005): Business Process Management's Success Hinges on Business-Led Initiatives. Stamford, CT: Gartner Research

[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. 5460. Berlin, Springer, pp. 208–224

[MeAa07] Mendling, J., van der Aalst, W.M.P. (2007): Formalization and Verification of EPCs with OR-Joins Based on State and Context. In: Krogstie, J., Opdahl, A.L., Sindre, G. (Eds.): Proc. of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), 11–15 June 2007, Trondheim, Norway. Berlin: Springer (LNCS), pp. 439–453

[PeWe09] Peters, N., Weidlich, M. (2009): Using Glossaries to Enhance the Label Quality in Business Process Models. 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)" Berlin, 26.11–27.11.2009, pp. 75–90

[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, 2006, Innsbruck, Austria, pp. 233–242

[SCA07] Salomie, I., Cioara, T., Anghel, I., Dinsoreanu, M., Salomie, T. I. (2007): A Layered Workflow Model Enhanced with Process Algebra Verification for Industrial Processes. In: Proceedings of the 2007 IEEE International Conference on Intelligent Computer Communication and Processing, September 6-8, Cluj-Napoca, Romania, pp. 185–191

[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 2003, Darmstadt, Germany. Berlin et al., Springer

[ThFe09] Thomas, O., Fellmann, M.: (2009) Semantische Prozessmodellierung – Konzeption und informationstechnische Unterstützung einer ontologiebasierten Repräsentation von Geschäftsprozessen. In: Wirtschaftsinformatik 51, Nr. 6, 2009, pp. 506–518

[TBB08] Touré, F., Baïna, K., Benali, K. (2008): An Efficient Algorithm for Workflow Graph Structural Verification. In: On the Move to Meaningful Internet Systems: OTM 2008: Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I. Berlin, Springer, pp. 392–408

[WeLa08] Wecker, G., van Laak, H. (2008) (Eds..): Compliance in der Unternehmenspraxis: Grundlagen, Organisation und Umsetzung. Wiesbaden : Gabler

[WMM09] Wolter, C., Miseldine, P., Meinel, C. (2009): Verification of Business Process Entailment Constraints Using SPIN. In: Proceedings of the 1st International Symposium on Engineering Secure Software and Systems. Berlin, Springer, pp. 1–15