Process Model Verification with SemQuu

Michael Fellmann, Oliver Thomas

Institut für Informationsmanagement und Unternehmensführung
Universität Osnabrück
Katharinenstr. 3
49074 Osnabrück, Germany
{michael.fellmann, oliver.thomas}@uni-osnabrueck.de

Abstract: In this contribution, we present an overview of our ongoing work in developing an integrated tool support for semantic process model verification. We present some of the requirements and insights gained by the implementation of a platform for semantic process model verification. The platform is based on OWL and SPARQL and provides for a user-friendly way of semantic model verification. First empirical evaluations have been very promising and confirm that the tool can be used to effectively query and verify semi-formal process knowledge.

1 Introduction

1.1 Background

A fundamental idea in the context of the model-driven development of information systems is to use models to manage the complexity accompanied with the development task. They provide a means for understanding business processes and serve for optimization, reengineering, and development of supporting IT systems. Especially in regard to the latter aspect, the quality of the models is of vital importance since it directly affects the quality of the information systems developed on the basis of the models. According to ISO 8402, quality is “the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs”. One fundamental aspect of model quality is the correctness of a model. There has been extensive research regarding syntactical correctness. In contrast to that, approaches to verify the semantic correctness, i.e. the correctness of the model contents, have been considered less frequently. With the term “verification”, we denote ensuring correctness with respect to explicitly stated requirements. In contrast to that, validation means the eligibility of a model in respect to its intended use [De02, p.24] – in other words: if the criteria is something outside the model [Me09, p.2]. Following this distinction, we call the procedures to ensure semantic correctness “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. Hence, 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. Therefore, we have developed an approach to encode the model element semantics in a precise, machine readable form using OWL ontologies. On top of the ontology-based representation, SPARQL-queries are used to check the correctness of a model based on its graph structure. We use this query language as (a) it is a language which has been standardized by the World Wide Web Consortium (W3C) and gained broad acceptance in the Semantic Web and AI community, (b) it provides a simple structure for queries consisting of triples which intuitively reflect the structure of process graphs and (c) the query language is not specialized to any specific process modeling language such as EPC, BPMN or UML Activity Diagrams. The approach is described in more detail in [FeTh11]. In contrast to our previous work where we elaborated on our concept of semantic verification, in this contribution we concentrate on the description of our prototypical implementation. We describe the requirements, design decisions and insights gained by extending a modeling tool with support for semantic verification.

1.2 Related Work

Approaches to ontology-based semantic verification can be found e.g. in the context of Semantic Web Services. Semantically annotated process models are verified with an emphasis on logical preconditions and effects which are specified relative to an ontology [Dr07], [We09], [WHM10]. These approaches usually require both the annotation of preconditions and effects and hence enable to check if the model is consistent. They do not build upon a formal representation of the (intentional) semantics of individual model elements (i.e. what is “inside the box” of a model element). In contrast to that, the underlying paradigm of our approach is capturing the semantics of model elements by using a single annotation of a model element in order to associate it with its intended meaning explicitly specified in a formal ontology. Semantic constraints are then formulated as SPARQL queries which are executed against the model repository. Such queries focus constraints related to the structure of a model (e.g. to verify the absence of an anti-pattern) and are in this respect orthogonal to approaches emphasizing the correct execution of process instances [Ly09]. In respect to the IT-support of SPARQL queries, query builders such as Konduit VQB have been developed [AMH10]. Whereas these tools provide complex forms aiming at a general-purpose support for SPARQL, our approach is tailored to query process models. We provide a simple yet powerful user interface and we have evaluated the effectiveness in a user experiment.
2 Requirements

In this section, we describe basic requirements regarding the functionality of the integrated tool support for our approach to semantic model verification described in [FeTh11]. To explore the general topic of queries and search in process models, we conducted a survey. The participants of the survey (50 from both academia and practice) clearly favored form-based approaches over all other approaches for querying model contents. As form-based approaches require less knowledge in comparison to querying model contents. As form-based approaches require less knowledge in comparison to the usage of a query language, it can be concluded that the users want to have a simple and user-friendly way of specifying queries. However, by relying exclusively on form-based queries, much of the freedom and flexibility a query language such as SPARQL for querying graph-like structures offers is lost. We therefore suggest using a hybrid approach where triple patterns of SPARQL queries can be constructed using a form and adapted as plain-text manually. In order to execute multiple queries e.g. when checking if a set of constraints is met, the tool has to provide for the batch-execution of queries. Finally, we do not want to confine ourselves to a specific modeling language such as BPMN or EPC. Rather, the tool should transform different model representations into the ontology-based process representation devised in our previous work. Below, we summarize the requirements.

(R1) Form-based, fail-safe query construction: Since our survey revealed that form-based approaches of query construction are favored over other alternatives, the tool has to offer support for query construction using forms. The user should construct a query exclusively by interacting with the form, i.e. no textual input should be required. Only valid queries both in respect to syntax and semantics should be producible with the form-based query construction feature.

(R2) User-friendly query refinement: Since we propose the usage of SPARQL for querying process models and for specifying semantic constraints, the user should be able to use the full expressivity of this query language. However, the user should be supported in refining a query which in its initial form is generated using the form-based approach of query construction (cf. R1).

(R3) Execution of batch queries: In order to execute multiple queries at once, the tool has to provide for the batch-execution of queries. The results should be summarized in a report whereby it should be possible to assign a type to each query result as either being information, a warning or an error.

(R4) Support of multiple modeling languages: In order to achieve a broad applicability of the tool, it should not be dependent on a specific modeling language such as the EPC, BPMN etc. but instead should be generic.

Moreover, other requirements such as performance and scalability were considered as well during the development process. Due to space limitations, we have not listed them as requirements but mention them in the appropriate sections of the platform description.
3 Using SemQuu for Process Model Verification

In the following, we describe the tool we have developed for process model verification. The tool is called SemQuu – Semantic Query Utility, since much of the functionality is also relevant when exploring ontologies in general. SemQuu is implemented using the Jena library (jena.sourceforge.net), the Pellet inference engine (pellet.owldl.com), the Tomcat web server and JSP, XSLT, JavaScript, CSS and HTML for the user interface.

3.1 Import of Process Models

Process models can be imported from arbitrary tools in arbitrary formats, since they can be transformed on the server into the ontology-based representation which is accomplished by a plugin-converter for each format (R4 is met). As an example, we have implemented an extension of Visio which can export EPC process models being annotated with ontology concepts (see Fig. 1 for an illustration of the following procedure). For export from Visio, we use the standard EPML format (Event-driven Process Chain Markup Language) with special attributes capturing the semantic annotation of the model. After having been exported from Visio to SemQuu, the model is transformed to an OWL-DL ontology and added to the repository (cf. Fig 1 right).

![Fig. 1 SemQuu Converter and Repository](image)

3.2 Querying the Semantic Process Repository using SPARQL

An overview of SemQuu is provided by Fig. 2 (B). In order to query the repository of ontology-based process representations with SPARQL, the user can use a simple form-based query builder (A) for successively constructing a graph pattern (R1 is met). This is done by inserting multiple triple patterns with the help of drop-down list boxes (A) which are aware of rdfs:domain and rdfs:range information so that no semantically wrong query can be constructed (R1 is met).
Moreover, drop-down list boxes are dynamically updated upon selection of a value in one of the boxes. Alternatively to the drop-down list boxes, the user can leverage the full expressivity of SPARQL by using the text area input field. Moreover, when the user modifies the query she or he is supported by an “intelligent” auto-completion feature (D) which is fully aware of the ontology schema and instance data and only proposes meaningful suggestions (R2 met). When queries are executed in batch mode, the result of the queries can be displayed as an information, warning or error with respective graphical symbols appearing for each type (C). The result for each query is initially collapsed but can be unfolded if the user clicks on the “+” sign symbols (R3 is met).

4 Evaluation

In order to measure the effectiveness of SPARQL queries for the task of semantic correctness checking, we conducted an experiment with 21 participants. We have described this experiment in more detail in [FeTh11]. We give a short overview of the results of this experiment here. We used models in three different sizes with 11, 39 and 217 model elements. The participants had to answer 10 questions per model (a) by browsing and searching in the model with the tool Microsoft Visio or (b) by using SemQuu. We conducted the experiment without the form-based query construction feature since it was not been available at the time we conducted the experiment (however, we plan to re-evaluate the tool). Fig. 3 shows the time required for answering the 10 questions for each of the three models in relation to the model size. As can be seen easily, at first there is a learning curve when using SPARQL. When the model size exceeds approx. 190 model elements, the query language outperforms manual search.
5 Conclusion and Outlook

In this contribution, we have introduced SemQuu, our prototype for semantic process verification. Results of an initial evaluation regarding the query formulation have been very promising. The tool is under active development, currently we are working on improvements regarding the import and export of model data from Microsoft Visio.

Literature


