Proposal for Bachelor Thesis 03

01 November 2014

William Bombardelli da Silva

Defining Relations Between Software Meta-models: Extending the CoWolf Synchronization Framework

Introduction

This document intents to describe the proposal number 01 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 using the CoWolf Framework [QUOTE] 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 [Czarnecki and Helsen 2006] "*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 themselves, 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 in software industry 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 (M2-level models, see [QUOTE]) commonly used in software systems of current industry projects, defining their meta-model definitions; second to define (or write) the relations between the chosen meta-models; and last, if possible, to extend the CoWolf Framework [QUOTE] to support at least some of these new meta-models for synchronization.

continues in next page...

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 [QUOTE]) meta-models, including for example *Java code*, *UML Diagrams*, *Java Doc*, among others.

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 most prominent result so far is the CoWolf Framework [QUOTE], which will be extended by this work.
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 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 of delivering a consistent and clear definition of a network of interconnected meta-models to both research and industry community. Therefore the availability of such a network might finally allow the extensive use of Model-driven Engineering in the praxis − helping bridging the gap between system abstractions and its 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 2007] gives an overview on the realm, and an interesting point of view about the challenges. In [Mens 2006] a taxonomy for model transformation is proposed, what helps to carry more precise analysis. In [Czarnecki and Helsen 2006] a survey was driven and a framework for classification of model transformation approaches was presented. In [Diskin 2015] a taxonomy for a network of models is presented and in [Diskin 2011] a theoretical algebraic basis is proposed. Between the attempts to build a model synchronization tool are the [QUOTE ATL ECLIPSE], which uses the *Atlas Transformation Language* to code the relations between models; the Medini QVT [to website], which claims to implement the *Query/View/Transformation Language* to code the relations; and the FUJABA [QUOTE FUJABA], in which relations are coded using *Triple Graph Grammars*. Other publications aim to solve specific problems like [QUOTE Hermann 01], [QUOTE Xiong], [GIESE],[IVKOVIC], OR [SONG], who have proposed advanced algorithms for bidirectional synchronization. But none or the above have accomplished to develop a extensible efficient tool capable to integrate 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" [QUOTE CoWolf]. Although the number of supported meta-models is rather small, the extensibility claimed and the brief analysis we have made over the tool has shown to us to be 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. And as meta-models are described in *EMF* [QUOTE EMF] and their relations are written in *Henshin* [QUOTE HENSHIN] the CoWolf Framework was chosen as tool to be used in this study.**

continues in next page...

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.

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" [Czarnecki and Helsen 2006]. Examples of models, according to this definition, are *UML diagrams*, *OCL 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 [Czarnecki and Helsen 2006]. 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 [Diskin 2011]. 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 the problem of defining the common meta-models used in information systems in the current state of industry and the relations relations between them. Second is the expanding of the *CoWolf Framework* [QUOTE], adding at least some of the relations defined in the first phase and thus endeavoring toward the creation of a unique tool integrating several current common meta-models.
An example of a possible network of models created by the end of this work is supposed to look basically like the figure 1:

continues in next page...

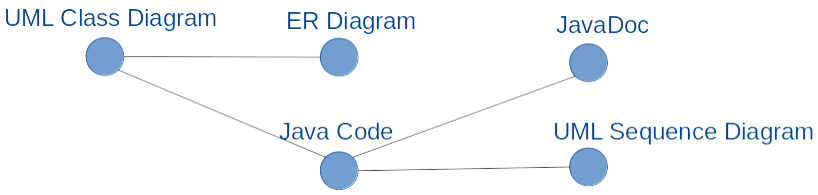

Figure 1: Example of a possible network of meta-models to be constructed. Like stated in the text, the meta-models are chosen accordingly basically to the frequency they are used currently in the software engineering industry.

In the first moment the meta-models comprised in the network are to be defined. So for example, in this phase the definition of the meta-model *UML* or the meta-model *Java Code* will be written. This can be made using the *XMI* standard from *OMG* [QUOTE]. Later on, given the defined meta-models (this is the vertices of the graph), their relations can be written (this is the edges of the graph). This can be made with model transformations languages like *ATL*, *QVT (Query/View/Transformation)* or even with *Triple Graph Grammars* (see [QUOTES]). 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*, *JUnit* and other meta-models can be also defined. Some of these relations might be found in current research literature, others might have to be found during the development of this thesis. After having this network of meta-models (M2-level models) ready, creating an actual network of models (M1-level models) of any software system is trivial, thus study cases and example networks can be developed. In the end the result might be a network of meta-models useful in the praxis, in benchmarks and further researches as well. Theoretical properties, like completeness, correction and soundness of relations, and a record of the difficulties found are expected to arise from such development.

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.
- Problems like cycles in the network might also bring some difficulties.

Time Schedule

Bibliography

Diskin, Zinovy, et al. "Towards a rational taxonomy for increasingly symmetric model synchronization." *Theory and Practice of Model Transformations*. Springer International Publishing, 2014. 57-73.

Diskin, Zinovy. "Model synchronization: Mappings, tiles, and categories." *Generative and Transformational Techniques in Software Engineering III*. Springer Berlin Heidelberg, 2011. 92-165.

Zinovy Diskin, Hamid Gholizadeh, Arif Wider, Krzysztof Czarnecki, "A Three-Dimensional Taxonomy for

continues in next page...

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 planed.

Bidirectional Model Synchronization", *The Journal of Systems & Software (2015)*, doi: 10.1016/j.jss.2015.06.003

France, Robert, and Bernhard Rumpe. "Model-driven development of complex software: A research roadmap." *2007 Future of Software Engineering*. IEEE Computer Society, 2007.

Käfer, Verena, Tim Sanwald, and Matthias Tichy. "CoWolf–A Generic Framework for Multi-view Co-evolution and Evaluation of Models." *Theory and Practice of Model Transformations: 8th International Conference, ICMT 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 20-21, 2015*. Proceedings. Vol. 9152. Springer, 2015.

Nickel, Ulrich, Jörg Niere, and Albert Zündorf. "The FUJABA environment." *Proceedings of the 22nd international conference on Software engineering*. ACM, 2000.