Preprint version of:
Hinkelmann, K., Laurenzi, E., Martin, A., & Thönssen, B. (2018). Ontology-based metamodeling. In R. Dornberger (Ed.), Business Information Systems and Technology 4.0. Springer International Publishing. https://doi.org/10.1007/978-3-319-74322-6_12

Ontology-Based Metamodeling

Knut Hinkelmann, Emanuele Laurenzi, Andreas Martin and Barbara Thönssen

Abstract Decision makers use models to understand and analyze a situation, to compare alternatives and to find solutions. Additionally, there are systems that support decision makers through data analysis, calculation or simulation. Typically, modeling languages for humans and machine are different from each other. While humans prefer graphical or textual models, machine-interpretable models have to be represented in a formal language. This chapter describes an approach to modeling that is both cognitively adequate for humans and processable by machines. In addition, the approach supports the creation and adaptation of domain-specific modeling languages. A metamodel which is represented as a formal ontology determines the semantics of the modeling language. To create a graphical modeling language, a graphical notation can be added for each class of the ontology. Every time a new modeling element is created during modeling, an instance for the corresponding class is created in the ontology. Thus, models for humans and machines are based on the same internal representation.

Keywords Modeling, ontologies, metamodel, enterprise modeling, domain-specific modeling language

1 Introduction

Decision makers use models to understand and analyze a situation, to compare alternatives and to find solutions. Business process models, for example, enable the identification of potential improvements and the communication of process variants with stakeholders. Enterprise models serve as a baseline for changing the enterprise. Engineers use models as blueprints for planning and construction.

Models describe and represent the relevant aspects of a domain in a defined language. There are many different kinds of modeling languages: graphical models, conceptual models, mathematical models, logical models. Even textual descriptions can serve as models. The choice of the modeling language depends on what the models is used for and who is using the model.

General-purpose modeling languages such as UML have the advantage that they can be used to represent any kind of information. However, they have the disad-

K. Hinkelmann (✉) • E. Laurenzi • A. Martin • B. Thönssen
FHNW University of Applied Sciences and Arts Northwestern Switzerland,
Riggenbachstrasse 16, 4600 Olten, Switzerland
e-mail: knut.hinkelmann@fhnw.ch

vantage that they do not guide people in modeling. People have difficulty conceptualizing the domain and different people might conceptualize the domain in different ways.

Domain-specific modeling languages, on the other hand, consist of modeling elements which have a pre-defined meaning that a domain expert can understand. Business process modeling languages, for example, are specialized for modeling the process flow using elements such as tasks and events and relationships to represent the order of the task execution. Domain-specific modeling languages reduce the degree of freedom for the modelers and thus support the understanding and reuse of models by different people.

Models are typically designed for a specific purpose. There are a huge variety of domain-specific modeling languages and modeling tools. Business Process Model and Notation (BPMN) has been designed to provide a standard visualization mechanism for business processes, which are defined in an execution optimized business process language (OMG 2011). BPMN engines allow the deployment and execution of business processes. Besides, process models can also serve the purposes of process optimization, governance, risk analysis and compliance management. Process models designed for execution, however, are often not compatible with models serving these purposes, although they have an overlapping set of modeling elements. Furthermore, there are typically specific tools for the various purposes, each with its own modeling language. Consequently, processes have to be re-modeled several times.

For a comprehensive view it would be beneficial to have modeling languages that serve several company-relevant issues such as decision-making, automation and compliance. Different applications can share or exchange models, or parts of them, and thus avoid re-modeling. As a prerequisite, the semantics of the modeling language has to be clearly defined.

Humans use graphical models for communication and to identify potential for improvement. Fig. 1 shows an example of a business process model in BPMN (OMG 2011). Humans can recognize that there is a deadlock for a known customer in the customer lane and they can propose parallelization to check the formal and financial consequences, because there is no data dependency between these tasks.

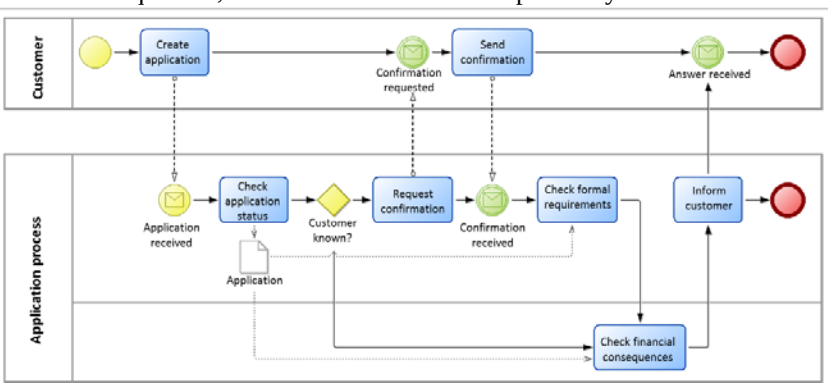

Fig. 1 Graphical business process model for human interpretation

In software engineering, graphical models of UML are widespread, e.g. using class diagrams for conceptual modeling. ArchiMate (The Open Group 2016) is a modeling language for enterprise architecture.

While graphical and textual representations are well understood by humans, they are not adequate for machine interpretation. Formal models are required to be interpreted by software systems must be formal models. Business process improvement can be supported in this with software tools, with which models can be checked for consistency, KPIs can be measured and simulations can be performed. A typical approach is to have separate models for humans and machines (see Fig. 2):

- Graphical notations which can easily be understood by humans are provided.
- Formal models such as databases, mathematical models and program code are used for machine interpretation.

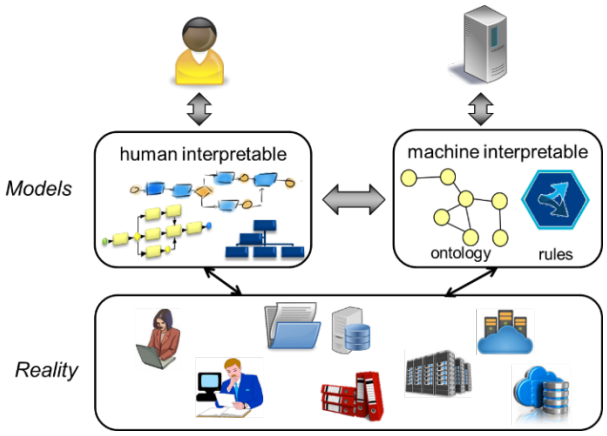

Fig. 2 Typical modeling approach: separate models for human and machine interpretation

The focus on machine-interpretable knowledge is known as knowledge engineering (KE) and is distinguished from knowledge management (KM), which focuses on human-interpretable knowledge (Karagiannis and Woitsch 2010). However, the models used by humans and machines are not strictly separated. Humans create the models and then use machines for analysis. The results are then again presented to humans for interpretation. However, if humans and machines use different models, it is hard to maintain consistency of the models. The use of both graphical and formal models has two challenges:

1. The semantics of the graphical and formal models must be identical. One way to achieve this is to define the semantics of both models in a formal language, which is known as semantic lifting (Azzini et al. 2013).
2. If part of the reality is represented in both graphical and formal models, a change in any of the models must be mirrored in the others.

Thus, a modeling language that can be interpreted by both humans and machines would be advantageous. This modeling language needs to be formal enough to be

interpreted by a machine and must have a graphical presentation layer, which facilitates interpretation and manipulation by humans.

Fig. 3 depicts the basic elements of the ontology-based modeling approach, which satisfies these requirements. Entities of reality are represented in graphical models suited for humans and at the same time in a machine-interpretable formal model (i.e. in an ontology). Both models are deeply intertwined. The ontology provides a formal semantics of the modeling language (Hrgovcic et al. 2013; Kappel et al. 2006) such that related models are interpretable by machine.

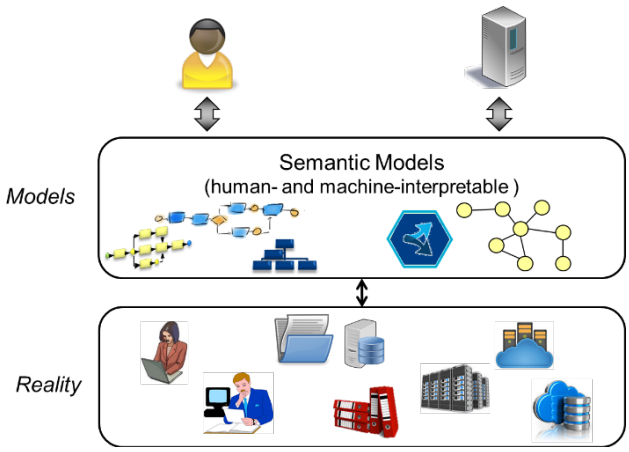

Fig. 3 Our proposal: integrated models for human and machine interpretation

In our ontology-based metamodeling approach, we use ontologies to define the semantics of the modeling language in a formal model, which can be interpreted by a machine. A model engineer can represent the domain knowledge as an ontology with classes, relations and rules. To create a graphical modeling language, the classes can be extended by graphical notations, which can be used by the modeler to create models. Thus, the ontology represents the domain knowledge, which at the same time is the metamodel of a domain-specific modeling language. By making a graphical model, the modeler creates instances of ontology classes. Thus, models are formal with clear semantics. As a result, the ontology-based metamodeling approach achieves two goals: (1) the definition of domain-specific modeling languages with an unambiguous formal semantics, for which (2) the models can be interpreted both by humans and machines.

2 State of the Art

In the following section, we detail research on modeling languages, its formalization for machine interpretation, and approaches to combine formal representation of models with models which are cognitively adequate for humans.

2.1 Modeling Languages

According to Karagiannis and Kühn (Karagiannis and Kühn 2002) a modeling method consists of a modeling language and a modeling procedure, as well as modeling mechanisms and algorithms. In the following, each of these three components is explained in the context of enterprise engineering.

Metamodels are the basis for modeling tools and for the development of the modeling languages. They provide the syntax of a modeling language. A metamodel contains the class hierarchy and the properties representing the modeling elements as well as the relations between them (Jonkers et al. 2003). This corresponds to the so-called abstract syntax. The specification of the graphical notation for each modeling element and relation corresponds to the concrete syntax. The latter should be cognitively adequate to ensure the users' understanding of models that are built from it. The domain-specific conceptualization addresses this aspect by providing modeling elements that are tailored to a given domain. Fill and Karagiannis (2013) analyzed the conceptualization of modeling methods: They use the ADOxx metamodeling platform[1] to investigate how to realize four selected functionalities of enterprise information systems to support user interaction, process-based optimization, interfaces to other systems, and complex analyses.

2.2 Machine Interpretability

To gain its full potential, the purpose of modeling must go beyond transparency and communication, which is what graphical models provide humans with. Models should also be used for automation, and operations such as decision making, analysis, adaptation, and evaluation.

For automation purposes, model knowledge should be machine-interpretable or at least machine-readable. In business process automation, for instance, process models determine the workflow executed by the workflow engine. For decision-making purposes, it is common practice to work with models, for example, as represented by the Decision Model and Notation (OMG 2016).

In keeping with (Hinkelmann et al. 2016), we distinguish between machine-interpretable models and machine-readable models by claiming that the former are represented in a format on which reasoning can be performed. Hence, machine-interpretable models can turn passive data storage into an active device. A machine-interpretable format can be expressed in logic-based languages such as ontologies. Different kinds of reasoning can be applied, depending on the expressivity of the ontology language. Ontologies expressed in the Resource Description Framework Schema (RDFS) (W3C 2014), for example, can be combined with semantic rules to draw new insights from the already existing knowledge base (KB).

[1] ADOxx is a commercial product and trademark of BOC AG.

2.3 Combining Human with Machine Interpretability

In the field of information systems, the human interpretability of modeling refers to metamodels, whereas machine interpretability mainly refers to the formal semantic aspects of models, i.e. ontologies (Hinkelmann et al. 2016). Höfferer (2007) discusses the relationship between metamodels and ontologies by emphasizing that metamodels and ontologies are different but complementary concepts. Ontologies basically furnish both modeling language constructs and their instances with formal semantics (Dietz 2006; Kramler et al. 2006; Kappel et al. 2006). Metamodels on the other hand, mainly provide the syntax and graphical representation for those modeling language constructs. Aßmann, Zschaler & Wagner (2006) assume that ontologies in the Semantic Web and models in model-driven engineering (MDE) were developed in isolation and investigate the role of ontologies, models, and metamodels to bridge the gap between the two communities.

2.4 Semantic Lifting

Semantic lifting is defined as "…the process of associating content items with suitable semantic objects as metadata to turn 'unstructured' content items into semantic knowledge resources" (Azzini et al. 2013). This approach requires the relationship between the human-interpretable and the machine-interpretable modeling languages to be defined (Hrgovcic et al. 2013).

The metamodels for the human-interpretable graphical representations and the machine-interpretable metamodels, e.g. represented in an ontology, are strictly separated. To align them, formal and non-formal (meta)models are mapped by transformation.

Fig. 4 shows the conceptual architecture for semantic lifting. Different model types in the enterprise architecture are created which correspond to different metamodels. These primarily define syntactical aspects as well as certain semantic aspects of model elements. The ontologies define the machine-interpretable semantics of the modeling concepts. In the literature, semantic lifting is also known as semantic annotation (Liao et al. 2015; Fill et al. 2013).

In these approaches, the ontologies are independent from the concepts of the human-interpretable, graphical languages. The ontology comprises class definitions which represent the formal semantics of modeling elements. Furthermore, it includes class definitions which serve to annotate models and model elements. The basis for interoperability is provided by linking model elements of the models and metamodels with ontology concepts.

This approach has been described in and used, for example, in the European research projects LearnPAd (De Angelis et al. 2016) and CloudSocket. (Hinkelmann, Kurjakovic et at. 2016; Hinkelmann, Kritikos et al. 2016; Woitsch, Hinkelmann et al. 2016).

The drawback of this approach lies in the consistency of the semantics between (meta)models and their representation in ontologies. Keeping them separate tends to cause incompatible semantics. This mainly occurs if the project stakeholders do not agree among themselves on a common understanding of important terms beforehand, or if little attention is paid when changes occur, i.e. poor maintenance.

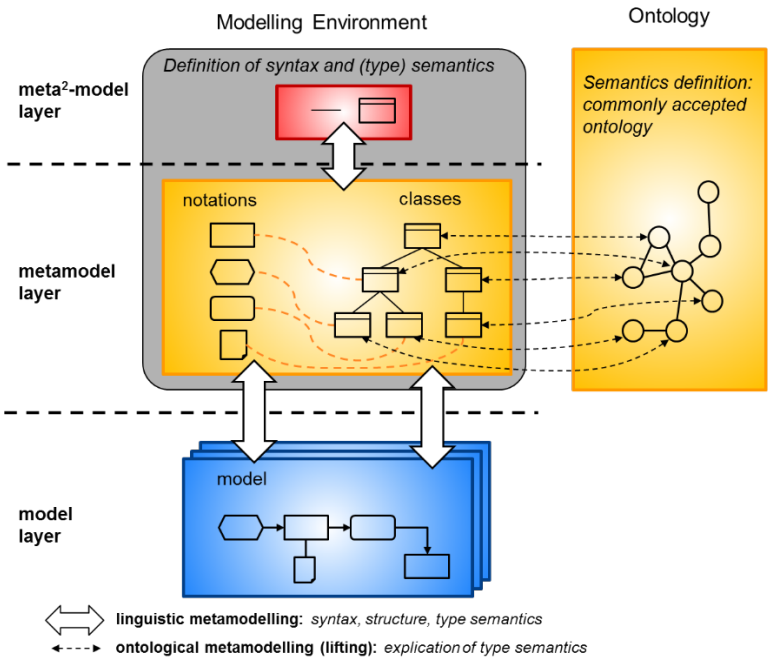

Fig. 4 Metamodels for human-interpretable and machine-interpretable models, (Höfferer 2007)

Having provided a brief overview of related work, we claim that human- and machine-interpretable models should become an integrated model in order to realize the full potential of modeling.

3 Conceptual Solution of Ontology-based Metamodeling

In order to avoid the inconsistency problem between (graphically represented) models and ontologies, a semantic metamodeling approach is proposed which merges the abstract syntax of metamodels with the semantics defined in the ontology. This means that the ontology is used to specify both the semantics and the abstract syntax.

The ontology is extended by a specification of the graphical notation. The difference to the transformation approach is that the semantics is expressed only once

for both human-interpretable and machine-interpretable models. The ontology-based modeling can be regarded as a variant of the MOF metamodeling framework (OMG 2014) where UML is replaced by an ontology language as a metamodeling language.

In the ontology-based metamodeling approach, the ontology itself is also the metamodel for the graphical modeling environment. Only the graphical notation for each concept is defined separately from the semantic description (see Fig. 5). A mapping is defined between concept definition and graphical definition (Nikles and Brander 2009).

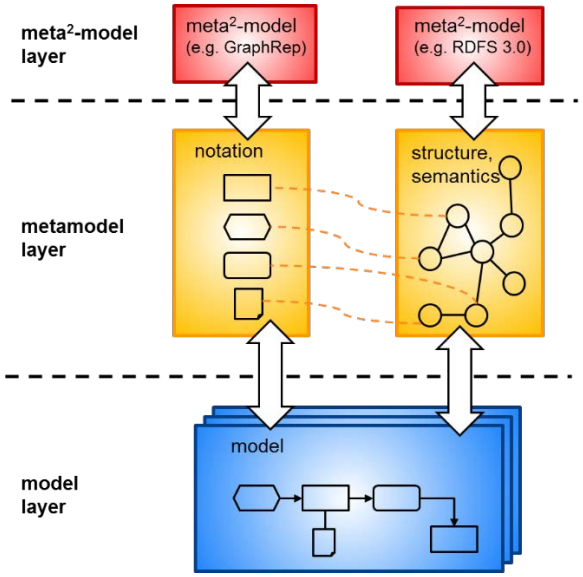

Fig. 5 Ontology-based metamodeling

The semantics is in the ontology, which consists of classes, attributes, relations and constraints. The model layer of Fig. 5 is an instantiation of both semantics and related notations that resides in the metamodel layer. Thus, the model in the bottom layer benefits from both, a semantics that is machine-interpretable and a graphical notation that makes it human-interpretable.

In addition, the ontology-based metamodeling approach fosters the adaptation of a modeling language to fit a specific domain. In order to have a common understanding of the term “adaptation” we refer to the work of Laurenzi et al. (Laurenzi et al. 2017), where the following operations were performed in the metamodel layer using existing modeling languages:

- Identification of needed and unneeded concepts
- Specialization/generalization of concepts
- Restrictions on attribute values
- Injection of constraints among concepts

Specializing/generalizing a concept can refer to both classes and relations. For example, the class “task” in BPMN generalizes the classes “user task”, “manual task”, “service task” and “business rule task”. These can then be further specialized. For example, “user task” can be specialized into “send electronic document”, “send package”, etc. An example for domain restrictions for attribute values is described by Hinkelmann et al. (2016), where an attribute that expresses the functionality of a cloud service can only have values from the APQC Process Classification Framework (APQC 2014). Injection of constraints refers to the additional relations that can occur among modeling elements. These restrict the way concepts can be connected in the models.

In the literature, these operations refer to the actions that typically take place at the design time of Domain-Specific Modeling Languages (DSMLs) (Fowler 2011, Frank 2010; Gray et al. 2008; Mernik et al. 2005; van Deursen et al. 2000). DSMLs shift the complexity of modeling from the model layer (M) to the metamodel layer (MM) (see Fig. 6). In the context of our ontology-based metamodeling, ontology experts work in the metamodel layer to make the modeling easier for the language user.

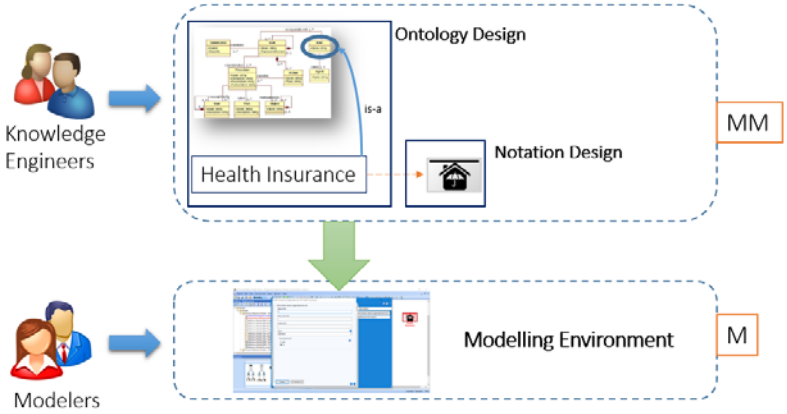

Fig. 6 Domain-specific conceptual modeling with an ontology-based metamodeling approach

In the model layer (M), users make use of the constructs developed in the metamodel layer (MM) to create models. In Fig. 7 we provide an explanatory example, already used by Emmenegger et al. (2016). We assume, for example, that the modeling element “C1” reflects the class “Lane” of BPMN (OMG 2011), while “C2” reflects the class “Role” of the Organizational Model. By adding a relation “r” between the two modeling elements, we allow one or more instances of “Lane” to be connected with one or more instances of “Role”. This enables the language user to refer a specific lane to a particular role so that, for example, the role can be further specified in an organizational model.

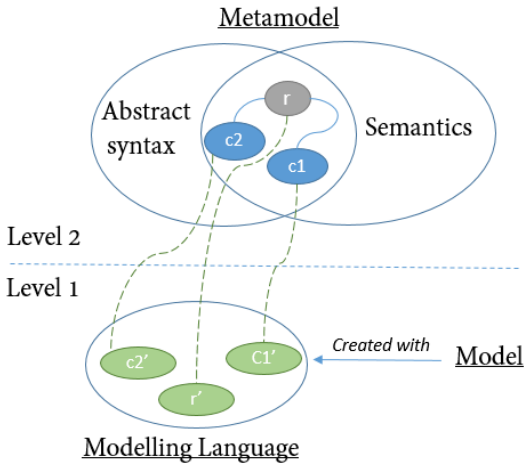

Fig. 7 Two-tier approach, adapted from (Laurenzi et al. 2017)

The degree of freedom in modeling in the model layer (also known as level 1) depends on the level of specificity of the modeling language (also called degree of semantics by Frank (2010)) inserted in the metamodel layer (also known as level 2). The higher the level of specificity is, the more domain-specific is the modeling language. The UML class diagram, for example, provides a general-purpose metamodel with a low specificity level. Hence, the language user can create and connect any classes without restrictions, i.e. the user has a high degree of freedom in the model layer. Such freedom may create ambiguous models, leading to the wrong interpretation of models or modeling the same reality differently, which can make the models questionable. Even worse, a low level of specificity can lead to nonsense models, inconsistency and wrong models. This is a critical issue, because, for example, in the context of Model-Driven Development, it can lead to

a) the creation of error-prone or unintended software, and
b) quality issues such as incomplete requirements or poor implementation.

The BPMN metamodel already provides a higher level of specificity than the UML class diagram. It contains

a) a taxonomy of concepts (e.g. user task specifies task),
b) attributes that might differ from concept to concept (e.g. task vs event), and
c) constraints among concepts following the BPMN guidelines that allow the modeling of structured processes.

For example, a start event initiates the process, an end event ends the process and one or more activities should occur between the two events.

Due to their higher level of specificity, DSMLs promise to enable domain experts to handle the designing and editing of models in a meaningful and less error-prone way, and therefore support the production of high quality models (Kelly and Tolvanen 2008). Moreover, allowing the domain experts to deal directly with familiar language constructs makes the language easy to learn and improves its applicability (Hudak and Paul 1996). DMSLs offer the benefit of a high level of understanding of models among domain experts, fostering not only productivity in design time, but also the optimization phase, where pain points are rapidly identified and actions can be taken accordingly.

One of the main challenges of DSMLs is to inject the metamodel with the appropriate level of specificity. This challenge relates to the design of a DSML, which can be supported by a machine if the language is grounded with a formal semantics, i.e. an ontology-based metamodel.

4 Implementation

In this section, we present the implementation of the ontology-based metamodeling approach as it was developed in the European research project CloudSocket (Woitsch, Hinkelmann et al. 2016).

4.1 The BPaaS Ontology

The metamodel for the service selection, allocation and deployment is represented in the BPaaS Ontology. The ontology in Fig. 8 conceptualizes functional and non-functional specifications of a cloud service.

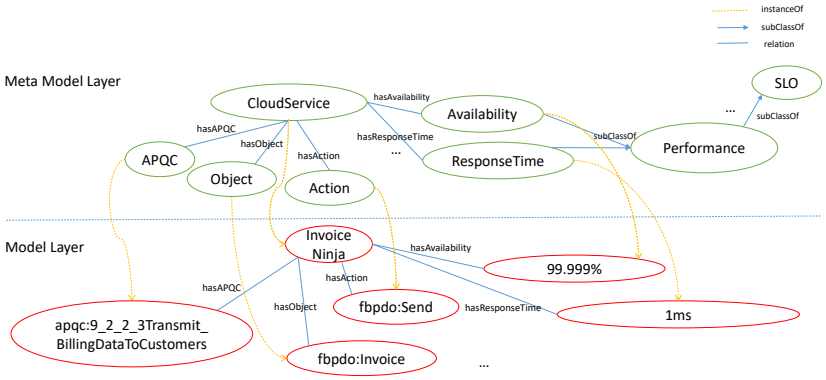

Fig. 8 Part of the BPaaS Ontology developed in the European research project CloudSocket

Fig. 9 shows some of the attributes of the cloud service concept. The functional aspects specify functionalities of a cloud service. They relate to the hierarchy of the APQC Process Classification Framework (APQC 2014), using the relation "cloudServiceHasAPQC". In addition, cloud services functionalities are specified through actions and objects from a predefined taxonomy with the relations "cloudServiceHasAction" and "cloudServiceHasObject". This corresponds to the convention of BPMN to name activities using a verb (i.e. action) and a noun (i.e. object), e.g. "Send Invoice".

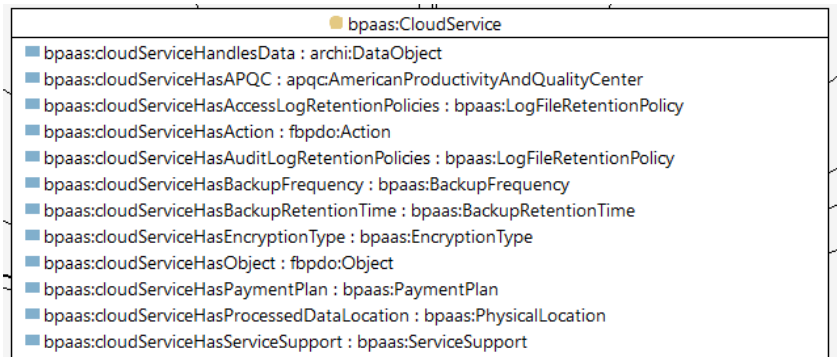

Fig. 9 The cloud service concept implemented in the BPaaS Ontology

The model layer in Fig. 8 depicts the instances of the concepts in the metamodel layer. For example, APQC category, action and object are used to specify the functionalities of the cloud service "InvoiceNinja", i.e. "9.2.2.3 Transmit Billing Data to Customer", "Send" and "Invoice", respectively.

Furthermore, from the BPaaS Ontology a cloud service can also be specified through non-functional aspects, for example availability and response time (see Fig 8). These two refer to the performance category, which in turn is reported in the Service Level Objectives (SLOs) that is listed in the Cloud Service Level Agreement Standardisation Guidelines (C-SIG 2014). Fig. 8 shows the conceptualization of the "availability" and "response time" as subclasses of the "performance" concept, which in turn is a subclass of the "SLO" concept. The bubbles containing values "99.999%" and "1ms" are instances of the classes "Availability" and "ResponseTime", respectively.

The graphical notation for the modeling elements in the model layer are added after the ontology design that takes place in the metamodel layer. In the following, we describe two different types of graphical representations that were implemented for the same ontology-based metamodel.

4.2 Model-Based Representation Implemented in the Metamodeling Tool ADOxx

The modeling language BPMN 2.0 was extended with the Service Description Model to specify both functional and non-functional aspects of a cloud service. The language extension was implemented in ADOxx. As Fig. 10 shows, each lane can be annotated with a cloud service description element, which is specified in the Service Description Model. Specifications occur in the notebook. Fig. 11 shows the functional specifications, while Fig. 12 shows the non-functional specifications. The functional specifications in the notebook reflect the relations to APQC, Action and Object as defined in the BPaaS Ontology. Their values represent the instances in the model layer of Fig. 8. As soon as the notebook is saved, an ontology instance is created.

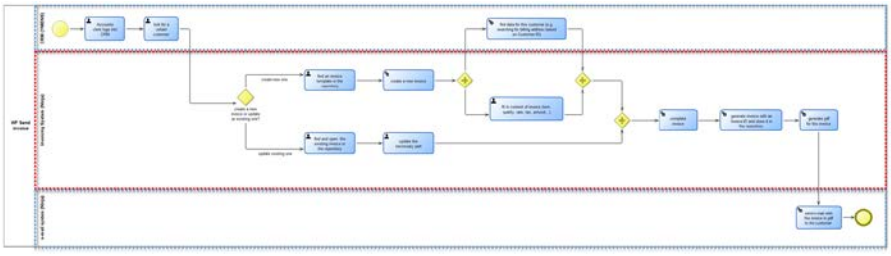

Fig. 10 Extension of BPMN 2.0 implemented in ADOxx

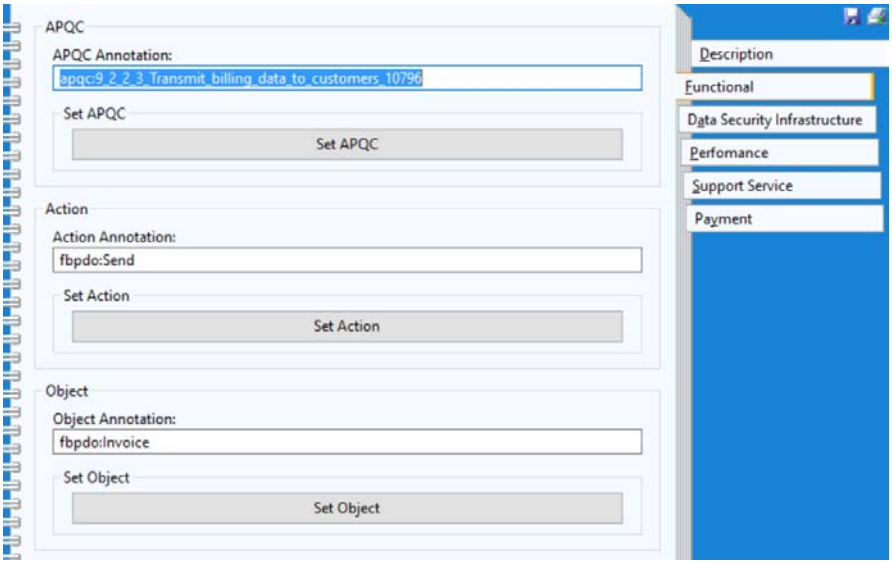

Fig. 11 Non-functional aspects of a cloud service implemented in ADOxx

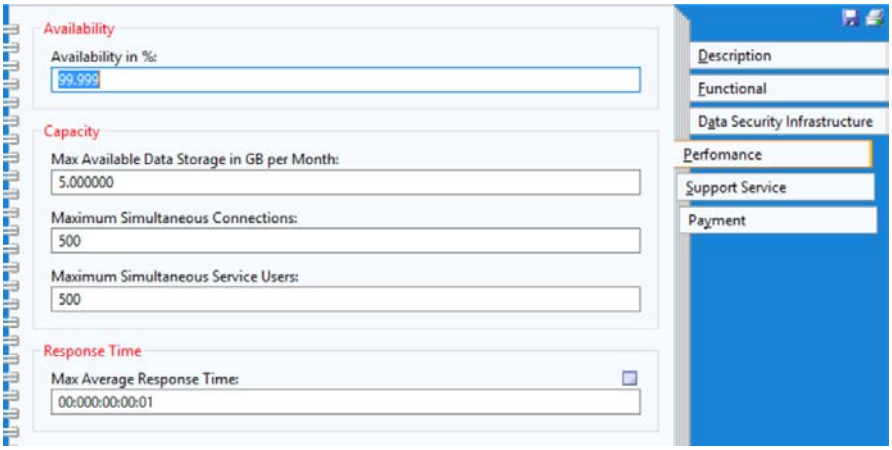

Fig. 12 Non-functional aspects of a cloud service implemented in ADOxx

4.3 Web-Based Representation

Fig. 13 shows the representation of the cloud service (functional and non-functional) specifications implemented in a web-based format. Instead of creating a graphical process model, the functional and non-functional requirements are specified in a web form. The web form is a human-oriented modeling language, which is used as an alternative to the graphical representation of the process using BPMN. Similar to the model-based approach of Section 4.2, all the specifications, including their values, become instances for the related ontology classes as soon as the web page is submitted. Hence, the instance model is formally grounded with an ontology. Both the web-based and the model-based representations refer to the same ontology-based metamodel.

5 Conclusion

With ontology-based metamodels as described in this chapter it is possible to define the semantics of domain-specific modeling languages using ontologies. Models represented in such a modeling language are instances of ontology classes. Because of the formal representation, the models can be interpreted by software systems. On the other hand, the metamodels can be extended with graphical notations for the different modeling elements. Graphical modeling is more appropriate for humans than the creation of formal models. Thus, the ontology-based modeling approach allows for easy modeling using graphical notation and at the same time creates models that can be used for automation and machine-based interpretation. The ontology-

based metamodeling was implemented by extending BPMN in a graphical modeling environment and as a web interface.

In future research, a modeling tool will be developed that integrates modeling and metamodeling in a graphical interface. A knowledge engineer can extend the domain-specific modeling language with new modeling elements on the spot and use them immediately.

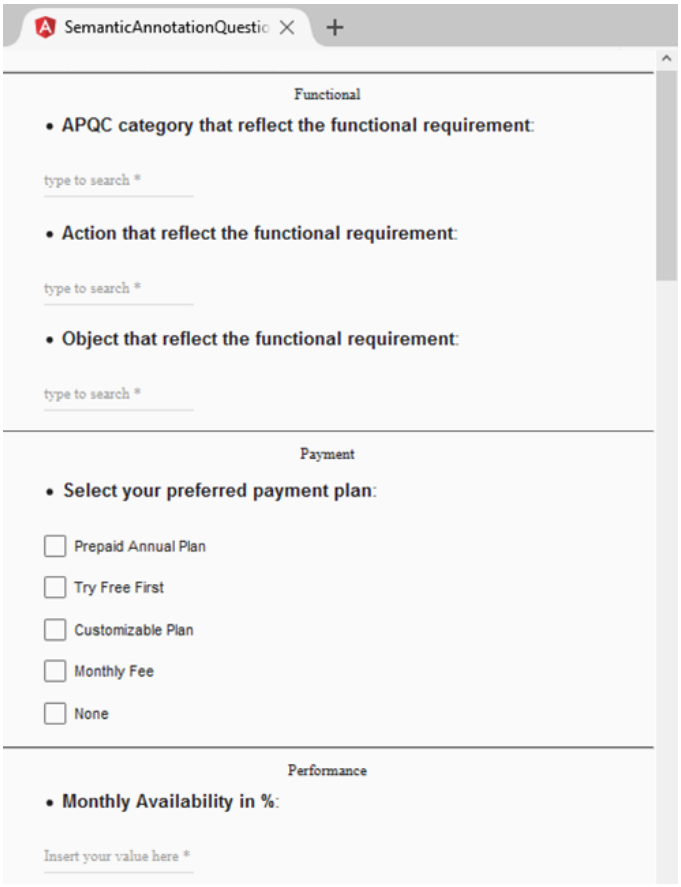

Fig. 13 Graphical representation of cloud service specifications implemented in Angular

Acknowledgement

This research has received funding from the European Community's Framework Programme for Research and Innovation HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

APQC. (2014). Process Classification Framework Version 6.1.1.

Azzini, A., Braghin, C., Damiani, E. and Zavatarelli, F. (2013) Using Semantic Lifting for improving Process Mining: a Data Loss Prevention System case study. *SIMPDA*, pp. 62–73.

C-SIG, 2014. Cloud Service Level Agreement Standardization Guidelines. EC Cloud Select Industry Group.

De Angelis, G., Pierantonio, A., Polini, A., Re, B., Thönssen, B., & Woitsch, R. (2016). Modeling for Learning in Public Administrations—The Learn PAd Approach. In *Domain-Specific Conceptual Modeling* (pp. 575–594). Cham: Springer International Publishing. doi:10.1007/978-3-319-39417-6_26

Dietz, J. L. G. (2006). *Enterprise Ontology. Theory and Methodology*. Berlin Heidelberg: Springer-Verlag.

Emmenegger, S., Hinkelmann, K., Laurenzi, E., Thönssen, B., Witschel, H. F., & Zhang, C. (2016). Workplace Learning - Providing Recommendations of Experts and Learning Resources in a Context-sensitive and Personalized Manner. In *MODELSWARD 2016, Special Session on Learning Modeling in Complex Organizations*. Rome.

Fill, H.-G., Schremser, D., & Karagiannis, D. (2013). A Generic Approach for the Semantic Annotation of Conceptual Models Using a Service-Oriented Architecture. *International Journal of Knowledge Management*, *9*(1), 76–88. doi:10.4018/jkm.2013010105

Fowler, M. (2011). *Domain-specific languages*. Upper Saddle River: Addison-Wesley.

Frank, U. (2010). Outline of a method for designing domain-specific modelling languages. University of Duisburg Essen: ICB.

Gray, J., Fisher, K., Consel, C., Karsai, G., Mernik, M., & Tolvanen, J.-P. (2008). DSLs: the good, the bad, and the ugly. In *Conference on Object Oriented Programming Systems Languages and Applications archive*. Nashville and {É}tats-Unis: ACM.

Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., van der Merwe, A., & Woitsch, R. (2016). A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. *Computers in Industry*, *79*, 77–86. doi:10.1016/j.compind.2015.07.009

Hinkelmann, K., Kritikos, K., Kurjakovic, S., Lammel, B., Woitsch, R. (2016). A Modelling Environment for Business Process as a Service. *CAiSE 2016: Advanced Information Systems Engineering Workshops*, Ljubljana, Slovenia, pp 181-192.

Hinkelmann, K., Kurjakovic, S., Lammel, B., Laurenzi, E. and Woitsch, R. (2016). A Semantically-Enhanced Modelling Environment for Business Process as a Service. *Fourth International Conference on Enterprise Systems ES2016*, Melbourne, Australia, 2-3 November 2016

Höfferer, P. (2007). Achieving Business Process Model Interoperability Using Metamodels and Ontologies. In *European Conference on Information Systems* (pp. 1620–1631). University of St. Gallen. http://www.dke.at/fileadmin/DKEHP/publikationen/metamodell/Hoefferer_BP_interoperability_ontologies.pdf

Hrgovcic, V., Karagiannis, D., & Woitsch, R. (2013). Conceptual Modeling of the Organisational Aspects for Distributed Applications: The Semantic Lifting Approach. In *COMPSACW 2013, 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops* (pp. 145–

150). IEEE. doi:10.1109/COMPSACW.2013.17

Hudak, P., & Paul. (1996). Building domain-specific embedded languages. *ACM Computing Surveys*, *28*(4es), 196–es. doi:10.1145/242224.242477

Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., et al. (2006). Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In O. Nierstrasz, J. Whittle, D. Harel, & G. Reggio (Eds.), *Model Driven Engineering Languages and Systems, Proceedings of the 9th International Conference, MoDELS 2006* (LNCS 4199., pp. 528–542). Genova, Italy: Springer-Verlag.

Karagiannis, D., & Kühn, H. (2002). Metamodelling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer (Eds.), *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.

Karagiannis, D., & Woitsch, R. (2010). Knowledge Engineering in Business Process Management. In *Handbook on Business Process Management 2* (pp. 463–485). Berlin Heidelberg: Springer.

Kelly, S., & Tolvanen, J.-P. (2008). *Domain-specific modeling: Enabling full code generation*. Hoboken: Wiley.

Kramler, G., Kappel, G., Reiter, T., Kapsammer, E., Retschitzegger, W., & Schwinger, W. (2006). Towards a semantic infrastructure supporting model-based tool integration. In *GaMMa '06: Proceedings of the 2006 international workshop on Global integrated model management* (pp. 43–46). New York, NY, USA: ACM Press.

Laurenzi, E., Hinkelmann, K., Reimer, U., Van Der Merwe, A., Sibold, P., & Endl, R. (2017). DSML4PTM: A domain-specific modelling language for patient transferal management. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems* (Vol. 3).

Y. Liao, M. Lezoche, H. Panetto, N. Boudjlida, and E. R. Loures (2015). Semantic annotation for knowledge explicitation in a product lifecycle management context: A survey. *Computers in Industry*, vol. 71, pp. 24–34.

Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, *37*(4), 316–344. doi:10.1145/1118890.1118892

Nikles, S., & Brander, S. (2009). Separating Conceptual and Visual Aspects in Meta-Modeling. In A. Gerber, K. Hinkelmann, P. Kotze, U. Reimer, & A. van der Merwe (Eds.), *Workshop on Advanced Enterprise Architecture and Repositories*. Milano.

OMG. (2011). *Business Process Model and Notation (BPMN) Version 2.0*. Needham, MA: Object Management Group OMG. http://www.omg.org/spec/BPMN/2.0/PDF/

OMG. (2014). *OMG Meta Object Facility (MOF) Core Specification Version 2.4.2* (Vol. 2).

van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific Languages: An Annotated Bibliography. *SIGPLAN Not*, *35*(6), 26–36. doi:10.1145/352029.352035

Woitsch, R. Hinkelmann, K., Juan Ferrer, A.M., Yuste, J.I. (2016). Business Process as a Service (BPaaS): The Smart BPaaS Design Environment. *CAiSE 2016 Industry Track CEUR Workshop Proceedings*, Vol-1600, http://ceur-ws.org/Vol-1600, Ljubljana, Slovenia, 2016

W3C. (2014). RDF Schema 1.1.