Supporting Software Language Engineering by Automated Domain Knowledge Acquisition

Henning Agt

Technische Universität Berlin
Einsteinufer 17
10587 Berlin
Germany
henning.agt@tu-berlin.de

Abstract. In model-driven engineering, domain-specific languages (DSLs) play an important role in providing well-defined environments for modeling different aspects of a system. Detailed knowledge of the application domain as well as expertise in language engineering is required to create new languages. This research work proposes automated knowledge acquisition to support language engineers in early language development phases. We describe an iterative approach in which DSL development benefits from formalized knowledge sources and information extraction from text supporting domain analysis and metamodel construction. We show how the acquired knowledge is used to guide language engineers and how knowledge acquisition is adapted according to modeling decisions.

1 Introduction and Problem Statement

Model-driven engineering (MDE) proposes systematic use of models as primary development artifacts for software system construction. These models describe different aspects of a system on a higher level of abstraction using particular modeling languages (e. g., UML or domain-specific languages (DSLs)). MDE aims at generating source code from the developed models and, as a consequence, reducing the effort at manually creating source code with programming languages. On the one hand, it has been shown that modeling and code generation increases productivity of software development projects [16], especially if developers can use ready-to-use domain-specific modeling languages and language workbenches [9]. On the other hand, if the available modeling languages do not suit the particular aspect to be modeled or they are not even available for a certain domain at all, additional initial language development effort neutralizes the productivity gain [15].

Creating new modeling languages requires expertise in language engineering, such as finding the right abstractions, creating metamodels and the correct usage of generalizations/specializations and aggregations. Assuming that language engineers have these competencies, these techniques are usually applied to different application areas and industrial sectors. These engineers are required to have detailed knowledge of the domain in order to create modeling languages.

Assembling domain knowledge is a time-consuming manual process (e.g. talking to domain experts and reading specific documentation).

As far as modeling techniques and programming are concerned, current state-of-the-art language workbenches, such as Eclipse Modeling Project, Spoofax and MetaEdit+, offer a lot of support in terms of metalanguages, model editors, transformation engines and code generators to build domain-specific languages and domain-specific tools. With regard to the actual content and meaning of the abstract syntax of a language, very limited assistance is provided.

In particular, the research question we are addressing is *"How can the development of domain-specific modeling languages be improved by automated knowledge acquisition?"*. To answer this question we consider the following: (1) *Where does the required knowledge come from?* Semantic knowledge bases and ontologies are an important source and this knowledge can also be obtained from text. (2) *How can the necessary knowledge be acquired automatically?* We propose the construction of queries from terms and relations in a model and their execution against multiple knowledge bases. (3) *How can the acquired knowledge be used to improve modeling?* We suggest that guidance through domain visualization and checking for semantically incorrect content leads to better quality models. (4) *How does model evolution effect knowledge acquisition?* In each step of a modeling process guidance shall be adapted according to the changing content of a model.

Section 2 discusses existing work related to this proposal. Section 3 explains our iterative approach for domain knowledge acquisition and modeling guidance. Section 4 describes the current status of this work and concludes the paper.

2 Related Work

This research proposal works on the connection of several research areas. We summarize the most important works in the following categories:

Software Language Engineering. The relatively new field of software language engineering [17] deals with methodologies and tools for the design of new modeling languages. Typical tasks in language engineering are abstract and concrete syntax design and semantics specification [8]. Systematic development of DSLs is discussed by Mernik et al. [19]. They identify several patterns for each DSL development phase. Strembeck and Zdun [22] describe activities for a DSL engineering process. In our research work we focus on metamodel-based [3] abstract syntax modeling as one of the most important early-stage activities of language creation in which classes, attributes and relationships of the language are defined. Current state-of-the-art techniques of modeling language design are presented by Selic [21].

Ontologies in Model-Driven Engineering. The use of ontologies in model-driven engineering has been investigated from different perspectives. Ontological foundations for metamodeling and modeling languages have been analyzed by Guizzardi [12], Evermann et al. [6] and Henderson-Sellers [14] with regard to the relation between conceptualizations, metamodels and ontologies. Results of

the MOST project [27] show the advantages of bridging ontology languages and modeling languages under a common metamodel and offering reasoning services for structural and behavioral modeling. Tairas et al. [24] show how ontologies improve the domain analysis phase of DSL development. Also, the connection between ontologies and MDE has been studied vice versa. Gasevic et al. [11] describe how ontology engineering benefits from methods of model-driven development. We focus on exploiting existing formalized knowledge to find suitable terms and relations for the abstract syntax of domain-specific languages.

Model Extraction From Text. In recent years a lot of research has been carried out to automatically create ontologies and knowledge bases with information extraction methods [5]. The field of knowledge harvesting [28] aims at automatically creating large scale knowledge bases by extracting facts from semi-structured and unstructured natural language text. For example, the YAGO [23] and DBpedia [4] projects show that large ontologies with millions of facts can be created automatically with high accuracy.

In the software engineering domain, requirements engineering still heavily relies on text documents. Natural language analysis is used in this field to automatically create early software design artifacts from textual specifications. Important works in this area are based on semantic annotations and semantic role labeling [26], use case parsing [30] and extraction of conceptual models [25]. Natural language processing (NLP) is also being applied to create process models from text [10]. We believe that these activities and the increasing amount of formalized knowledge can significantly support the creation of modeling languages in terms of model quality and the reduction of the time needed to create them.

3 Proposed Solution and Contribution

This PhD thesis aims at developing methods and tools to guide language engineers during the early stages of the creation of domain-specific languages using automated knowledge acquisition. Figure 1 shows the concept of the proposed EXAMINE system (**Ex**tr**a**cting infor**m**ation for Software Language Eng**ine**ering). Our approach incorporates an iterative process with three steps. Each of the steps is associated with tools that support the activity and artifacts that are produced or consumed, respectively.

During **Model Refinement**, the language engineer develops the DSL and modifies its abstract syntax. He uses an existing *Language Workbench* to carry out his tasks. The objective of the EXAMINE system is to build services for modeling environments that provide modeling assistance. In order to provide this assistance, the necessary knowledge of the according domain has to be acquired based on the terms of a created model.

The goal of **Knowledge Acquisition** is to automatically query existing knowledge bases, such as ontologies and lexical databases for semantically related concepts by using the content of a model. In case the results are insufficient, the

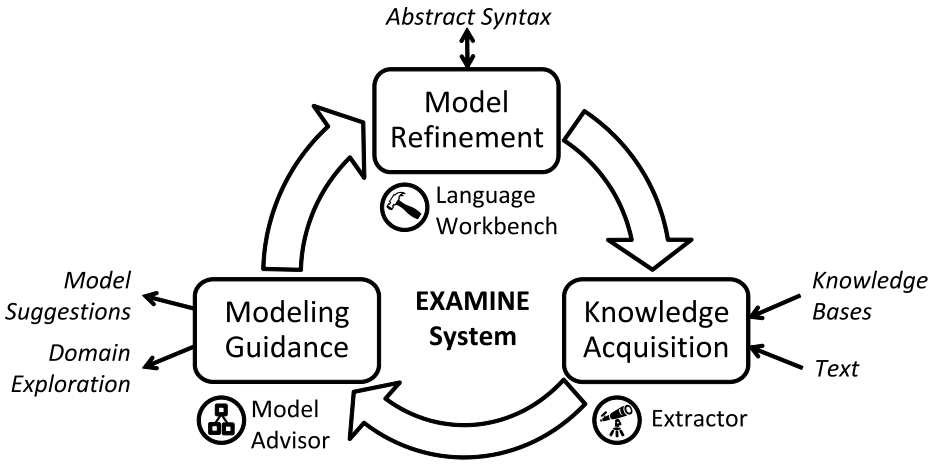

Fig. 1. Concept and Iterative Approach of the EXAMINE System

system tries to extract the required knowledge from text corpora. Querying and result integration is handled by the *Extractor*.

In the **Modeling Guidance** step, the obtained knowledge is processed to guide the engineers with the help of domain exploration and modeling advice. Furthermore, the *Model Advisor* also checks for semantic mismatches between the created model and the acquired knowledge.

In the transitions between the steps, the EXAMINE system **adapts** itself according to the modeling decisions made by the engineer (e.g. adding, changing and removing elements). The system also considers that the language engineer can directly use information from the acquired knowledge for the created model and that he can deselect information which he is not interested in.

In the following sections each of the steps is described in more detail.

3.1 Model Refinement

Current state-of-the-art modeling language design is based on language workbenches [9] that offer tools and languages for almost all DSL development phases. Nevertheless, for identifying and modeling domain concepts and relations, tool support is very limited [19].

In our approach, language engineers shall use existing language workbenches for abstract syntax design and make modeling decisions in a familiar environment. On top of that, services are offered that interact with the modeling environment in order to provide the engineers with domain knowledge and to give advice and corrections. These services can help in various places: thesauri for class name auto completion, graphical visualization of domain concepts and relations and drawing the user's attention to semantically incorrect parts of the model.

Furthermore, it shall be possible to integrate information from the acquired domain knowledge into the abstract syntax, thus explicitly describing the semantics of a modeling language. For example, the TwoUse approach [20] is a possible solution to connect models and domain knowledge contained in ontologies.

3.2 Knowledge Acquisition

In order to provide modeling assistance to language engineers, first, we propose automated querying of semantic knowledge bases. There exist several types of information sources that contain a variety of domain and conceptual knowledge: foundational ontologies (e.g. DOLCE, UFO), upper ontologies (e.g. SUMO), lexical databases (e.g. WordNet [7], FrameNet), manually created ontologies (e.g. Cyc), automatically constructed knowledge bases (e.g. YAGO [23], DBpedia [4]) and knowledge bases created by community effort (e.g. Freebase). The formalized knowledge has been significantly increased over the past years, also supported by the Linked Open Data Initiative.

Starting from a few initial terms contained in a created model, the *Extractor* component of the EXAMINE system constructs queries for knowledge bases (e.g. in SPARQL) to derive an initial top term ontology. In further modeling steps, the more the model is refined, the more specific the queries become. To achieve this, (1) a translation of the model content into queries is required, (2) the queries are executed against several sources, (3) the results are integrated (e.g. based on already existing owl:sameAs relations) and (4) an intermediate ontology specific to the information need is compiled.

In case existing knowledge bases do not provide enough information with regard to the targeted domain, we also address the acquisition of knowledge from unstructured information sources, such as natural language text corpora. Natural language processing relies on linguistic analysis pipelines in order to do tokenization, part-of-speech tagging, parsing, named entity recognition and information extraction tasks on top of that. In our approach, we are especially interested in deriving conceptual knowledge from text: concepts and different types of relations (i.e. hypernym/hyponym and meronym/holonym relations). For example, many approaches that learn taxonomic relations from text use Hearst patterns [13] which consider syntactic patterns between noun phrases. Another state-of-the-art mechanism is Open Information Extraction [5] that extracts any type of relation between entities.

3.3 Modeling Guidance

Once the required domain knowledge is obtained, the *Model Advisor* processes the results and assists the language engineer in the further development of a model. We propose the following kinds of assistance: (1) a graphical and navigable visualization of the domain concepts and relations, (2) suggestions on what should be included in the developed model, and (3) identification of possible semantic mismatches between the model and the acquired knowledge.

Graphical information visualization [18] is one method to provide exploratory access to domain knowledge. We intend to use a representation of the acquired domain knowledge that is familiar to the language engineer (e.g. generating class diagrams or semantic networks and hiding the URI-based subject-predicate-object statements of knowledge bases). The model advisor shall offer navigation by expanding and collapsing nodes in the graph. Providing suitable excerpts and highlighting information according to the model content is a challenge in that area.

During the modeling process the language engineer should receive suggestions on what he might include in the model by determining missing classes and relations (e.g. *"You created the classes 'patient' and 'doctor'. You may also add 'nurse' and 'hospital'."*). Of course, this example is a very simple way of making a suggestion. The goal of the model advisor is to develop more sophisticated methods for doing that. Giving this advice is similar to topic suggestion [29].

Finally, existing formalized domain knowledge can be used to detect semantically incorrect parts of a model (e.g. *'nurse' is a subclass of 'doctor'*). In a first step, we exploit WordNet's hypernym/hyponym and meronym/holonym relations to verify subclasses and aggregations in a model. In subsequent research we also aim at detecting incorrect associations, cardinalities and attributes.

3.4 Adaptation

Our approach considers, firstly, that knowledge acquisition and modeling guidance should be adjusted according to the evolving content of the models. For example, if an engineer deletes a class from his model it should not be suggested by the model advisor again and again. In a first step, a systematic catalog of basic model operations (such as adding, changing and removing a class or an association) and their impact on the iterative process will be defined.

Secondly, the EXAMINE system reacts to the amount of information contained in a model. If a model contains just a few terms the domain exploration should start with a domain overview. The more elements are added to the model, the more details can be acquired and the more specific queries can be executed.

Finally, integrating acquired knowledge directly into a model is considered as positive user feedback. Due to the fact that current state-of-the-art methods of knowledge acquisition (especially NLP) still lack high accuracy, knowledge acquisition can be iteratively improved through user feedback.

4 Current Status and Outlook

This research effort is still at an early stage. The idea of automatically providing guidance for language engineers using knowledge bases and ontologies was developed from the author's previous work in the area of using semantic technologies for model-based software integration [2] and development of domain-specific languages [1] and from the author's interest to work on the connection of different research areas.

The expected contributions from this PhD thesis are methods for automated domain knowledge acquisition and modeling guidance based on the acquired knowledge. The methods will be implemented by the *Extractor* and the *Model Advisor* component of the EXAMINE system as described in Section 3. The results of this research work shall enable language engineers to create models more efficiently and at higher quality.

The presented concept was developed in the context of the research project BIZWARE[1], a collaboration of two academic and eight industrial partners to investigate the potential of domain-specific languages and model-driven engineering for small and medium enterprises. The industrial partners are software companies working in different domains: healthcare, production/logistics, facility management and publishing.

Currently, our approach is analyzed by performing the iterative process manually for small domain models in the healthcare and facility management domain in collaboration with domain experts of our industrial partners. From that we expect to refine and improve the proposed method. We also survey the coverage of existing knowledge bases and lexical-semantic databases for developing these models, thus estimating the degree of automation that can be achieved.

References

1. Agt, H., Bauhoff, G., Kutsche, R.D., Milanovic, N.: Modeling and Analyzing Non-functional Properties to Support Software Integration. In: CAiSE 2011 Workshops. Springer-Verlag, Berlin, Heidelberg (2011)
2. Agt, H., Bauhoff, G., Kutsche, R.D., Milanovic, N., Widiker, J.: Semantic Annotation and Conflict Analysis for Information System Integration. In: Proceedings of the MDTPI at ECMFA (2010)
3. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Softw. 20, 36–41 (September 2003)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: 6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007). Springer-Verlag, Berlin, Heidelberg (2007)
5. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Commun. ACM 51, 68–74 (December 2008)
6. Evermann, J., Wand, Y.: Ontology based object-oriented domain modelling: fundamental concepts. Requir. Eng. 10, 146–160 (May 2005)
7. Fellbaum, C.: WordNet : An Electronic Lexical Database. The MIT Press, Cambridge, MA (1998)
8. Fowler, M.: Domain Specific Languages. Addison-Wesley Professional, Boston (2010)
9. Fowler, M.: Language Workbenches: The Killer-App for Domain Specific Languages? (2005), http://www.martinfowler.com/articles/languageWorkbench.html
10. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: CAiSE (2011)

[1] This work is partially supported by the Bundesministerium für Bildung und Forschung BMBF under grant number (Förderkennzeichen) 03WKBU01A.

11. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Architecture and Ontology Development. Springer-Verlag New York, Inc. (2006)
12. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Proceeding of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006. pp. 18–39. IOS Press, Amsterdam, The Netherlands (2007)
13. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics - Volume 2. COLING '92, Stroudsburg, PA, USA (1992)
14. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. J. Syst. Softw. 84, 301–313 (February 2011)
15. Hudak, P.: Modular Domain Specific Languages and Tools. In: Proceedings of the 5th International Conference on Software Reuse. ICSR '98, IEEE Computer Society, Washington, DC, USA (1998)
16. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Press (Mar 2008)
17. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley Longman Publishing Co., Inc., Boston (2009)
18. Mazza, R.: Introduction to Information Visualization. Springer Publishing Company, Incorporated, 1 edn. (2009)
19. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. 37, 316–344 (December 2005)
20. Parreiras, F.S., Staab, S.: Using ontologies with UML class-based modeling: The TwoUse approach. Data Knowl. Eng. 69(11), 1194–1207 (2010)
21. Selic, B.V.: The theory and practice of modern modeling language design for model-based software engineering. In: Proceedings of AOSD '11. pp. 53–54. ACM, New York, USA (2011)
22. Strembeck, M., Zdun, U.: An Approach for the Systematic Development of Domain-Specific Languages. Softw. Pract. Exper. 39, 1253–1292 (October 2009)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th international World Wide Web conference (WWW 2007). ACM Press, New York, NY, USA (2007)
24. Tairas, R., Mernik, M., Gray, J.: Using Ontologies in the Domain Analysis of Domain-Specific Languages. In: MoDELS Workshops. pp. 332–342. Springer-Verlag, Berlin, Heidelberg (2008)
25. Thonggoom, O., Song, I.Y., An, Y.: EIPW: A Knowledge-based Database Modeling Tool. In: CAiSE 2011 Workshops. Springer-Verlag, Berlin, Heidelberg (2011)
26. Tichy, W.F., Körner, S.J., Landhäußer, M.: Creating software models with semantic annotation. In: Proceedings of ESAIR '10. pp. 17–18. ACM, New York, USA (2010)
27. Walter, T., Parreiras, F.S., Staab, S., Ebert, J.: Joint Language and Domain Engineering. In: ECMFA. pp. 321–336 (2010)
28. Weikum, G., Theobald, M.: From information to knowledge: harvesting entities and relationships from web sources. In: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data. pp. 65–76. PODS '10, ACM, New York, NY, USA (2010)
29. West, R., Precup, D., Pineau, J.: Automatically suggesting topics for augmenting text documents. In: Proceedings of the 19th international conference on Information and knowledge management. CIKM '10, ACM, New York, USA (2010)
30. Yue, T., Ali, S., Briand, L.C.: Automated Transition from Use Cases to UML State Machines to Support State-Based Testing. In: ECMFA. pp. 115–131 (2011)