Proposal for Bachelor Thesis 01

26 October 2014

William Bombardelli da Silva

Beyond the UML2Java: Defining a Network of Models

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 building a network of software meta-models describing the relations (or transformations) between them is to be tackled.

Key-words

Model Synchronization, Iterative Model Transformation, Model Transformation, 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 possible diversity of models in a system and the vast number of their relationships suggests the creation of a network of models for a system. Where each component of the network is a different model, and two components are interconnected if, and only if their respective models have a relationship between them. One can see this network alternatively as a graph, whereas each vertex v represents a model, and an edge connecting v_i and v_j exists if, and only if there is a relationship defined between both models i and j.
The goal of this bachelor thesis is then to create a network of typical meta-models (M2-level models, see [QUOTE]) commonly used in software systems in current industry projects, writing down firstly the meta-model definitions, and later writing their relationships. In the end of the creation of this network it is expected the inference of theoretical properties to arise and the creation of a report comprising the difficulties and the next challenges for the problem. The meta-models used might be narrowed to MOF compliant (see [QUOTE]) meta-models though, including *Java code*, *UML diagrams*, *formal specifications*, among others.

Beyond the UML2Java: Defining a Network of Models continues in next page. . .

Motivation

The lack of relation definitions in current literature for non-common models in research, but extensively used models in industry, is the main motivation for this thesis. Examples of these models are *OCL contracts* (used in design by contract methods), unit tests, textual documentation and formal specification. It means, that the success of this thesis might bring the contribution of delivering a consistent and clear definition of a network with these non-common models to both the research and industry community. Therefore the availability of such a network might finally allow the extensive use of Model-driven Engineering in the praxis − 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.

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. These both works may be used extensively in our further development. Additionally, one can judge by the date of publication of these works, that the topic of building a network of models 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. The borders between model transformation and model relation are sometimes fuzzy and both terms might be used interchangeably.
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 the problem of constructing a network of meta-models of an example system, which should be similar to the common information systems from the current state of industry.
An example of a possible network is supposed to look basically like the figure 1:

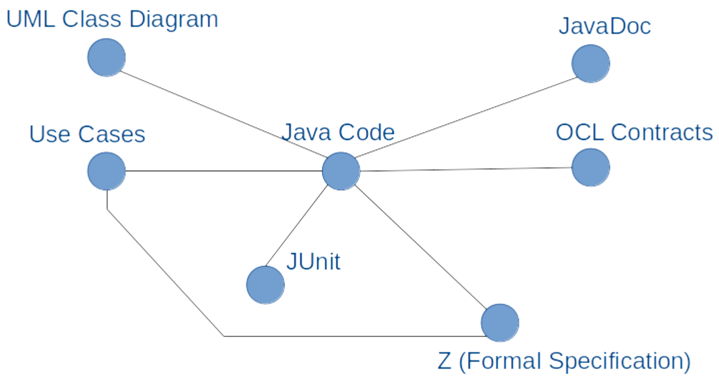

Figure 1: Example of a possible network of meta-models to be constructed. Like stated in the text, the meta-models are chosen accordingly basically to the frequency they are used currently in the software engineering industry.

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

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.

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 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.