Inconsistencies in modeling

IBM Software Group

Claudio Grolimund: „Inconsistencies in models should be quickly identified and resolved. Within the framework of the Rational Software Developer Platform 2009, Prof. Dr Egyed presented a process with which models can be tested for inconsistencies after a change rapidly, accurately and automatically. Following his post doctoral research, Dr Egyed worked for seven years in software development on industry-related research projects for the Techknowledge Corporation in the USA. He later moved to University College in London. Today he is leader of the Institute for System Engineering & Automation at the Johannes Kepler University in Linz, and is currently mainly occupied with model-based software development. In a short interview, Dr Egyed presented his approach. He explained why inconsistencies in model-based software development remain a great problem for developers, even after many years of experience.“

Alexander Egyed: „What’s special about inconsistencies is that they are basically not an actual problem – they have in fact both advantages and disadvantages. Inconsistencies in software arise quite simply because there is an attempt to model various aspects differently. An analogy to this is the graphic representation of a house. You can present a house from different angles, for example in a frontal or side view. Other perspectives could also be selected, such as a depiction of the cabling and pipes. Specific consistency rules also apply here. For example, the height dimensions of the house in the side view should be identical to those of the frontal view. If there is a discrepancy here, the diagrams won’t fit together. So we see that consistency rules can be developed for different perspectives, even in normal engineering. In software development, there is a similar issue. We have various perspectives and can define them differently, and it is precisely because of this that we need to deal explicitly with the problem of consistency. It also needs to be said, that differences in the modeling don’t just bring disadvantages – they are sometimes completely intentional. After all, in a

model we don’t want to describe the entire product all at once, but just those parts that are directly relevant for an argument or analysis. When different people each define one part of a system, you need to take care that all the individual parts fit together again at the end.“

Claudio Grolimund: „In the course of the interview Dr Egyed explained how inconsistencies following a change can be quickly identified, assessed and resolved.“

Alexander Egyed: „Discovering inconsistencies is actually a very simple matter. Even rudimentary consistency rules can solve highly complex errors in POP models, as I demonstrated with the house example. A consistency test involves defining preferably all important consistency rules, so that they can also be recognized automatically. There are already very many automation tools that do nothing other than evaluate, by various ways and means, these consistency rules in the models, and generate a corresponding value. The result is either consistent, i.e. correct, or inconsistent, i.e. wrong. This is essentially a simple process. Problems with the testing of consistency rules arise when, amongst other things, there are many of these rules and the model is very large. In this case, the testing can take a great deal of time, despite the automation. The automated process is still better than manual processing, but it is by no means optimal if the feedback about an error doesn’t occur quickly. In today’s programming environments, for example in Eclipse, this problem often arises. You write a string, a piece of code for example, and then something is underlined to show that there is an error in this string. Here only partial solution proposals are offered, but with the modeling however, it can be hours before you receive this feedback. If such a process lasts a long time, it will generally only be carried out rarely. And if it is only seldom executed, the problem naturally arises that you are possibly working with errors that have not be identified. Other errors develop from this error. If you notice the existence of an error after a certain time, you not only have to remember what you meant when writing, but you also need to consider how to rectify this error together with all other possible subsequent errors. Up until now, we have actually just spoken about the discovery, and in a rudimentary way, about the removal of inconsistencies. It is the removal itself however, that basically presents the greatest problem. When removing inconsistencies, you need to understand precisely where the origin of the inconsistency lies. An error that I make can have an

effect on five other perspectives and thereby generate five different error messages. This does not mean that I have made five mistakes. On the contrary, there is just one single error, but this results in one of several effects. And the understanding of this, i.e. deducing the cause from the effect, remains a completely unresolved problem.“

Claudio Grolimund: „Within the framework of his presentation, Dr Egyed introduced a solution with which the consistency of a model, even after changes, can be evaluated fast, accurately and automatically. He explained how this model works and how this solution differs from others.“

Alexander Egyed: „There is an approach that I like a lot. In principle, it consists of saying: we are attempting to analyze the consistency rules. So someone writes down what these consistency rules have to say, and we simply test what they actually do. On the basis of the results, we then try to find out how best to handle changes. And this method works both for the identification as well as the resolution of errors. The problem is that the automatic analysis of a consistency rule is an extremely complex process, which however essentially only serves to define and identify simple inconsistencies. This is the point at which we start. We proceed in precisely the opposite direction and pursue a radical new path. We observe the consistency rule. We make no attempt to understand it. We just surround it with a small wrap that has no other purpose than to show how the consistency rule reacts to the model. Here it doesn’t matter at all for what reason the consistency rule checks a model element. What is more important for us is that the consistency rule subjects precisely this model element to a consistency check. If a consistency rule has checked a model element during the evaluation, and this model element changes, then this consistency rule must be readjusted. The technology that we have developed for this is based on a so-called Model Profiler, whose only task is to monitor the System C Checker. In principle, this information provides us with all we need to decide when and how a change affects a model and what effect this can have on inconsistencies. The greatest advantage of this approach is that, with the help of this information you don’t just discover the effect of an inconsistency, but you can also decide where to eliminate errors. So if we return to the aforementioned example, where I explained that there must be differences between error cause and error identification, we can see that an inconsistency is just a message that shows that there is an error

somewhere. This means that the places where an inconsistency will be corrected emerge from precisely the same information from the Model Profiler, which had previously checked this consistency rule. Thereby, in a relatively large model with many model elements, you can find those 10 or 20 elements that need to be checked when an inconsistency arises.

Claudio Grolimund: „In conclusion, Dr Egyed explained his point of view concerning the future of model-based software development and the options offered in the area of consistency assurance for models.“

Alexander Egyed: „In my opinion, the future of model-based software development lies in its integration into the software development process. The modeling phase is very important for software development for one very simple reason: the longer an error in software development remains undetected, the more expensive it is to eliminate it. The resolution of a problem already identified in the initial analysis, i.e. right at the beginning, is associated with specific costs. The removal of the same error within the framework of maintenance during testing or integration costs about 30 times that amount. If the presence of an error that makes the software unusable is not identified until onsite at the customer location, the costs for resolving it can be up to 200 times that of a resolution during the initial phase. Modeling is part of that development phase in which the resolution of errors is cheapest, if the problem can not be solved in the initial analysis.“

Claudio Grolimund: „We would here like to thank Dr Egyed for this interview, and point out that you can find further links on the subject of model-based software development on our homepage.“