Metamodel Evolution through Metamodel Inference

Qichao Liu

Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, Alabama, USA

qichao@cis.uab.edu

Abstract

Serving as the schema of models, a metamodel defines the abstract syntax of models and the interrelationships between model elements. Model instances are often inaccessible due to metamodel evolution or the metamodel becoming lost. This poster describes our research recovering a metamodel from model instances to support metamodel driven evolution.

Categories and Subject Descriptors I.6.5 [**Simulation and modeling**]: Model Development

General Terms Algorithms, Design, Languages.

Keywords model-driven engineering; domain-specific modeling; grammar inference; metamodel

1. Background and Motivation

Model-driven engineering (MDE) is considered an alternative to traditional code-based software development due to its potential to increase software productivity and quality [1]. In MDE, a user defines a metamodel to represent a schema definition of the syntax and static semantics of a model. A programming language depends on a grammar similar to how a model depends on a metamodel. A metamodel serves as the grammar of a model and both grammars and metamodels represent a schema that defines the syntax of a language.

Under most conditions the schema needs to evolve to address new features resulting in previous instances being orphaned from the new definition. Lämmel and Verhoef [2] [3] addressed the schema evolution problem in the area of programming languages and their approach is to recover grammars from grammar-related artifacts and create a parser for language instances. In MDE, modeling language designers often need to modify metamodels even if they have created many instance models that depend on a previous metamodel. As a result, those former models that depend on the previous metamodel could not be interpreted and used by the modified metamodel which is a waste of model instances in most cases. The common approach toward addressing the metamodel evolution problem is to create model transformations that update existing model instances to be interpretable by the latest metamodel. This work requires that both the old and new metamodel are available for mapping and comparison. However, users usually make changes to a metamodel without restoring the old definition or more generally a metamodel may be lost due to the version change or hard disk crash. Without the old metamodel it is very hard to perform model transformation.

This research addresses the metamodel evolution problem in MDE through metamodel recovery from model instances so that users could perform model transformation with both the old and evolved metamodel and enable the latest metamodel to interpret existing model instances.

SPLASH'10 October 17-21, 2010, Reno/Tahoe, Nevada, USA.
ACM 978-1-4503-0240-1/10/10.

2. Limitations of Related Work

Sprinkle and Karsai proposed to update the domain models created by a domain-specific visual language (DSVL) using graph-rewriting (GR) techniques in [4]. Their approach could be considered as rewriting a domain model to another one as required by the new DSVL to make an old model evolve to conform to a new DSVL. However, the domain models created by a DSVL is represented using a graph structure and could not be applied to general domain-specific modeling environment.

There is a variety of work that has been done or is being conducted in the area of grammar inference. Traditionally, schema evolution has been related with the problem of database schema evolution to adapt to changes in the modeled reality. Grammar inference has been applied to DTD and XML Schema extraction from XML documents. For example XTRACT [5] can induce the DTD from a set of XML documents using its regular grammar induction engine. Our research is concentrated on recovering a metamodel from model instances contained in XML documents to address the metamodel schema evolution problem.

Favre [6] presented a generic metamodel-driven process, CacOphoNy that integrates software architecture and MDE. Although their work includes metamodel recovery, the approach requires manual intervention. Our research also incorporates MDE through the effort toward the metamodel recovery problem and our process is semi-automatic beyond the information from the model instances. Javed et al. [7] presented work on metamodel recovery using grammar inference which addressed the more general problem of metamodels lost due to disk crash. The work is greatly limited to a simple metamodel and is also platform dependent. Our research provides a general approach aiming at solving the problem of metamodel evolution.

3. Solution Approach: MRMI

The Metamodel Recovery from Model Instances (MRMI) research described in this poster is the first step toward addressing the metamodel evolution problem. The idea behind metamodel inference is to analyze the characteristics exhibited in the model

instances and infer a metamodel. As a result, we have implemented EMARS (Extended MetAmodel Recovery System) [8]. The modeling tool used in EMARS is GME [9] which could export a model instance into XML file and modeling concepts like "model" and "atom" are established as nodes in XML.

The metamodel inference begins with reading in a set of instance models in XML as input. An XSLT translator performs the XSL (Extensible Stylesheet) Transformation [10] on XML files. XSLT uses the XML Path Language (XPath) [11] to retrieve values of interest at specific nodes in an XML document. As the output of the XSLT translator, a domain-specific language (DSL) called model representation language (MRL) containing the essence of model instances is produced. MRL is composed of components of the model instance in a form that could be used by the metamodel inference process. The following is an example for 'model' definition in MRL. As such, a mapping is constructed from model instances in XML to MRL.

```
model folderX::X
{
  submodelsY,Y;
  fields fieldX1, fieldX2;
  connections;
}
```

The MRL is then loaded into the LISA language development environment [12]. Our metamodel inference algorithm could infer the corresponding XML representation for MRL having an inferred metamodel as the output. This inferred metamodel could be loaded back into the modeling tool (e.g., GME) to view the previous model instances. Figure 1 illustrates the MRL example in GME.

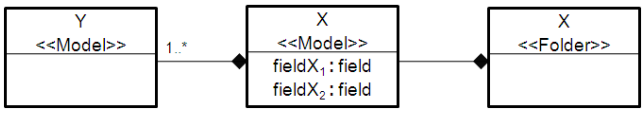

Figure 1. MRL example in GME

4. Results and Contribution

We have tested MRMI successfully on various simple domains with a small number of elements and our inference is almost exactly the same as the original metamodel. We have also tested the approach with some complex domains like ESML [13] with multiple viewpoints. Due to the large number of metamodeling elements used in ESML, the quality of our inference greatly relies on the quality of model instances used to do the inference. We applied MRMI on three instances created by the original ESML metamodel and over 90% of the metamodeling elements of the original are inferred accurately in our inference. Additionally, MRMI currently can infer accurate generalization of elements sharing common features and the cardinality is also inferred as being the same as the original. Detailed experimental results will be presented in the poster.

Our ultimate goal is to infer a metamodel exactly the same as the original which could be used to view model instances just like the original. However, the semantics contained in a metamodel could not be inferred from static model instances except the containment cardinality. Likewise, inference of OCL constraints is not possible with the proposed technique. OCL (Object Constraint Language) is used to describe domain semantics and can only be captured by dynamic class diagrams. Without OCL, the inferred metamodel may reject model instances legally created by the original metamodel. The related work will be addressed in our future work.

The contribution of this paper is to present MRMI for metamodel recovery from model instances. A host of technologies such as XSLT, LISA and metamodel inference algorithm are utilized to solve the problem of inaccessible existing model instances due to metamodel evolution or the metamodel becoming lost. MRMI is the most succesful work in applying grammar inference in the field of MDE and also serves as our first step towards addressing the metamodel evolution problem.

Acknowledgments

This work is supported in part by NSF award CCF-0811630.

References

[1] Schmidt, D. C.: Guest Editor's Introduction - Model-Driven Engineering. IEEE Computer, vol. 39, no. 2, Feb. 2006, pp. 25-31.

[2] Lämmel, R., Verhoef, C.: Semi-automatic grammar recovery. Software-Practice & Experience, vol. 31, no. 15, Dec. 2001, pp.1395-1448.

[3] Lämmel, R., Verhoef, C.: Cracking the 500 language problem. IEEE Software, vol. 18, no. 6, Dec. 2001, pp.78-88.

[4] Sprinkle, J., Karsai, G.: A domain-specific visual language for domain model evolution. Journal of Visual Languages and Computing, vol. 15, no. 3-4, Aug. 2004, pp. 291-307.

[5] Garofalakis, M. N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K.: XTRACT - A system for extracting document type descriptors from XML documents. In Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, Dallas Texas, USA, May. 2000, pp. 165-176.

[6] Favre, J-M.: CacOphoNy - Metamodel driven architecture reconstruction. In Proceedings of the 11th Working Conference on Reverse Engineering, Nov. 2004, pp. 204-213.

[7] Javed, F., Mernik, M., Gray, J., Bryant, B.: MARS - A metamodel recovery system using grammar inference. Information and Software Technology, vol. 50, no. 9-10, Aug. 2008, pp.948-968.

[8] Liu, Q., Bryant, B.R., Mernik, M.: Metamodel recovery from multi-tiered domains using extended MARS. In Proceedings of the 34th Annual International Computer Software and Applications Conference, Seoul, South Korea, Jul. 2010, pp. 279-288.

[9] The Generic Modeling Environment, http://www.isis.vanderbilt.edu.

[10] Clark, J.: XSL Transformations (XSLT) (Version 1). W3C Technical Report, Nov. 1999, http://www.w3.org/TR/1999/REC-xslt-19991116.

[11] Clark, J., DeRose, S.: XML path language (XPath) (Version 1.0). W3C Technical Report, Nov. 1999, http://www.w3.org/TR/1999/REC-xpath-19991116.

[12] Mernik, M., Lenič, M., Avdičaušević, E., Žumer V.: LISA - An interactive environment for programming language development. In Proceedings of the 11th International Conference on Compiler Construction, Apr. 2002, pp. 1-4.

[13] Karsai, G., Neema, S., Sharp, D.: Model-driven architecture for embedded software - A synopsis and an example. Science of Computer Programming, vol. 73, no. 1, Sep. 2008, pp.26-38.