Software Modelling Support for Small Teams

Jakub ONDIK*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
jakubondik@gmail.com

Small teams face various problems during software modeling process, such as model synchronization, authorship assessment of model parts and model defects identification. These problems are present both in working and educational environment [1][2].

In our thesis we focus on educational aspect of software modelling. We propose the method for fast detection and correction of defect in UML diagrams of students. We believe, that early feedback on students' actions during software modeling can result in lower amount of defects in their models and in better modelling habits. We focus on small teams consisting of two to three members, specifically on small teams of students.

As a basis for our work we analyzed software models of students enrolled in course named *Principles of Software Engineering* at our faculty. We identified frequent defects in activity, use case, class, sequence and state diagrams. These defect include missing names for initial and final nodes in activity diagrams, missing extend and include relationships between use cases and their references in scenarios descriptions in use case diagrams, wrongly named classes, missing associations names and cardinalities in class diagrams, wrong or misplaced combined fragments in sequence diagrams and missing triggers of transitions in state diagrams.

These types of defects require specific method of detection and correction. Therefore, we propose the method based on rules specifically crafted to detect and correct these types of problems. Example of such a rule can be seen on Figure 1.
This rule consists of multiple elements, each describing a significant part of element this rule is bound to, with an exception of *name*. This rule is bound to name attribute of element of type class with any possible stereotype. It contains a checking expression, which checks for validity of specified element's attribute, in this case the name of the class element. This is checked by searching the name of class in embedded dictionary. If this name is not present in the dictionary, the checking expression returns *false*, which means this name is not valid. Defect message specified in rule element *defectMsg* is then shown to student, with *{0}* parameter replaced by defective class name. Student is presented with option to highlight defective class in his diagram, to hide this defect and

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

to automatically correct it. If student chooses to correct it, correction method is executed. This method is specified in *correct* part of the rule. If the defect is corrected by the student on his own, the defect message disappears. Activations of the rules is based on student's actions such as addition, removal and modification of elements in model.

```
{
name: 'Class name should be noun',
element: {
                  type: 'Class',
                  stereotype: '*'
   },
attribute: {
        type: 'Name',
        stereotype: '-'
     },
content: {
     defectMsg: 'Nespravny nazov pre triedu "{0}".',
       valid: 'dict.getWord(Name).isNoun',
       correct: 'setName(dict.getBaseNoun(Name))'
   }
}
```

Figure 1. Example of rule used to detect wrongly named classes.

We implemented this method as an extension for the modelling tool named Sparx Enterprise Architect. We also provide it to students of *Principles of Software Engineering* course to observe, if quality of their models improves. As a side result we collect data about their actions. This data can serve as a basis for future work, such as defect prediction in software models using neural networks [3].

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] Soler, J., Boada, I., Prados, F., Poch, J., Fabregat, R.: A web-based e-learning tool for UML class diagrams. IEEE EDUCON 2010 Conference, 2010, pp. 973–979.
[2] Hasker, R. W.; Rowe, M.: UMLint: Identifying defects in UML diagrams. In American Society for Engineering Education, American Society for Engineering Education, 2011.
[3] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA, 1995.