On the Conceptualization of a Modeling Language for Semantic Model Annotations

Hans-Georg Fill

Stanford University BMIR / University of Vienna DKE,
251 Campus Drive, Stanford CA 94305, USA
fill@stanford.edu

Abstract. In this paper we describe the theoretical foundations, formal considerations, and technical characteristics that were taken into account for the conceptualization of a modeling language for the semantic annotation of visual models. Thereby it is envisaged to give insights into the underlying processes for the development of new visual modeling languages and thus provide input for a future model of the conceptualization process. To illustrate the realization of the approach we revert to the semantic annotation of a business process model using concepts from the web ontology language OWL, which allows us to show the iterations that were conducted to develop the approach. As a first evaluation the approach has been implemented on a meta modeling platform and will be made freely available to the interested community in the course of the SeMFIS project on www.openmodels.at.

Keywords: Conceptualization, Design, Semantic Annotation, Conceptual Models, Ontologies.

1 Motivation

When a new modeling language is being defined, it is mostly not made explicit how the designers of the modeling language derived the particular elements, relations, attributes, visual representations and model types from the underlying theoretical bodies. Rather, it is often referred to previously existing, similar languages and the incorporation of best practice examples, as e.g. stated in the UML and BPMN specifications [1,2]: "...UML originated with three leading object-oriented methods (Booch, OMT, and OOSE), and incorporated a number of best practices from modeling language design, object-oriented programming, and architectural description languages." [1][p.1] and "This specification represents the amalgamation of best practices within the business modeling community..." [2][p.1]. It is thus not clear whether the creators of the language applied some kind of structured, reproducable process, whether a formal, mathematical approach has been used, or whether the outcome is the result of a serendipitous inspiration of one or several persons. From a scientific point of view it seems however necessary to investigate this process in more detail in order to make the used techniques learnable and thus usable by a wider audience. In order to gain insight into such a process we will describe in the following the underlying theoretical foundations, formal considerations, and technical characteristics that have led to the development of a modeling language for the semantic annotation of conceptual visual models.

2 Background: Semantic Annotation of Models

Several areas in information systems make use of conceptual visual modeling methods today. They are used to represent both static and dynamic phenomena and thus support human communication and understanding [3,4]. Their usage spans from business oriented applications such as for strategic and business process management to the technical realization of information systems in software engineering and IT infrastructure management, c.f. [5]. In recent years, several approaches have been elaborated that focus on the processing of unstructured semantic information that is contained in these models. Similarly to the idea of a semantic web where unstructured textual information is enhanced by using references to ontologies, the unstructured information that is expressed in the models using natural language is made machine processable by defining mappings to semantic schemata. In contrast to approaches that aim for a complete, a-priori formal definition of all model content, e.g. [6,7], the use of semantic annotations gives way to a flexible, stepwise semantic enrichment. Thus, a domain expert can easily choose which parts of the contained semantic information shall be expressed in a formal way and which parts can be left in the traditional natural language format. At the same time, the traditional way of using the modeling language is not changed.

Especially in the area of business process modeling a range of benefits have been described that are based on the processing of such semantic annotations. Examples include the measuring of the similarity of models [8], the automatic execution of process instances using web services [9] or the detection of regulatory compliance [10]. However, the approaches that have been described in the past are usually targeted towards a specific business process modeling language. It therefore seemed worthwhile to develop an approach that can be applied to arbitrary conceptual modeling languages and thus reap the benefits of semantic annotations also for other knowledge areas and according modeling languages.

For this purpose we will first describe the foundations and concepts our approach is based on and then discuss the process of how a modeling language has been conceived that permits to annotate arbitrary conceptual models with concepts from ontologies.

3 Fundamental Concepts

For describing our approach we will refer to a number of specific terms in regard to modeling methods and modeling languages. Therefore, we will briefly define some fundamental concepts and thus clarify their meaning in our context.

3.1 Components of Modeling Methods

To define the components of modeling methods we refer to the generic framework developed in [11] - see also figure 1. A modeling method is thereby composed of a modeling technique and mechanisms and algorithms. The modeling technique contains a modeling language, which is specified by a syntax, semantics, and visual notation, and a modeling procedure that defines the steps and results for applying the modeling language. The semantics of the modeling language defines the meaning of the syntactic elements by establishing a mapping to a semantic schema. The mechanisms and algorithms are used in the modeling procedure and can either be generic, i.e. applicable to arbitrary modeling languages, specific, i.e. applicable only to a particular set of modeling languages or hybrid, i.e. they can be parameterized for a specific modeling language.

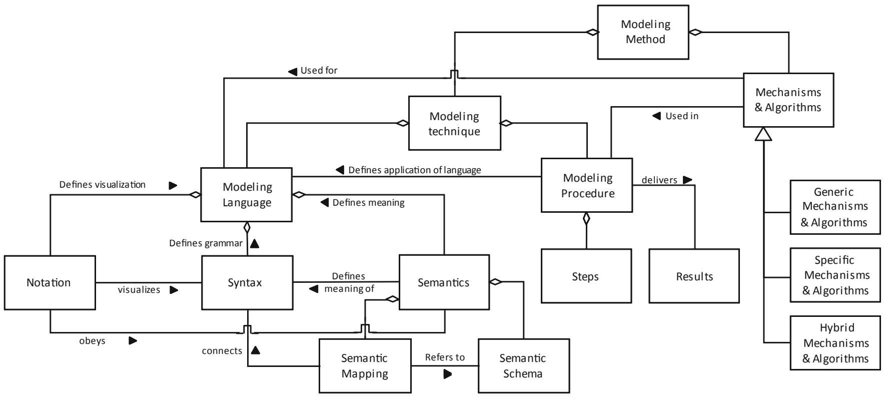


Fig. 1. Components of Modeling Methods [11]

For the conceptualization of a modeling method at least some of these generic concepts need to be specified. To do so it can be chosen from several entry points: One would be to start with the definition of the syntax of the modeling language and its semantics. Then assign an appropriate visual notation to the syntactic elements based on the semantics and finally specify the details of the modeling procedure and the corresponding mechanisms and algorithms. Another direction would be to focus on the modeling procedure and its results and then derive the according modeling language and the mechanisms and algorithms. Similarly, if one wants to focus on particular mechanisms and algorithms, e.g. for conducting mainly machine based processing, it is also an option to start from there and then develop the modeling language and the modeling procedure. However, a modeling method does not necessarily need to contain all the elements shown in the framework and not all of them to the full extent. The minimal set to create visual models is certainly the definition of a syntax and a visual notation together with at least a semantic description in the form of natural language explanations. This allows to create visual models that may even be processed by generic mechanisms and algorithms that do not require a further semantic specification. Many of the currently used standards for visual modeling languages, e.g. UML and BPMN, only contain such a minimal set and leave it to the modeler to determine the most suitable modeling procedure.

3.2 Conceptual Models and Ontologies

To describe our approach we also need to define the terms *conceptual model* and *ontology.* By a conceptual model we understand a visual representation of some aspects of the physical and social world for the purpose of human understanding and communication [4]. Based on the previous statements about the components of modeling languages, such a conceptual model is created based on a formal - in the sense of machine-processable - abstract and concrete syntax, which are specified by a schema or grammar, and a distinct graphical notation [4,3]. As it is not primarily directed towards machine processing, the definition of formal semantics is not compulsory but may be added if necessary, e.g. to conduct simulations. For defining the grammar of conceptual models it can be reverted to proprietary specifications such as the Eclipse modeling framework [12] or ALL [13] or standardized specifications such as the meta object facility (MOF) [14].

An ontology on the other hand can be characterized as ”a shared and common understanding of some domain that can be communicated across people and computers” [15][p.184]. In contrast to conceptual models, ontologies are based on computer-usable definitions and are usually expressed in a logic-based language [16]. This makes them particularly useful for specifying and processing structured vocabularies that make the relationships between different terms explicit [17]. In addition, ontologies can today be easily interchanged using some of the widely used languages such as RDF, RDFS or the web ontology language (OWL) that come with a formal semantic specification that allow for automated reasoning mechanisms such as consistency checking and the classification of terms [18].

4 Conceptualization of a Semantic Annotation Modeling Language

Based on these foundations the process of developing a modeling language for semantically annotating arbitrary types of conceptual models can now be described. In particular we will investigate the theoretical foundations that were considered at the beginning, the formal considerations for realizing the modeling language in a way that can be processed by machines and the technical characteristics that needed to be taken into account.

4.1 Theoretical Foundations

As there are several approaches available that discuss the semantic annotation of conceptual models - in particular business process models - it was first investigated how semantic annotations have been conducted previously and which of the described techniques could be reused for our purposes. Thereby, two main directions could be identified - see also figure 2: The first direction concerns the *translation* of all model information into an ontology language that provides formal semantics, e.g. [19,8]. By enriching the resulting ontology skeleton with additional information about the semantic contents of the models, a machine processable representation can be established. To illustrate this with an example, consider the representation of a business process as an event driven process chain (EPC), c.f. [19]. To translate such a process into an ontology, at first the modeling language for EPCs is translated into ontology concepts, e.g. OWL classes. The resulting ontology can then be refined in OWL, for example by defining the "control flow" properties that connect "functions" and "events" of the EPC as a transitive property. Based on the formal semantics defined for OWL a reasoner can thus correctly interpret that instances of functions and events that are connected with "subsequent" properties are transitively connected to each other.

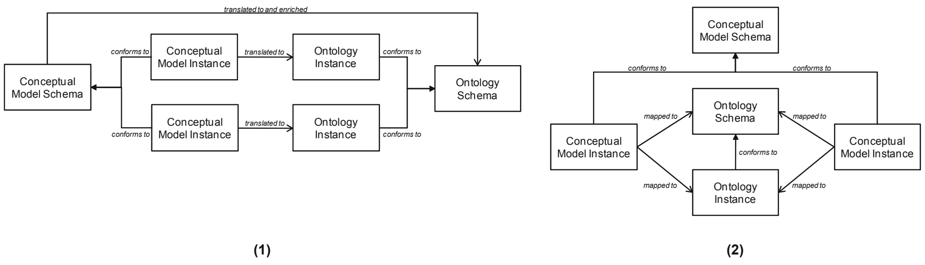


Fig. 2. Two Variants for Semantic Annotations of Conceptual Models

The second direction that can be found in the literature on semantic annotations for conceptual models is characterized by using mapping relations between the elements of a conceptual model and concepts in an ontology, e.g. [20,21,22]. This permits to process the elements in the ontology using formal semantics and thus detect for example ontology concepts that are similar to a concept that participates in such a mapping. Based on this information, the mappings from these similar ontology concepts to the same or other conceptual models can then be used to detect similar elements on the side of conceptual models, as e.g. described by [20].

Although these directions have been described primarily in the context of business process modeling so far, they seemed to be applicable also to other conceptual modeling languages. However, the literature on the application of these approaches to practical scenarios that would allow an evaluation of which of the approaches may be beneficial compared to the other is not yet available. At best, descriptions about the successful application to concrete use cases can be found, e.g. [23,21,22], thus illustrating potential advantages and shortcomings of the approaches. Therefore, it had to be decided which direction should be taken for our approach and we selected the second direction, i.e. the mapping between conceptual models and ontologies. The main reason for this choice was, that we envisaged that the approach would thus be more loosely coupled and could be more easily applied to arbitrary conceptual modeling languages. This loose coupling mainly stems from the fact that no semantic enrichment of the schema of the conceptual models is necessary but that only a mapping has to be defined.

The next theoretical consideration concerned the choice of the language for specifying ontologies. As Obrst has pointed out there is a spectrum of languages available that can be used for this purpose, each with its own advantages and pitfalls [16]. As we wanted to keep our approach as flexible as possible we chose the web ontology language (OWL) for representing ontologies. OWL is widely used in several domains, comes in the form of an official standard by the W3C and is well supported by a range of tools and APIs. In addition, it can serve various goals of using semantic annotations for conceptual models such as the building of formal thesauri, the representation of complex domain knowledge in a formal, machine processable way or as a starting point for executing queries on the formal specifications and the definition of rules, e.g. by using the semantic web rule language SWRL [24,25].

4.2 Formal Considerations

Based on the theoretical choices for using a mapping approach to the web ontology language, the next part was the actual translation of these choices into a concrete modeling language that would be able to use these concepts. Thus, it had to be decided how the mapping should be formally defined in terms of a modeling language and how an ontology in the OWL format should be represented. At the same time it should be ensured that the models that would result from using such a modeling language could be easily processed in a meaningful way. To represent OWL ontologies in the form of models several solutions have been described in the past. These include for example the ontology definition meta model by OMG [26] and a number of prototypes that originated from research projects, e.g. [27,21]. For the purposes of semantic annotations we chose to re-use a previously developed implementation that allows to represent OWL ontologies in the form of visual models but does not itself require the implementation of formal semantic restrictions [28]. In particular, this approach provides algorithms that allow to exchange representations of OWL ontologies with the Protégé ontology management platform. Thereby, the formal semantic restrictions are enforced on the Protégé platform and its attached reasoners and the ontologies are only represented in the modeling environment. To conduct reasoning tasks, requests can thus be sent to Protégé and the results then transferred back into the modeling environment. This not only permits to reuse already developed algorithmic solutions, e.g. for the similarity matching between concepts of an ontology, but also as a basis for advanced manipulations of the ontology concepts such as the execution of semantic queries or rules on the side of Protégé, e.g. by using the SPARQL or SWRL extensions [25].

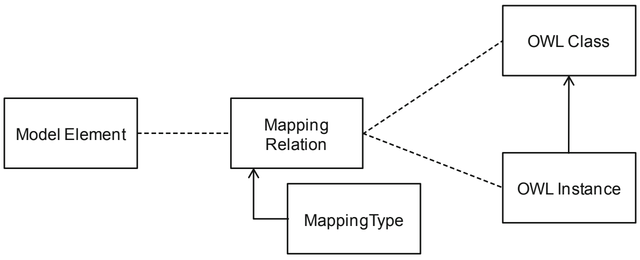


Fig. 3. Theoretical Conception of the Semantic Annotation Approach

To illustrate the path of developing the formal description of the semantic annotation approach, we will describe three evolution steps for realizing a modeling language for semantic annotations. Thereby we intend to show what considerations have to be made for creating the modeling language. In figure 3 the fundamental idea of the approach is shown: For any kind of model element a mapping relation is defined to either an OWL class or an instance of an OWL class. To keep the illustration simple we omitted the possibility of creating mappings to properties. Furthermore, the mapping relation can be detailed by a mapping type, which defines whether the mapped ontology entity *is equal*, *broader* or *narrower* to the meaning of the model element or *refers to* a particular meaning of an ontology entity.

Based on this idea, a first variant for translating these concepts into a modeling language is shown in figure 4. The left part in figure 4 shows the concrete syntax and according visual representation of a sample part of a business process model. It contains an *activity* element, a *decision* element and a *subsequent* relation between the two. In addition, *name* attributes with the values "Write letter"

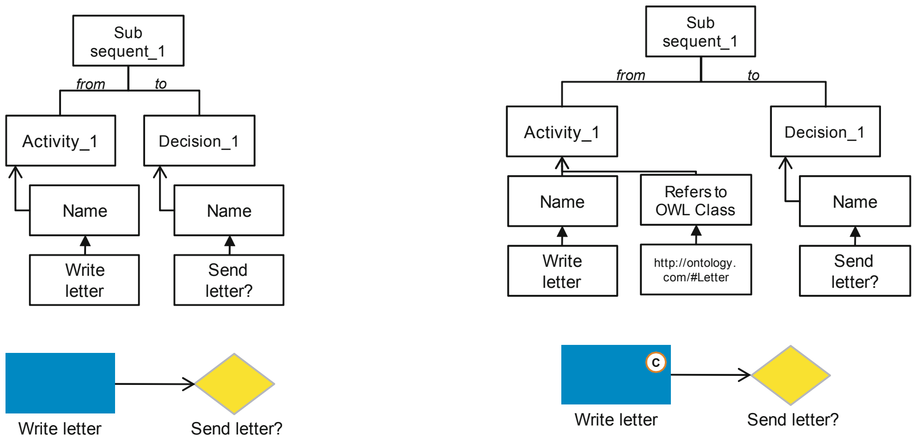


Fig. 4. Sample of an Excerpt of a Conceptual Business Process Model and a Variant for a Semantic Annotation

and "Send letter?" are linked to the model elements. As a first variant, the set of attributes for activities can be extended to give users of the modeling language the possibility to specify an OWL class that stands for the refers-to mapping type. If a value for such an attribute is present it can also be used to change the visual representation accordingly as shown on the right side of figure 4.

Although this first variant contains already all necessary information for the mapping between model elements and ontology concepts and could be directly processed by algorithms, it has several limitations. One limitation is that the original modeling language needs to be extended with an attribute to contain the mapping to the ontology concepts. This may not pose a serious limitation for many modeling languages, but it may lead to difficulties if certain algorithms depend on the original state of the modeling language and may need to be adapted in case of a modification. Another limitation is that for each annotation the exact reference to the ontology concepts needs to be known, i.e. the user of the modeling language has to know the URI of the ontology concept and insert it as an attribute value. Although this could be resolved on the user interface level, it seemed worthwhile to investigate further options for a better representation.

Based on these considerations a second variant was created as depicted in figure 5. It features a separate model type for representing the information contained in OWL ontologies. As mentioned above it does however not include any formal semantic restrictions but just presents the syntactic information

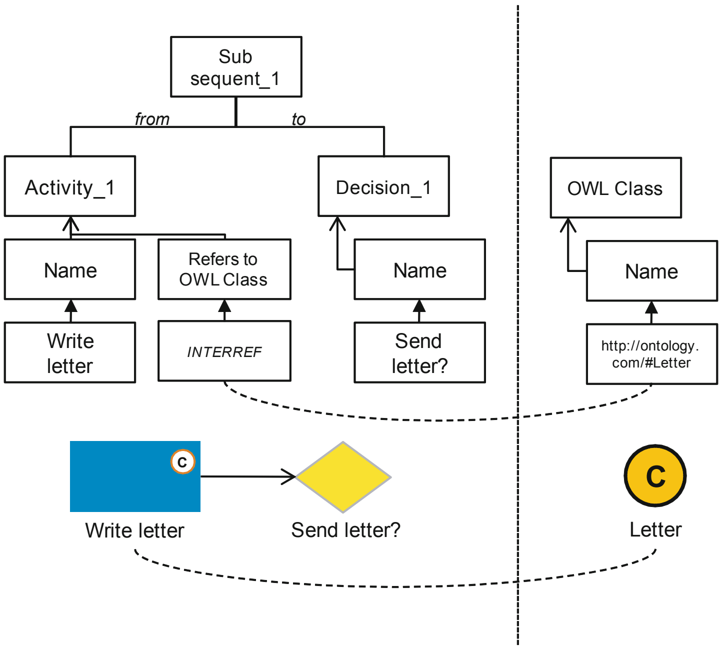


Fig. 5. Variant of a Semantic Annotations using a Model-based Ontology Representation

contained in an OWL ontology. For the example here, the OWL representation has been simplified to highlight the key aspects. In this way, only an *OWL Class* model element and a name attribute are shown. In a similar fashion also OWL properties and instances could be represented. However, the presented information is already sufficient to enable users to map the extended modeling language for a business process model to the OWL model representation. In contrast to the first variant, a user can easily select the ontology concepts that shall be used for the annotation without knowing about the exact reference of the ontology concepts. The ontology representation on the right side can thereby either created by hand - e.g. by an expert user - or imported from an ontology toolkit such as Protégé by using an import mechanism. Due to the lack of formal semantics in the model representation, any modifications of the ontology relationships must however be checked using an external source, e.g. through a reasoner attached to Protégé.

Despite the progress that could be made with the second iteration, the problem of modifying the original modeling language had not been solved. This led to a third iteration as described in figure 6. Here, the loose coupling between the original modeling language, the annotations, and the ontology representation is fully achieved. This is accomplished by introducing a third model type besides the business process model and the ontology model: This *annotation model type* contains references to both the elements of the modeling language that shall be annotated and the ontology model. It also permits to specify the annotation type - shown in figure 6 again by the example of the *Refers-to* annotation type. As the annotation model type provides all necessary information no modification of the original modeling language is necessary. This approach is similar to previously discussed approaches in the context of model weaving [29], however

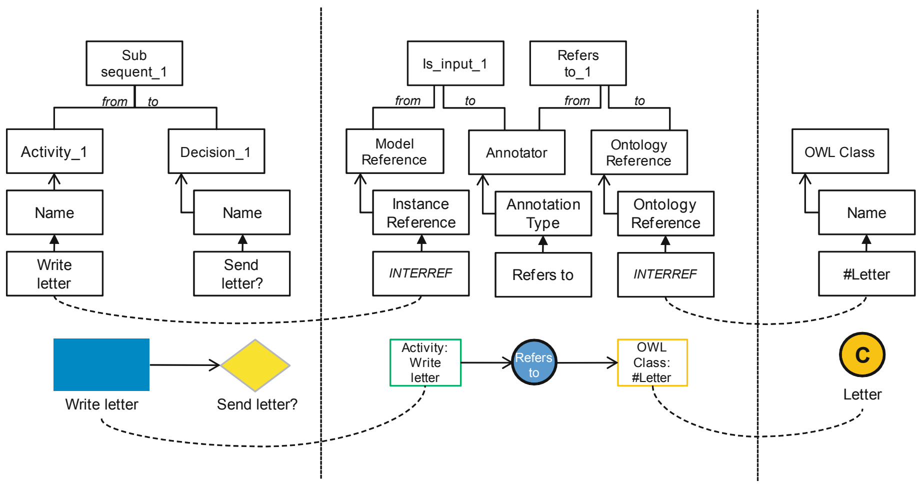
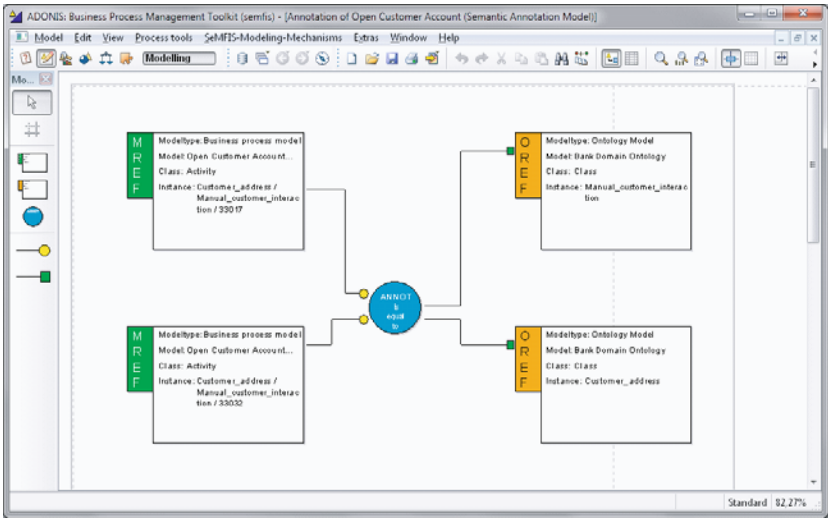


Fig. 6. Resulting Final Variant Using Two Additional Model Types for the Semantic Annotation

we use an explicit visual representation for the annotation model type. Thereby, a user can edit the annotations in a visual form without having to deal with a formal specification language.

4.3 Technical Characteristics

Besides the formal considerations for the conception of the modeling language also certain technical characteristics need to be taken into account. This stems from the fact that in case the modeling language shall be implemented in software, a translation from the abstract formal definitions into a concrete programming or machine processable description language needs to be accomplished. For this purpose it can be chosen from two directions: Either the modeling language is built from scratch using a standard programming language such as Java or C++ or some modeling tools such as specific programming APIs or a meta modeling platform are used. Although implementing the modeling language in a programming language from scratch may offer a maximum degree of freedom, it is certainly not an efficient way to do so when taking into account all the necessary details for the user interface, logic, and data storage level of such an application. And even if an existing platform or API is used, there are still several technical choices that need to be made.


ADONIS: Business Process Management Toolkit (semfis) - [Annotation of Open Customer Account (Semantic Annotation Model)]
Model
Edit
View
Process tools
SeMFIS-Modeling-Mechanisms
Extras
Window
Help
Modelling
Mo...
MREF
OREF
ANNOT is equal to
Standard
82,27%

Modeltype: Business process model
Model: Open Customer Account...
Class: Activity
Instance: Customer_address / Manual_customer_interaction / 33017

Modeltype: Ontology Model
Model: Bank Domain Ontology
Class: Class
Instance: Manual_customer_interaction

Modeltype: Business process model
Model: Open Customer Account...
Class: Activity
Instance: Customer_address / Manual_customer_interaction / 33032

Modeltype: Ontology Model
Model: Bank Domain Ontology
Class: Class
Instance: Customer_address

Fig. 7. Screenshot of the Implementation of the Semantic Annotation Model

For the technical realization of the described semantic annotation modeling language we used the ADOxx[1] meta modeling platform that is provided for free by the www.openmodels.at community. The choice for this platform was mainly

[1] ADOxx is a registered trademark of BOC AG.

based on its extensive funcationalities for implementing modeling languages, its industry ready scalability and robustness as well as existing skills on the side of the authors with its proprietary configuration languages ALL, ADL, and GRAPHREP [13]. The platform offers many options for implementing arbitrary types of visual modeling languages. To highlight some of the technical characteristics we will discuss two aspects: the user interface level and the aspects of data exchange and storage. For the user interface level it had to be taken into account that the semantic annotations can be easily created by hand and that the visual representation offers some guidance to the users. This concerned in particular the representation of the elements that reference the model elements and the ontology elements - see figure 7. It was therefore decided to integrate the details of the reference such as the model type, the model instance, the referenced class, and the referenced instance into the visual representation. Thus, the user can immediately see without further interaction what the reference stands for. The same considerations were applied for the references to ontology models. Another aspect of the user interface level that we would like to highlight is the choice of the colors: Here the decisive factor was that the user should be able to easily distinguish the elements from each other. Therefore different colors were chosen for the *model reference* and the *ontology reference* elements and for the two relations, the *is input* relation and the *refers to* relation. The choice for the shapes was not driven by any particular consideration but was targeted towards a neutral representation that would not evoke any references to existing symbols.

In regard to the data exchange and storage aspects the used meta modeling platform offered several functionalities that helped to focus on the conceptualization of the modeling language itself. Thus, it had not to be taken into account how the modeling language in the form of a meta model nor the actual model instances are stored in a database as the platform would handle that automatically and in an efficient way. The same applied to the use of import and export interfaces for exchanging ontology models with the Protégé platform. ADOxx provides a generic XML format for importing and exporting models that could be easily created by a specifically developed plugin for Protégé. If these functionalities had not been available particular effort would have been required to implement according database and XML schemata.

Also in terms of scalability and applicability to real scenarios, the choice for the ADOxx platform provides several advantages. These include the fact that ADOxx has been applied to many use cases in industry where several domain specific modeling languages (DSML) have been developed [30]. The proposed approach for using semantic annotations can be easily adapted to support a variety of existing other modeling languages and practical scenarios. In addition, also technical functionalities in regard to programmatic access to the models provide further opportunities. It is thus planned to use the WSDL interface for ADOxx to develop a web-based annotation tool. A first prototype for this approach that is based on the Google Web Toolkit and the SmartGWT[2] API is currently under development and will be shortly available [31,32]. Based on

[2] SmartGWT is a registered trademark of Isomorphic Software.

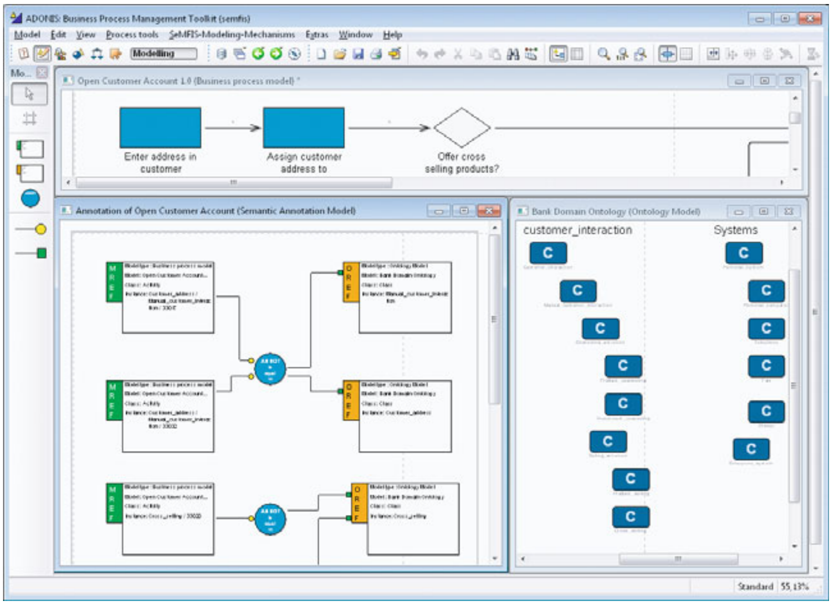

ADONIS: Business Process Management Toolkit (semfis)
Model Edit View Process tools SeMFIS-Modeling-Mechanisms Extras Window Help
Modelling
Open Customer Account 1.0 (Business process model) *
Enter address in customer
Assign customer address to
Offer cross selling products?
Annotation of Open Customer Account (Semantic Annotation Model)
Bank Domain Ontology (Ontology Model)
customer_interaction
Systems
Standard 55,13%

Fig. 8. Screenshot of the Implementation on the ADOxx Platform

these developments it is then envisaged to further investigate and advance the scalability of the approach by making it available for a wider audience on the web.

For the further evaluation of the implementation of the semantic annotation modeling language it will be made freely available in the course of the SeMFIS project on the www.openmodels.at platform[3]. Thereby, it is envisaged to receive further feedback from the community, especially in regard to the scalability and practical applicability of the approach.

5 Conclusion and Outlook

With the above description that highlighted some of the key choices during the development of the semantic annotation modeling language it can now be discussed which implications such a description may have. Clearly, the way how modeling languages are being realized today is not - or maybe not yet - a process that adheres to a well-defined reproducable process. Many of the choices that are made during the development are currently more based on intuition and previous experience than on a sound theoretical foundation. However, there are some parts in the development process where either existing theoretical approaches could be directly applied or where it seems worthwhile of developing them.

In particular we see three parts where this is the case: The first and probably most obvious concerns the choice of the graphical representation of the elements

[3] See `http://openmodels.at/web/semfis/`

and relations of a modeling language. For a long time several areas of science have studied the meaning of signs, the perception of color or the cultural implications that follow from this. It has already been described how systems could be realized that support the creators of modeling languages in choosing appropriate visual representations for this purpose [5]. This could be further developed and generate a direct benefit for the conceptualization of modeling languages in general.

Another part where the use of existing theories and their further adaptation to the conceptualization of modeling languages may be beneficial is the optimization of the syntax of the modeling language. The field of databases developed a large number of optimization techniques that may be able to serve as starting points for further developments. Similar to the principles of normalization and optimization of relational models [33], one could imagine also a theoretical approach for optimizing the syntactic representation of a visual modeling language. As described by the iterations that were presented in section 4 such an optimization needs to take into account several dimensions at the same time to be successful. It would not only have to focus on an efficient implementation in terms of a data structure, but the data efficiency may even be sacrificed to a certain extent to allow for a better user experience or a better application of algorithms.

The third aspect that is probably the most difficult to achieve and that has not been considered so far is the tighter collaboration with prospective real users of the modeling language. Although this is usually hard to achieve for scientists who are not tightly integrated with industry - e.g. based on common projects as it is done in consortium research [34] - this seems to be the only way to enhance the conceptualization process in terms of real usability. By receiving continuous feedback of the future users and the immediate, structured response to the needs expressed therein, is likely to be the sole option for arriving at a quasi-optimal solution that not only has an academic impact. For this purpose new developments in the fields of crowd-sourcing and social network approaches could bring about interesting options for realizing such tasks also on a low-budget basis and for integrating professionals in scientific development processes.

Acknowledgements

We would like to thank the three anonymous reviewers for their very constructive and valuable comments. The work on this paper has been funded by the Austrian Science Fund (FWF) in the course of an Erwin-Schrödinger-Fellowship grant number J3028-N23.

References

1. Object Management Group OMG: OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2 (2007), http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/ (accessed March 01, 2011)

2. Object Management Group (OMG): Business Process Model and Notation (BPMN) Version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0/PDF/ (accessed March 01, 2011)
3. Wand, Y., Weber, R.: Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. Information Systems Research 13(4), 363–376 (2002)
4. Mylopoulos, J.: Conceptual Modeling and Telos. In: Loucopoulos, P., Zicari, R. (eds.) Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development, pp. 49–68. Wiley, Chichester (1992)
5. Fill, H.G.: Visualisation for Semantic Information Systems. Gabler (2009)
6. Peleg, M., Tu, S.: Design Patterns for Clinical Guidelines. Artificial Intelligence in Medicine 47(1), 1–24 (2009)
7. Becker, J., Breuker, D., Pfeiffer, D., Raeckers, M.: Constructing Comparable Business Process Models with Domain Specific Languages - An Empirical Evaluation. In: 17th European Conference on Information Systems (ECIS), Verona, Italy (2009)
8. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: Roddick, J., Hinze, A. (eds.) Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007). Australian Computer Science Communications, vol. 67, pp. 71–80. ACM, New York (2007)
9. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: a vision towards using semantic web services for business process management. In: IEEE International Conference on e-Business Engineering, 2005, ICEBE 2005, pp. 535–540 (2005)
10. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting Regulatory Compliance for Business Process Models through Semantic Annotations. In: 4th International Workshop on Business Process Design, Milan (2008)
11. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2002. LNCS, vol. 2455, p. 182. Springer, Heidelberg (2002)
12. McNeill, K.: Metamodeling with EMF: Generating concrete, reusable Java snippets (2008), http://www.ibm.com/developerworks/library/os-eclipse-emfmetamodel/index.html?S_TACT=105AGX44&S_CMP=EDU
13. Fill, H.G.: UML Statechart Diagrams on the ADONIS Metamodeling Platform. Electronic Notes in Theoretical Computer Science 127(1), 27–36 (2004)
14. OMG, O.M.G.: Meta Object Facility (MOF) Specification 2.0 (2006)
15. Studer, R., Benjamins, R., Fensel, D.: Knowledge Engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197 (1998)
16. Obrst, L.: Ontologies for semantically interoperable systems. In: Proceedings of the 12th International Conference on Information and Knowledge Management. ACM Press, New Orleans (2003)
17. Horrocks, I., Patel-Schneider, P., Van Harmelen, F.: From SHIQ and RDF to OWL: The Making of a Web Ontology Language. Web Semantics: Science, Services and Agents on the World Wide Web 1(1), 7–26 (2003)
18. W3C: OWL Web Ontology Language - Overview W3C Recommendation 10 February 2004 (2004), http://www.w3.org/TR/owl-features/ (accessed September 16, 2005)
19. Thomas, O., Fellmann, M.: Semantic Business Process Management: Ontology-based Process Modeling Using Event-Driven Process Chains. IBIS 2(1), 29–44 (2007)

20. Hoefferer, P.: Achieving Business Process Model Interoperability Using Metamodels and Ontologies. In: Oesterle, H., Schelp, J., Winter, R. (eds.) 15th European Conference on Information Systems (ECIS 2007), University of St. Gallen, St. Gallen, pp. 1620–1631 (2007)
21. Fill, H.G.: Design of Semantic Information Systems using a Model-based Approach. In: AAAI Spring Symposium. Stanford University, CA (2009)
22. Fill, H.G., Reischl, I.: An Approach for Managing Clinical Trial Applications using Semantic Information Models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) Business Process Management Workshops - BPM 2009. Lecture Notes in Business Information Processing. Springer, Ulm (2009)
23. De Francisco, D., Grenon, P.: Enhancing telecommunication business process representation and integration with ontologised industry standards. In: Hepp, M., Hinkelmann, K., Stojanovic, N. (eds.) Proceedings of the 4th International Workshop on Semantic Business Process Management (SBPM 2009). ACM, New York (2009)
24. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004), `http://www.w3.org/Submission/SWRL/` (accessed September 16, 2007)
25. O'Connor, M., Knublauch, H., Tu, S., Musen, M.A.: Writing Rules for the Semantic Web Using SWRL and Jess. In: Protégé with Rules Workshop, Held with 8th International Protégé Conference, Madrid, Spain (2005)
26. OMG, O.M.G.: Ontology Definition Metamodel, Third Revised Submission to OMG/ RFP ad/2003-03-40. Technical report (2005), `http://www.omg.org/docs/ad/05-08-01.pdf` (accessed September 16, 2005)
27. Leutgeb, A., Utz, W., Woitsch, R., Fill, H.G.: Adaptive Processes in E-Government - A Field Report about Semantic-based Approaches from the EU-Project FIT. In: Proceedings of the International Conference on Enterprise Information Systems (ICEIS 2007). INSTICC, Funchal, Madeira, Portugal, , pp. 264–269 (2007)
28. Fill, H.G., Burzynski, P.: Integrating Ontology Models and Conceptual Models using a Meta Modeling Approach. In: 11th International Protégé Conference, Amsterdam (2009)
29. Del Fabro, M.D., Valduriez, P.: Semi-automatic model integration using matching transformations and weaving models. In: SAC 2007 Proceedings of the 2007 ACM Symposium on Applied Computing. ACM, New York (2007)
30. BPTrends: The 2005 EA, Process Modeling and Simulation Tools Report - Adonis Version 3.81 (2005), `http://www.boc-group.com/info-center/downloads/detail/resource/bptrends-review-of-adonis/` (accessed March 30, 2011)
31. Smeets, B., Boness, U., Bankras, R.: Beginning Google Web Toolkit - From Novice to Professional. Apress (2008)
32. Software, I.: Smart GWT(TM) Quick Start Guide - Smart GWT v2.4 November 2010 (2010), `http://www.smartclient.com/releases/SmartGWT_Quick_Start_Guide.pdf` (accessed March 30, 2011)
33. Codd, E.: A relational model of data for large shared data banks. Communications of the ACM 13(6), 377–387 (1970)
34. Oesterle, H., Otto, B.: Consortium Research - A Method for Researcher-Practitioner Collaboration in Design-Oriented IS Research. Business & Information Systems Engineering 5/2010 (2010)