Marrying Ontologies and Model Driven Engineering Technical Spaces: The TwoUse Approach

Fernando Silva Parreiras

ISWeb — Information Systems and Semantic Web, Institute for Computer Science, University of Koblenz-Landau
Universitätsstrasse 1, Koblenz 56070, Germany
parreiras@uni-koblenz.de

Abstract. The Ph.D. proposal addresses challenges in composing metamodel-based modeling and OWL technologies. Although the two paradigms, UML-like models and OWL, and their related technological spaces seem closely related, in the state-of-the-art research and practice the two technologies are just beginning to converge and the relationship between the two is still under exploration. The proposal comprises different facets of the composition challenge, namely: (1) domain analysis of integrating both technical spaces; (2) the specification of a coherent framework for integrated use of both modeling approaches, comprising the benefits of UML-like models and OWL; Applications of the proposed framework to improve (3) Model Driven Engineering and (4) Ontology Engineering. Additionally, the PhD proposal enlightens the research methods to be used as well as the correspondent validation procedures.

1 Introduction

The Unified Modeling Language (UML) [1] and Web Ontology Language (OWL) [2] constitute modeling approaches with different strength and weaknesses that make them appropriate for use of specifying different aspects of software systems. In particular, OWL is well suited to specify classes using an expressive logical language with highly flexible and dynamic class membership, while UML is much more suitable for specifying not only static models including classes and associations, but also dynamic behavior.

Though MOF based metamodels [3] and UML profiles for OWL have been proposed in the past, an integrated usage of both modeling approaches in a coherent framework has been lacking so far. This PhD proposal unveils research problems involving the composition of these two paradigms and present research methods and validation techniques to assess the application of a novel framework integrating UML-like models and OWL.

* This work is supported by CAPES Brazil and EU STReP-216691 MOST.
** Under the supervision of Prof. Dr. Steffen Staab
Firstly, a characterization of metamodeling technical spaces and ontological technical spaces is required to recognize the features of the different paradigms and to elicit the requirements of an integrated framework (Problem 2.1). Consequently, the method of integration, the TwoUse approach, is specified (Problem 2.2) and conceptually verified against the requirements resulted from the first investigation.

Thereafter, shortcomings of canonical approaches in Model Driven Engineering (MDE), e.g., software design patterns, are addressed by applying the proposed framework with the aim at reducing complexity and improving reusability (Problem 2.3).

Concerning Ontology Engineering, issues addressing the gap in clarity and accessibility of languages that operate ontologies, e.g., ontology translation languages, are undertaken (Problem 2.4). The TwoUse framework is then used to support the development of platform independent models, aiming at improve maintainability and comprehensibility. The application of TwoUse is endorsed through experimental validations relying on case studies as observational method.

2 Problems and Goals

2.1 Requirements for Integrating Ontological and Metamodeling Technical Spaces

Problem. Over the last decade, the Web, AI and database communities have successfully investigated and promoted the use of ontologies as modeling and reasoning frameworks for the management of models and corresponding (Web) data.

While the focus of these communities is somewhat different, the following question arises: What are the scientific and technical results around ontologies, ontology languages and their corresponding reasoning technologies that can be used fruitfully in model-driven software engineering and vice versa?

Objectives. While investigating this problem, our goal is to analyze different integration approaches available in the literature. The result of such analysis is a feature model, preliminarily discussed in [4].

The feature model reveals the different possible choices for an integrated approach of metamodeling and ontologies technical spaces and can also be used as a taxonomy to categorize the existing approaches published in the literature. Furthermore, the classification allows for eliciting requirements for an composed approach, used to validated the results of Problem 2.2.

Initial patterns we find in our own work as well as in related works shows that next to existing technical spaces of established meta modeling frameworks, new technical spaces are positioned that either enrich or exploit software engineering capabilities by or for ontology technologies.
Method and Validation. We undertake a domain analysis over approaches integrating metamodeling technical spaces and ontological technical spaces found in literature. Domain analysis is concerned with analyzing and modeling the variabilities and commonalities of systems or concepts in a domain [5].

The research result is a descriptive model characterized by a feature model for the problem area. The feature model is then validated using formal semantics and reasoning support of OWL as describe in [6].

2.2 The TwoUse Approach

Problem. UML class-based modeling and OWL comprise constituents that are similar in many respects, like: classes, associations, properties, packages, types, generalization and instances [7]. However, both approaches have their advantages and disadvantages. UML provides means to express dynamic behavior, whereas OWL does not. OWL offers support for inferring generalization and specialization between classes as well as class membership of objects based on the constraints imposed on the properties of class definitions, whereas UML class diagrams do not count on automated reasoning for dynamic specialization/generalization of classes and class memberships.

UML class diagrams can be translated into Description Logic [8] to have a model-theoretic semantics, which enables the usage of automated reasoning over UML class-based modeling. However, when translating an Description Logic knowledge base back into UML Class Diagram, the UML encoding of assertions involving constructs like union, complement and equivalent class may look awkward [8].

Therefore, combining OWL with UML class-based modeling enhances Model Driven Engineering tasks by (i) providing more intuitive constructs, since OWL is a logical class definition language, and by (ii) providing additional constructs to UML class-based Modeling. Moreover, OWL can describe and work with incomplete knowledge, which may be required in contexts where complete information is not available.

Contemporary software development should make use of the benefits of both approaches (UML and OWL) to overcome their restrictions. This consideration led us to the following challenge: How can we develop and denote models that benefit from the advantages of the two modeling paradigms?

While mappings between modeling paradigms have been established a while ago (e.g., [7]), the task of an integrated language for UML and OWL has not been dealt with before. The challenge of this task arises from the large number of differing properties germane to each of the two modeling paradigms (cf. [4] for an analysis) making the integration of UML class-based models and OWL difficult.

Objectives. To overcome such a challenge, our aim is to provide a framework comprising the following building blocks: (i), an integration of the MOF-based metamodels for EMOF, OCL and OWL, (iii), the specification of model library
referring to OWL reasoning (using OCL-like expressions), (iv), the definition of a joint profile for denoting hybrid models, and, (v), closure operations that determine whether entities must diffuse from one paradigm into the other and, hence, how they are to be mapped onto metamodels. Together, these building blocks constitute our novel approach to Transform and Weave OWL and UML in Software Engineering — TwoUse. An outline of such framework is explored in [9]. We propose an OCL extension as pivot metamodel, allowing to plug different metamodels that use OCL, extending the range of application.

**Method and Validation.** The feature model resulting of Problem 2.1 is used as requirements to undertake Problem 2.2. A case study will help to improve the framework for further applications. The research result is a technique involving the integration of metamodeling and ontologies — The TwoUse approach. The TwoUse approach will be firstly validated against the requirements resulting from Problem 2.1, following by case study requirements. Such validation is helpful to detect, develop, refine frames of reference.

In the case study, we apply reverse engineering to the software developed internally in the study group, KAT [10], in order to have UML models. Since TwoUse aims at improving maintainability and at reducing complexity, these UML models will be compared with TwoUse models using software quality metrics for such attributes.

**2.3 On Ontologies into Model Driven Engineering**

**Problem.** Design patterns provide elaborated, best practice solutions for commonly occurring problems in software development [11]. However, in [12] we have observed that, in software design patterns managing variants, the decision of what variant to choose is delegated task to client classes.

Hence, the question arises of how the selection of specific classes could be determined using only their descriptions rather than by weaving the descriptions into constructs like client classes.

Solutions based on patterns like Strategy embed the treatment of variants into clients’ code at various locations, leading to an unnecessary tight coupling of classes. This issue has already been identified by [13] as a drawback of pattern-based solutions e.g. when discussing the Strategy Pattern and its combination with the Abstract Factory Pattern.

**Objectives.** To remedy this problem, we work towards identifying patterns able to decouple class selection from class definition by exploiting OWL-DL modeling and reasoning. Thus, we explore Design Patterns that includes OWL-DL modeling and that leads us to a minor, but powerful variation of existing practices.

1 [http://isweb.uni-koblenz.de/Projects/twouse](http://isweb.uni-koblenz.de/Projects/twouse)
One of these new design patterns was previously described in [12]. The Selector Design Pattern is applied to overcome the drawback of delegation presented by the Strategy Pattern or by the Abstract Factory Pattern, when both Strategy and Abstract Factory are used in conjunction.

For the dissertation, we aspire to recognize slots in software design patterns where an ontology-based approach improves the software quality. To realize the new design patterns, we apply our TwoUse approach in order to allow for joint UML-like modeling and OWL-DL modeling.

Method and Validation. Using the TwoUse framework, we examine the literature of design pattern identifying patterns that could benefit from an integrated approach and propose new patterns. Since it is hard to measure quality improvements of design patterns, we intend to show that ontology-based design patterns make it possible to do something that was impossible heretofore. Hence, we address shortcomings design patterns but still fully preserve their functionality.

2.4 On Model Driven Engineering into Ontologies

Problem. Much attention has been given to ontology mapping. However, as these tasks get more complex, current approaches fail to provide clarity and accessibility to the modelers who need to see and understand the semantic as well as the lexical/syntactic part of the specification. The modelers usually can modify them only in a very intricate and disintegrated manner, drawing his attention away from the core task proper down into the diverging technical details of the solution used.

From this scenario, the problem of supporting generative techniques in ontology field like ontology mapping emerges, adding expressiveness without going into platform specifics, i.e, how to fill the abstraction gap between specification languages and programming languages.

Filling the gap in ontology translation domain between languages for ontology mapping and general purpose programming languages helps to improve productivity, since modelers will not have to be aware of platform-specific details. Moreover, maintenance and traceability would be facilitated because mapping knowledge is not longer embedded in source code of programming languages anymore.

Objectives. We propose a representation approach for generative specification of ontology tasks based on model-driven engineering (MDE). In order to reconcile semantic reasoning with idiosyncratic lexical and syntactic translations, we integrate the different layers into a representation based on a joint meta model. The joint meta model allows for to specify and represent semantic operations as well as OCL constraints for specifying lexical and syntactic translations.

A preliminary effort into this direction is the language for Model Driven Specification of Ontology Translations presented in [14]. Such paper presents an ongoing solution for ontology translation specification that intends to be more
expressive than ontology mapping languages and less complex and granular than programming languages (MBOTL).

Methods and Validation For Ontology Engineering, we apply twoUse to propose generative approaches, i.e., ontology translation specification or Ontology API generation. Since we claim that TwoUse improves comprehensibility and productivity, we assess the size of the produced TwoUse models from the perspective of the user against current approaches. As domain for ontology translation we use two ontologies of bibliographic references from the test library of the Ontology Alignment Evaluation Initiative (OAEI) [15].

References