Software Modelling Support for Small Teams

Martin OLEJÁR*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
xolejarm1@stuba.sk

Currently, teams comprising many people participate in software modelling and despite usage of different tools for software modelling using UML they have to face problems such as model inconsistency, the determination of the authorship of model parts, fast detecting, identification and correction of defects in models and model synchronization.

In our thesis, we analyse small teams comprising 2-3 members that encounter software modelling techniques for the first time. We focus our attention primarily on model synchronization and secondarily on detection, identification and correction of defects in models. In comparison with large teams, small teams can analyse the mentioned problems in more details and solve them at a lower level.

For the purpose of analysis of small teams' work, we have made an analysis of 20 projects worked out in the course *Principles of Software Engineering* at our faculty. We have found 2 types of defects. On the one hand, there are defects directly concerning software model, for instance missing specific item or useless item. On the other side, there are syntactic defects, for example missing name of item or filling incorrect caption. The most frequent defects for each type of diagram are in the Table 1.

Table 1. The most frequent defects in UML diagrams.

Type of diagram	The most frequent defect
Activity diagram	*Missing merge node*
Use case diagram	*Use case at a low abstraction level*
Class diagram	*Incorrect cardinality*
Sequence diagram	*Incorrect message type*
State machine diagram	*Missing transition trigger*

The mentioned defects could be also a consequence of missing model synchronization. That´s why our aim is to propose a method for model synchronization and support small teams of 2-3 students by implementing this method as an add-in for the tool *Enterprise Architect* that is used in the course.

* Supervisor: Karol Rástočný, Institute of Informatics, Information Systems and Software Engineering

Spring 2016 PeWe Workshop, April 2, 2016, pp. 19–20.

Our method for model synchronization is based on graph analysis and on the algorithm of incremental model synchronization [1]. This algorithm starts the synchronization process directly at the nodes of the correspondence model that belong to the modified elements. The correspondence model contains the nodes that maintain information about identifiers of the corresponding elements in models. Our method for model synchronization is described in the next paragraphs.

All modifications that have been made in model are saved and processed. Each modification contains basic information about modification, identifier of modified element, identifier of team member, who has made it, and time stamp of modification. In general, we can divide possible modifications into addition, change, transfer and deletion of the item. For the purpose of fast and optimized model synchronization, the method processes modifications immediately after they were made.

Model synchronization comes after processing modifications. Depending on a type of a modification, required operation is provided in the target model according to its parameters and identifiers. After completion of all operations, the whole model is refreshed for a current display of all items.

For the needs of the next model synchronization, it is necessary to update the correspondence model. After each addition of an element, we need to add the identifier of just created item to the correspondence model. This operation is not required after other modifications.

In order to optimize our method, we identify the modifications that we do not have to synchronize. In case of a deletion of an item, it is possible to ignore the whole sequence of modifications of this item. We merge multiple modifications of the same element that are of the same type. It is enough to consider only the last modification and ignore all modifications of the same type made before.

Extended version was published in Proc. of the 12th Student Research Conference in Informatics and Information Technologies (IIT.SRC 2016), STU Bratislava.

Acknowledgement. This contribution is the partial result of the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

[1] Giese, H., Hildebrandt, S.: Incremental Model Synchronization for Multiple Updates. In: *GraMoT '08: Proceedings of the Third International Workshop on Graph and Model Transformations*, ACM Press, New York, (2008), pp. 1–8.