Delta-P: Model Comparison Using Semantic Web Standards

Martín Soto
Fraunhofer Institute for Experimental Software Engineering
Kaiserslautern, Germany
soto@iese.fraunhofer.de

Abstract

The Delta-P [1] approach relies on Semantic Web notations and techniques to perform comparison of potentially complex models. The key feature of the approach is to allow for flexible adaptation, by using queries to detect changes relevant to different schemata. We briefly describe the approach and its initial implementation (Evolyzer) and discuss the direction of our future work.

1. Introduction

After more than 30 years of continuous progress, software configuration management offers us a wide selection of powerful and highly reliable software versioning tools. However, it strikes one as surprising that despite the advanced state of the discipline, and given the widespread use of modeling in modern software development, no complete solution is available that supports versioning and collaborative work on software models in an efficient and reliable way.

This situation has motivated us to work on Delta-P, an approach for model comparison that processes models at a level of abstraction that is appropriate for humans to work with. In so doing, we have found a number of notations and techniques originally developed for the Semantic Web to be especially useful for the task. In this paper, we briefly describe our model comparison approach. This approach was originally conceived in the context of software process modeling, but we consider it general enough to be adequate for comparing models of other types. Right now, we are taking the first steps to extend it to generic UML models.

2. The Delta-P Model Comparison Approach

By making use of notations and techniques drawn from the Semantic Web, we have developed a comparison approach that achieves the following goals:

- Operate on a variety of schemata. New schemata can be supported with relatively little effort.
- Be flexible about the changes that are recognized and how they are displayed.
- Allow for easily specifying change types that are specific to a particular schema or even to a particular application.
- Be tolerant to schema evolution, by allowing the comparison of model instances that correspond to different versions of a schema (this sort of comparison requires additional effort, though.)

2.1. General Approach

More specifically, our approach can be decomposed in the following steps:

1. Convert the model instances to a normalized triple-based notation. We use the RDF notation for this purpose.
2. Perform an identity-based comparison of the the resulting RDF models, to produce a so-called *comparison model.* This model corresponds roughly to the results of a simple comparison based on unique identifiers.
3. Find relevant changes by recognizing the corresponding change patterns in the comparison model.

2.2. The Resource Description Framework

The Resource Description Framework (RDF) [3] was originally defined as a generic notation for describing resources in the World Wide Web. In actuality, however, it can be seen as a highly generic notation for representing arbitrary sets of objects and their relationships.

RDF models are sets of so-called *statements,* each of which makes an assertion about the world. Statements are always composed of three parts, called the *subject*, the *predicate*, and the *object*. The subject is a unique identifier of an entity. The predicate identifies an attribute of this entity or a relation the entity has with another entity. Depending on the predicate, the object contains either the value of the attribute or the identifier of the entity pointed to by the relation. Although a relatively high number of statements is necessary to cover all details of a model expressed in RDF, the final representation is highly uniform. Also, in our experience, converting models from different notations (e.g, XML) and schemata to RDF is usually a straightforward process.

2.3. The Comparison Model

Since RDF statements refer to the modeled entities by using unique identifiers, two versions of a

model expressed as RDF can be efficiently compared to determine which statements are common to both versions, and which are exclusive to only one of them. For the comparison of two versions A and B, we express the results of this comparison by constructing the union set of A and B, and *marking* the statements in it as belonging to one of three subsets: statements *only* in A, statements *only* in B, and statements common to A and B. The resulting set, with the statements marked this way, is called the *comparison model* of A and B. One central aspect of the comparison model is that it is also a valid RDF model. This is possible through a theoretical device called *RDF reification* that allows for statements to make assertions about other statements.

2.4. Identification of Specific Changes

In most practical situations, comparison models are too large and detailed to be useful for direct human analysis. The main problem lies in the fact that most changes that human beings perceive as a unit correspond potentially to many added and/or deleted statements. For this reason, a mechanism that makes it possible to automatically identify changes at a higher level of abstraction is necessary. In our approach, this mechanism is provided by the SPARQL RDF query language [4]. Our experience shows that SPARQL queries can concisely describe a wide variety of interesting changes by identifying patterns of deleted and added statements.

3. Implementation

Our current implementation of model comparison was especially designed to work on large software process models, such as the German V-Modell [5] and its variants. Nevertheless, since the comparison kernel implements a significant portion of the RDF specification support for other types of models can be added with relatively small effort.

Until now, we have mainly tested our implementation with various process models, including many versions of the V-Modell (both standard releases and customized versions.) Converted to RDF, the latest released version of the V-Modell (1.2) contains over 13.000 statements that describe over 2000 different entities. Most interesting comparison queries on models of this size run in under 5 seconds on a modern PC.

4. Related Work

Several other research efforts are concerned in one way or another with comparing model variants syntactically and providing an adequate representation for the resulting differences. For example, [6] and [7] deal with comparing various types of UML models. The focus of these works is to supporting software development in the context of the Model Driven Architecture. To our knowledge, they do not attempt to describe changes generically, as our approach does.

An extensive base of theoretical work is available from generic graph comparison research (see [2]). We consider this work generally complementary to ours, and appropriate for a different set of problems.

5. Conclusions and Outlook

Delta-P can only work properly on models that assign unique, stable identifiers to relevant model entities. Although, in theory, this characteristic may appear too restrictive, most actual modeling tools support the feature, a fact that makes our approach useful in a wide variety of practical applications. Our work with the V-Modell has shown that our approach is viable and efficient even on relatively complex models.

Having a solid comparison system is only the first step towards a complete model versioning environment. Particularly, good support for collaborative work on models requires also a complementary merge algorithm. We have done initial work on extending our algorithms to support comparisons with common ancestor. Our experience so far, shows that it is possible to also use queries to specify a variety of relevant merge conflicts. This way, a merge is done by simply uniting the models and running a number of queries that identify possible conflicts.

Additional ongoing work is concentrated in turning our database based system into a full flexed model versioning system. An initial database containing about 600 development versions of the V-Modell, shows us that the approach is viable from a performance point of view. We are looking forward to building an operational prototype versioning system in the following months.

Acknowledgments. This work was supported in part by the German Federal Ministry of Education and Research (V-Bench Project, No. 01| SE 11 A).

6. References

[1] Soto, M., Münch, J.: Process Model Difference Analysis for Supporting Process Evolution. In: Proceedings of the 13th European Conference in Software Process Improvement, EuroSPI 2006. Springer LNCS 4257 (2006)

[2] Kobler, J., Schöning, U., Toran, J.: The Graph Isomorphism Problem: Its Structural Complexity. Birkhäuser (1993)

[3] Manola, F., Miller, E. (eds.): RDF Primer. W3C Recommendation, available from http://www.w3.org/TR/rdf-primer/ (2004) (last checked 2006-03-22)

[4] Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Working Draft, available from http://www.w3.org/TR/rdf-sparql-query/ (2006) (last checked 2006-10-22)

[5] V-Modell XT. Available from http://www.v-modell.iabg.de/ (last checked 2007-02-16).

[6] Alanen, M., Porres, I.: Difference and Union of Models. In: Proceedings of the UML Conference, LNCS 2863 Produktlinien. Springer-Verlag (2003) 2-17

[7] Lin, Y., Zhang, J., Gray, J.: Model Comparison: A Key Challenge for Transformation Testing and Version Control in Model Driven Software Development. In: OOPSLA Workshop on Best Practices for Model-Driven Software Development, Vancouver (2004)