Automated Quality Assurance for UML Models

Tilman Seifert, Florian Jug
Technische Universität München
Software & Systems Engineering
seifert@in.tum.de, florian@jugga.de

Günther Rackl
BMW Group
guenther.rackl@bmw.de

Abstract: Model based development, like proposed by the OMG's Model Driven Architecture (MDA), aims to raise the level of abstraction from working on the code to working with models. For a professional production environment, the means for quality assurance on the model level are scarce. We introduce an approach for automated quality assurance on the model level, integrated into the tool of a modern development environment.

1 Model Based Software Development

In an idealized vision for a model based development process, models are developed, used, and refined in all phases of the process. For requirements analysis, models help to find a common vocabulary between customers and developers. In the specification phase, models are used to completely describe the business logic of the system while abstracting from technical infrastructure and implementation detail. In the development phase, models are the input for the code generation process. For the maintenance phase, models are an important part of the system documentation since they provide views of the system on any desired level of abstraction.

The Model Driven Architecture (MDA) as proposed by the OMG [MM01] is a UML-based approach that aims to realize this vision of model based development. The BMW Group has developed a concrete realization of the MDA called "Component Architecture (CA)" [RSB+04] to improve the development process for information systems. The Component Architecture features a modeling profile to support component based development as well as model transformation and code generation facilities. The platform independent model (PIM), which is the central model in CA development, contains modeling elements like Business Activities (BA), Business Entities (BE), Business Components (BC), and others. The usability of this model based approach has been proven in different projects which use the CA environment to develop business applications.

While the technical experiences with this framework have been very positive, we realized that in order to gain full advantage of the model based approach, it is necessary to adapt the development and maintenance process. Since the UML models are the central concept of the approach we started to improve the quality assurance (QA) process for models.

In this paper, we present our conclusions about the QA process in its model based context, and introduce our tool called "Model QA" to support the QA process for UML models.

2 Automated Quality Analysis of Models

Project work for developing and maintaining information systems at BMW Group is typically done in mixed teams with internal and external developers, with a varying degree of subcontracting development work. For models delivered by external partners, it is important to be able to perform an efficient and complete QA process before accepting the product. In general, a QA process contains three major steps:

Quality planning. The quality criteria of all deliverables are defined. This part is usually done for the whole organization. The key is to identify quality goals and to define measurable quality attributes that contribute to the quality goals. In our case, guidelines were developed as part of the CA approach that can be used as a reference for developers and as a checklist for the review or acceptance process.

Quality analysis. This is the operative part that must be conducted for every deliverable. Our aim is to support these tasks with tools that are integrated into the development environment so that they are easily accessible for developers and reviewers alike.

Quality controlling ensures that quality criteria are met and takes corrective action where necessary.

The quality analysis is usually done with more or less formal reviews based on checklists. For source code, however, it is possible to use automated checks of quality conditions and metrics. BMW Group has experience with automated quality checks for Java systems; a collection of tools that check feasibility conditions and certain quality metrics is in use. Since the experiences have been very good, we decided to develop tool support for checking UML models in a similar fashion. Our main goal was to give easily usable support to the developers as well as to the reviewers of UML models for the Component Architecture. Therefore we aimed at a seamless integration into the development tool chain.

Note two important properties of quality analysis for UML models. First, improving the quality analysis for models has an impact on the whole development life-cycle, since models are delivered in early phases of the project (whereas code analysis can be done only while and after the implementation phase).

Second, it is difficult to do quality analysis on general UML models since there are only few general rules that can be used as quality indicators (e. g. regarding cycles, coupling, or coherence – just the same rules that are used on source code). However, the UML allows to to define domain specific modeling profiles. Such domain specific profiles allow to assign a more precise semantic to modeling elements and relations. We can use this semantic information to derive rules to perform quality checks on the models that are based on these profiles. These rules on the model level are more concrete, more powerful, and more useful than comparable analysis on the source code level.

BMW Group's Component Architecture (CA) defines a UML profile to characterize the "syntax" of a CA PIM (Platform Independent Model). UML stereotypes are used to define Business Components (BC), Business Entities (BE), Business Activities (BA), and more. PIMs that are developed in conformance to the CA are UML models that are subject to the

syntactic restrictions given by the CA. These restrictions can be used to design automated, simple but powerful feasibility checks. For example, the CA contains the architecture principle: "BEs may only be accessed by BAs, not by other BEs." This architecture principle can be directly used as a syntax rule which can be easily implemented as a syntax check on the CA model because model elements use stereotypes that explicitly characterize them as e. g. BEs or BAs.

3 Case Study: Tool Support for QA of UML Models

We conducted a case study with a prototype implementation of the concepts in [Jug04] (section 2) to support the quality analysis step in the QA process. Our case study demonstrates how to use quality requirements to create tool support that is integrated into the development environment.

Together Control Center [TOG05] can be extended by adding modules and integrate them into the IDE. Those modules can access information about the UML models; they can traverse the UML model and check it for quality criteria. We used this mechanism to develop an environment called "Model QA" which allows us to easily implement and integrate new quality audits. Violations of quality criteria are reported in the same way as compiler errors and warnings are presented to the developer. Since models are developed early in the process, it helps to avoid modeling mistakes or quality flaws as early in the process as possible.

At BMW Group, automated QA measures on the code level are a widely adopted standard procedure for Java projects. Therefore, the general acceptance of automated QA for models is good. The tight integration in the development environment lowers the barrier to using automated support even further. The main advantage is that there is no need to switch tools for the quality checks; in fact, as long as no violations are detected, the Model QA tool doesn't interrupt the workflow at all.

Reviewers are supported in the same way as developers are. They just have to load the models into their IDE and are directly pointed to modeling errors and possible design flaws – if there are any left in the models at all. The main goal is to improve the efficiency of the review process by helping the developers to avoid such mistakes, and to enable reviewers to concentrate on the content of models.

The implemented audits are based on the CA quality checklist that was developed for the "manual" review process of CA models. We found that it is possible to fully automate the quality check for 47 % of the given criteria, and that another 12 % of the criteria could be partially checked automatically. Examples for criteria that can be checked automatically include "Is the dependency graph of the model elements free of cycles?" or "Are all methods of a Business Component Interface implemented by the elements in the component?". Even criteria that are somewhat fuzzy can be checked, e. g. "Is it easy to overview the number of Business Component Interfaces (BCI) for each Business Component (BC)?". In this case, the number of BCIs per BC is checked against some thresholds.

The remaining criteria are intended for human readers and cannot be supported by a tool,

e. g. questions like “Are all model elements identified by a descriptive name?” or “Do all model elements of the Business Component (BC) contribute to the goal of the BC?”

We chose a subset of the fully automatic testable criteria for our prototype. For productive use of the Model QA tool as part of the daily development work, the criteria should be revised and extended with respect to the possibilities of tool support to extend the coverage of the quality checks.

The audits that are implemented in Model QA so far strictly check for the model syntax. A further step could be to include other measures, e. g. metrics that give hints to judge more abstract quality properties like readability, extendibility, adaptability, or reusability. However, metrics need be chosen sensibly and used carefully in this context [KB04].

4 Conclusion

For a wide adoption of the model based development approach, an appropriate QA process is indispensable. We introduced tool support for automated QA checks that are integrated into the development process. Note, however, the difference between quality analysis and quality assurance. The analysis is one step in the QA process that can be automated, while the QA as a whole remains a “manual”, creative process that involves interpreting the results of the automated quality analysis and identifying appropriate measures to improve the quality where necessary.

While our approach is only a first step in the direction of tool-supported QA for UML models, it clearly demonstrates the potential of a model based development approach, the possibilities for process improvement by integrating tool support for quality analysis, and the importance of well-defined modeling profiles.

Well-designed models offer rich information which helps to automate the quality analysis which improves the efficiency of the QA process. This is especially interesting because models are developed earlier in the process; therefore, the quality of work products can be improved earlier in the process by using automated support for model QA. The prerequisite is a process that actually develops and uses models in earlier phases of the project.

The crucial requirement for achieving quality and productivity advantages is a modeling profile that appropriately matches the domain, offers the right abstractions for each development phase, and makes it possible to formulate powerful quality criteria. The effectiveness of the automated analysis depends on the underlying quality attributes and how well they are suited to describe the quality of the system. Quality analysis on “plain” UML models that don’t use a specific profile are as powerful as analysis on the source code. Therefore the key to successful automated support is to find the right quality criteria. In our case study, we demonstrate that it is important to take advantage of the additional semantic information given by the stereotypes.

4.1 Related Work

Critical views on the MDA are expressed by Dave Thomas [Tho04], Steve Cook [Coo04] and Martin Fowler [Fow04] who compare the MDA approach to the ideas of CASE tools in the 1980's, "and CASE clearly failed to live up its promises" [Coo04]. Cook promotes Microsoft's MDA alternative. All authors mainly discuss the development of systems using MDA. The maintainability of such systems is analyzed in [SBB04].

Refer to [KB04] for a critical view on metrics and measurement in the software development and maintenance process.

4.2 Outlook

For information systems, a very important quality criterion is the maintainability in the long run, which in turn depends on the readability, the structuring of the system, and a number of other factors. The CA approach is too young to report any experiences with respect to maintainability of CA systems. Analyzing this issue will be future work.

References

[Coo04] Steve Cook. Domain-Specific Modelling and Model Driven Architecture. *MDA Journal, Business Process Trends*, pages 2–10, January 2004.

[Fow04] Martin Fowler. Model Driven Architecture. `http://www.martinfowler.com/bliki/ModelDrivenArchitecture.html` (June 2004), February 2004.

[Jug04] Florian Jug. Methoden und Techniken zur Qualitätssicherung im Software-Prozess der BMW-Group. Master's thesis, Technische Universität München, September 2004.

[KB04] Cem Kaner and Walter P. Bond. Software Engineering Metrics: What Do They Measure and How Do We Know? In *10th International Software Metrics Symposium*, 2004.

[MM01] Joaquin Miller and Jishnu Mukerji. Model Driven Architecture – A Technical Perspective. http://www.omg.org/cgi-bin/apps/doc?ormsc/01-07-01.pdf (June 2004), July 2001.

[RSB+04] Günther Rackl, Ulrike Sommer, Klaus Beschorner, Heinz Kößler, and Adam Bien. Komponentenbasierte Entwicklung auf Basis der Model Driven Architecture. *ObjektSpektrum*, 5, May 2004.

[SBB04] Tilman Seifert, Gerd Beneken, and Niko Baehr. Engineering Long-Lived Applications Using MDA. In *Intl. Conference on Software Engineering and Applications*, pages 241–246, Cambridge, MA, November 2004. IASTED.

[Tho04] Dave Thomas. MDA: Revenge of the Modelers or UML Utopia? *IEEE Software*, 21(3):15–17, May 2004.

[TOG05] Together Website. `www.borland.com/together`, April 2005.