

Proposal for Bachelor Thesis

08 November 2014

William Bombardelli da Silva

Towards Synchronizing Relations Between Artifacts in the Java Technological Space

Introduction

This document intends to describe the proposal for bachelor thesis by William Bombardelli da Silva, student of Informatik Bachelor in the Technische Universität Berlin, student number 364927.

The goal of this bachelor thesis is to report the current state of research and to develop new contributions both theoretical and practical. By being so, the thesis aims to investigate a problem of software model synchronization in the context of Model-driven Engineering. More specifically the problem of defining relations between common software meta-models, in order to maintain their models synchronized is to be tackled.

Key-words

Model Synchronization, Iterative Model Transformation, Model Transformation, Model Evolution, Model Co-evolution, Model-driven Engineering, Software Engineering.

Theme Description

Recent techniques of software engineering have been using the concept of software models in the construction of software systems. According to [2] "*Models are system abstractions that allow developers and other stakeholders to effectively address concerns, such as answering a question about the system or effecting a change*". By defining a model as a system abstraction, it becomes clear, that a software system might have several models abstracted from it, each one representing certain aspects of the whole system. These models also have relations between them, in the sense that they all are supposed to describe the actual system consistently by not presenting logical contradictions. Here examples of models are *UML class diagram*, *Use Cases*, or even the source-code itself.

The constant evolving nature of current large-scale software systems causes its models to be constantly changed. But in order to maintain this whole network of interconnected models consistent the changes have to be forwarded through the network, i.e. all models have to be synchronized. Suppose one have a *UML Class Diagram*, a series of *UML Sequence Diagrams* and the source-code. If a method has its name changed in the class diagram, all occurrences of this method has to have its name updated in the sequence diagrams and in the source-code. It turns out though, that neither a model synchronization tool comprising the most common meta-models used nowadays has been developed nor are clearly defined relation definitions between them available on literature.

The goal of this bachelor thesis is therefore first to choose software meta-models commonly used in current Java object oriented software systems, defining the meta-model definitions; second to define (or write) the relations between the chosen meta-models; and last, to showcase how these artifacts can be synchronized in some representative cases.

In the end of this thesis development it is expected the creation of a report comprising the difficulties found and the next challenges for the problem. The meta-models used might be narrowed to *MOF* compliant (see [13]) meta-models.

Motivation

The lack of a functional model synchronization tool integrating a broad range of meta-models used currently in software engineering is the main motivation for this thesis. Although an expressive effort has been made by the academic community to create solutions for the problem of model synchronization, no study known by us presents an effective tool, that could be used extensively in practice. We believe though, that the contribution of this thesis can be useful for the creation of such a tool.

Another motivation for this thesis is the lack of relation definitions in current literature for extensively used meta-models in industry. Examples of these relations include relations between *UML Class and Sequence Diagrams* and *Java Code*, between *Use Cases* and *Requirements Diagrams*, between *OCL contracts (used in design by contract methods)* and *Unit Tests*, among others. It means, that the success of this thesis might bring the contribution towards the definition of a network of interconnected meta-models useful to both research and industry community. Therefore the availability of such a network might finally allow the extensive use of Model-driven Engineering in practice – helping bridging the gap between system abstractions and their concrete form – and the further development of more sophisticated model synchronization methods. It is worth to note also, that the contribution of this thesis might help enhancing the quality of current software construction and therefore lessening the number of software problems and errors, what is an endemic problem nowadays, by fomenting the wide application of Model-Driven Software Development.

State of Current Research

A research road-map for model synchronization found in France and Rumpe [5] gives an overview on the realm, and an interesting point of view about the challenges. In Mens and Van Gorp [11] a taxonomy for model transformation is proposed, what helps to carry more precise analysis. In Czarnecki and Helsen [2] a survey was driven and a framework for classification of model transformation approaches was presented. In Diskin et al [4] a taxonomy for a network of models is presented and in Diskin [3] a theoretical algebraic basis is proposed.

Between the attempts to build a model synchronization tool are the *ATL Eclipse Plug-in* [10], which uses the *Atlas Transformation Language to code the relations between models*; the *Medini QVT*¹, which claims to implement the *Query/View/Transformation Language to code the relations*; and the *FUJABA* [12], in which relations are coded using Triple Graph Grammars. Other publications aim to solve specific problems, like the ones in Hermann et al [8], Xiong et al [16], Giese and Wagner [7], Ivkovic and Kontogiannis [9], or Song et al [14], where advanced algorithms for bidirectional synchronization have been proposed. But none of the above have accomplished to develop an extensible and efficient tool capable of integrating the most used types of meta-models.

Nevertheless, recently Käfer et al have published their work on the *CoWolf Framework*, which is "an extensible framework for model evolution and co-evolution management" [6]. Although the number of supported meta-models is rather small, the extensibility claimed by them and the brief analysis we have made indicate us, that it is a promising option. The *CoWolf* is built as a Eclipse plug-in, that employs other Eclipse plug-ins, all of them supported by the community; uses *EMF* [15] to code the meta-models; and uses *Henshin* [1] to code their relations.

Additionally, one can judge by the date of publication of these works, that the topic of model synchronization is extremely active and is actually the edge of current academic research, what motivates even more the development of this thesis.

¹<http://projects.ikv.de/qvt>

Concepts Definition

Below is a list of necessary basic concepts, that will be used throughout this document.

Model: The definition for software model used is: “Models are system abstractions that allow developers and other stakeholders to effectively address concerns, such as answering a question about the system or effecting a change” [2] Examples of models, according to this definition, are *UML diagrams*, *OCLE expressions*, *relational database diagrams*, or even source-code.

Modeling Language: For the scope considered here, modeling language is broadly defined as a language used to create models, that can be textual or graphical and may occur in several paradigms. Among them, functional, declarative or operational paradigms.

Model Relation: Model relation here is defined abstractly as every relationship or constraint possible to happen between one source model and one target model.

Model Transformation: Model transformation can be viewed as common data transformation – very common in computer science – with the specificity of dealing with models [2]. In these terms, one can see model transformation from several points of view, for example as a sequence of operations/modifications over one model; or as a function, whose inputs relate to initial models and the output reflect the updated models.

Model Synchronization: The goal of model synchronization is to maintain all relations between the models of a system consistent/correct as updates are performed over them [3]. Model synchronization may be seen as a procedure, a series of model transformations; as a function, which output – e.g. consistent updated set of models – is determined by the inputs – e.g. consistent set of models plus transformations to be executed; or even as a relation between set of models. As model synchronization is a relative new field of study, no fixed definition or approach is consensus between researchers, what brings up a variety of possible strategies and conjectures to solve the problem.

Network of Models: A network of models of a system S is an undirected graph $G = (V, E)$, whereas each vertex $v_i \in V$ represents a unique model i of S , and an edge (v_i, v_j) exists if, and only if there is a relation defined between both models $i, j \in S$.

Problem Definition

The problem tackled in this thesis is first to identify currently commonly used meta-models in the Java technological space. Second to capture common relations between these meta-models. And third to showcase the synchronization of such relations in a tool.

An example of a possible network of meta-models created by the end of this work is supposed to look basically like the figure 1:

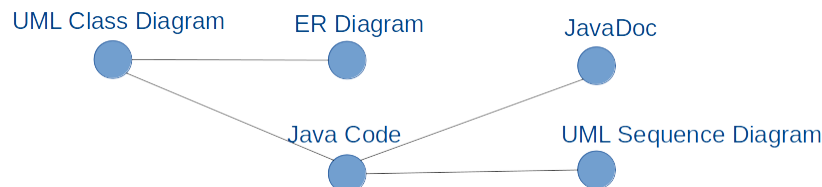


Figure 1: Example of a possible network of meta-models to be constructed. Like stated in the text, the meta-models are chosen accordingly basically to the frequency they are used currently in the Java technological space.

In the first moment the meta-models comprised in the network are to be defined, this is done through an extensive state-of-art research. So for example, in this phase the choice of applied meta-models (i.e. *UML Class Diagram* and *Java Code*) will be done and their definitions will be written. Possibly, simplifications

on the meta-models shall be done, so that they fit the necessities of our scope, and can be made using the *EMF* standard [15].

Later on, given the defined meta-models (this is the vertices of the network), their relations (this is the edges of the network) can be written, what can be made with Triple Graph Grammars. So for example, in this phase the inherent relation between the *UML Class Diagram* meta-model and the *Java Code* meta-model will be written. Analogously, the relations between *Java Code*, *JavaDoc*, *UML Sequence Diagram* and other meta-models of the Java technological space can be also defined. Some of these relations might be found in current research literature, others might have to be invented during the development of this thesis.

After having this network of meta-models ready, creating an actual network of more concrete models of any software system is trivial, thus study cases and example networks can be developed.

The next step is then taken, and the synchronization of the created network of models is showed using a specific tool or method, which are to be chosen during the development of the thesis, so that the necessities can be accordingly fulfilled.

In the end the result might be a network of meta-models useful in practice, in benchmarks and further researches as well. The showcase created will also be available as a possibly very useful tool. Furthermore, the report of this thesis recording the difficulties and experiences found during the work process and a possible analysis and discussion of future development is also supposed to be a legacy.

Possible Difficulties

Possible difficulties that can be found throughout the development of this thesis include:

- The definition of the meta-models in the network of meta-models might be complicated and require relatively extensive work. For example, the definition of the *Java Code* meta-model can be very broad and thus a narrowing of it is likely to be necessary.
- Some relations of the network might be difficult or complicated to build, requiring much time, so that it is not possible to include it in the scope of this thesis.
- Learning the Eclipse plug-ins like *EMF* might represent some effort.
- Finding the best way to showcase the synchronization of the model network built might be the greatest challenge of this research and surely represents a danger.

Time Schedule

Duration	Start and End Dates	Activities
2 Weeks	13/10/2015 to 26/10/2015	Initial research; definition of theme; finding of literature
1 Weeks	27/10/2015 to 02/11/2015	Detailed research; write of proposal; definition of scope
2 Weeks	03/11/2015 to 16/11/2015	Deepening in the theme; sketch of the development
3 Weeks	17/11/2015 to 07/12/2015	Analysis; design; pre-development phase; review; start of the writing
5 Weeks	08/12/2015 to 11/01/2016	Development; testing, validation and verification; review; writing
2 Weeks	12/01/2016 to 25/01/2016	Validation and verification; review; writing
3 Weeks	26/01/2016 to 15/02/2016	Finalization of writing; review
1 Weeks	16/02/2016 to 22/02/2016	Preparation of presentation
1 Weeks	23/02/2016 to 29/02/2016	Final review

Table 1: Plan for the researching, developing and writing of the bachelor thesis. This schedule is organized in weeks, whereas each week has its respective activities planned.

References

- [1] Thorsten Arendt, Enrico Biermann, Stefan Jurack, Christian Krause, and Gabriele Taentzer. Henshin: advanced concepts and tools for in-place emf model transformations. In *Model Driven Engineering Languages and Systems*, pages 121–135. Springer, 2010.
- [2] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.
- [3] Zinovy Diskin. Model synchronization: Mappings, tiles, and categories. In *Generative and Transformational Techniques in Software Engineering III*, pages 92–165. Springer, 2011.
- [4] Zinovy Diskin, Arif Wider, Hamid Gholizadeh, and Krzysztof Czarnecki. Towards a rational taxonomy for increasingly symmetric model synchronization. In *Theory and Practice of Model Transformations*, pages 57–73. Springer, 2014.
- [5] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society, 2007.
- [6] Sinem Getir, Lars Grunske, Christian Karl Bernasko, Verena Käfer, Tim Sanwald, and Matthias Tichy. Cowolf—a generic framework for multi-view co-evolution and evaluation of models. In *Theory and Practice of Model Transformations*, pages 34–40. Springer, 2015.
- [7] Holger Giese and Robert Wagner. Incremental model synchronization with triple graph grammars. In *Model Driven Engineering Languages and Systems*, pages 543–557. Springer, 2006.
- [8] Frank Hermann, Hartmut Ehrig, Fernando Orejas, Krzysztof Czarnecki, Zinovy Diskin, and Yingfei Xiong. Correctness of model synchronization based on triple graph grammars. In *Model Driven Engineering Languages and Systems*, pages 668–682. Springer, 2011.
- [9] Igor Ivkovic and Kostas Kontogiannis. Tracing evolution changes of software artifacts through model synchronization. In *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, pages 252–261. IEEE, 2004.
- [10] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl: A model transformation tool. *Science of computer programming*, 72(1):31–39, 2008.
- [11] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, 2006.
- [12] Ulrich Nickel, Jörg Niere, and Albert Zündorf. The fujaba environment. In *Proceedings of the 22nd international conference on Software engineering*, pages 742–745. ACM, 2000.
- [13] Omg. Omg meta object facility (mof) core specification version 2.5. *Final Adopted Specification (June 2015)*, 2015.
- [14] Hui Song, Gang Huang, Franck Chauvel, Wei Zhang, Yanchun Sun, Weizhong Shao, and Hong Mei. Instant and incremental qvt transformation for runtime models. In *Model Driven Engineering Languages and Systems*, pages 273–288. Springer, 2011.
- [15] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.
- [16] Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 164–173. ACM, 2007.