The Quamoco Tool Chain for Quality Modeling and Assessment*

Florian Deissenboeck, Lars Heinemann, Markus Herrmannsdoerfer,
Klaus Lochmann, Stefan Wagner
Technische Universität München, Garching b. München, Germany

ABSTRACT

Continuous quality assessment is crucial for the long-term success of evolving software. On the one hand, code analysis tools automatically supply quality indicators, but do not provide a complete overview of software quality. On the other hand, quality models define abstract characteristics that influence quality, but are not operationalized. Currently, no tool chain exists that integrates code analysis tools with quality models. To alleviate this, the Quamoco project provides a tool chain to both define and assess software quality. The tool chain consists of a quality model editor and an integration with the quality assessment toolkit ConQAT. Using the editor, we can define quality models ranging from abstract characteristics down to operationalized measures. From the quality model, a ConQAT configuration can be generated that can be used to automatically assess the quality of a software system.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Software Quality Assurance (SQA)*

General Terms

Management, Measurement

Keywords

Quality Assessment, Quality Modeling, Tool Chain

1. INTRODUCTION

To effectively manage software costs as well as to prove the excellence of software products, a comprehensive assessment of software quality is essential. Two basic ingredients are currently available to support this quality assessment.

First, existing code analysis tools can be applied to obtain indicators for software quality. However, each code analysis tool only focuses on a certain aspect of software quality. To obtain a complete overview of the quality of a software system, the results of many code analysis tools need to be aggregated. However, due to the heterogeneity of the results of different code analysis tools, it is often unclear how to best aggregate them.

Second, quality models define abstract characteristics that influence software quality. However, existing quality models do not provide sufficient detail to operationalize them for assessing software quality. Therefor, the abstract quality characteristics need to be decomposed until they can be quantified by measures. The measures in turn can be determined by applying existing code analysis tools and manual techniques like code reviews.

To support a comprehensive assessment of software quality, adequate tool support thus needs to integrate both code analysis tools and quality models.

Problem. Currently, there is no tool chain that integrates quality models with code analysis tools. However, such a tool chain is required to operationalize the quality models for assessing software quality. When operationalizing quality models, many measures and aggregations need to be defined to get a complete overview of the software quality. As a consequence, quality models can become quite large and complex. To master this complexity, adequate tool support for editing quality models is required. Due to the size of the quality models, the quality of a software product cannot be assessed manually. Further tool support is thus required to automatically assess a software product using the measures defined in the quality model. The analysis results need to be collected and integrated according to the aggregations defined in the quality model.

Contribution. In this paper, we present the tool chain for defining and assessing software quality that has been developed in the Quamoco[1] project. To define software quality, we have implemented a quality model editor that is based on an explicit meta model. The editor provides different views to master the complexity of operationalized quality models. Using the editor, the user can define the aggregations of quality characteristics based on the measures that quantify the quality factors. To assess software quality, the tool chain integrates with the existing quality analysis toolkit ConQAT[2] [2], which we extended with additional functionality for reading a quality model and assessing a software system according to the model. The quality model editor supports the automated generation of a ConQAT configu-

*The presented work was funded by the German Federal Ministry of Education and Research (BMBF), grant "Quamoco, 01IS08023B".

ICSE '11, May 21–28, 2011, Waikiki, Honolulu, HI, USA

[1] `http://www.quamoco.de`
[2] `http://www.conqat.org`

ration. Using the configuration, the quality model and the software system as input, ConQAT produces a dashboard that allows to inspect the results of the quality assessment.

2. RELATED WORK

The literature proposes a large number of quality models [1, 4, 5] which define the term *quality* by decomposing it into more tangible attributes. However, these quality models remain on a high level of abstraction, and no tool support exists to conduct quality evaluations based on them.

Besides, countless code analysis tools support tasks as different as bug pattern identification, clone detection and dependency cycle analysis. These tools, however, focus on very specific aspects of software quality and are, hence, ill-suited for a holistic quality assessment. Moreover, these tools provide sophisticated analysis techniques, but often fail to support quality engineers in interpreting the analysis results.

Multiple dashboard tools (QALab[3], Sonar[4], XRadar[5]) are built on top of the specialized quality analysis tools. Dashboard tools collect, aggregate and visualize data of metric calculators, static code checkers, and other automatically accessible sources. They aim at providing a quality overview of a software system to monitor and control development activities. However, none of these tools establishes an explicit link between the specified quality requirements (provided by the quality model) and the actual quality characteristics of a software system (provided by the code analysis tools).

Besides experimental research tools like [8, 10], the research project *Squale*[6] develops an explicit quality model and a tool for evaluating software products. The main difference to our approach is that Squale uses a fixed quality model, while our tool chain offers an editor to specify project-specific quality models. Moreover, Squale supports only a fixed set of code analysis tools. In contrast, the editor and ConQAT allow a flexible configuration and integration of code analysis tools, without the need to change the source code. Finally, Squale is limited to automated measures, while Quamoco allows to seamlessly integrate results generated by manual analyses like inspections and reviews.

3. QUAMOCO TOOL CHAIN

The Quamoco tool chain[7] consists of the graphical quality model editor used for *quality modeling* and the quality assessment toolkit ConQAT used for *quality assessment*. ConQAT is open source software licensed under the Apache 2.0 license and the quality model editor will be released as open source in 2011. The complete tool chain presented in this paper is available on the Quamoco website[8]. Figure 1 gives an overview of the Quamoco tool chain. The following sections detail the two main parts of the tool chain.

Quality Modeling. The Quamoco quality model editor is an Eclipse/EMF-based tool, supporting to author quality models based on a defined meta model. Quality models for real-life quality evaluations grow large (several hundreds of model elements), thus requiring adequate tool support.

[3] `http://qalab.sourceforge.net`
[4] `http://www.sonarsource.org`
[5] `http://xradar.sourceforge.net`
[6] `http://www.squale.org`
[7] `http://www.youtube.com/watch?v=D9sKbctf0Cw&hd=1`
[8] `http://www.quamoco.de/webportal/icse2011.zip`

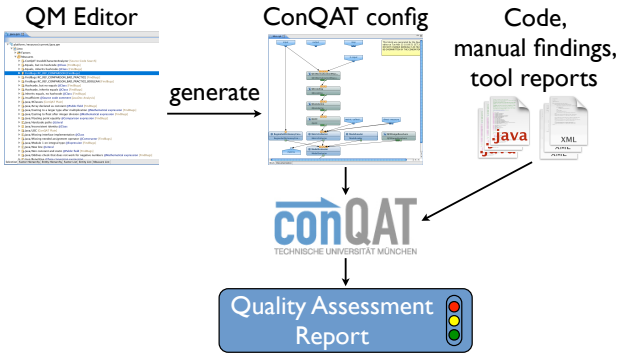

Figure 1: Quamoco Tool Chain

The quality meta model extends activity-based quality models [3] by operationalization concepts, code analysis tools and aggregation formulas. A detailed explanation of the meta model is beyond the scope of this paper. The meta model consists of 18 model entities and 41 relations. To effectively model instances of this meta model, the editor provides three hierarchical views that allow to browse and edit the model represented as a tree. Additionally, there are three table views, with filtering and sorting capabilities.

Based on the meta model, a large number of consistency rules are defined. These rules are automatically checked by the editor. All rule violations are shown in a list view, and by clicking an entry, the editor navigates to the affected element. Furthermore, all elements with rule violations are marked with warning symbols.

Since the quality model is used to conduct concrete quality evaluations, it also contains aggregation formulas and references to code analysis tools. For these formulas, we developed a scripting language that is able to perform various mathematical calculations and that can refer to elements in the quality model. The editor supports this language directly and performs syntax checks of it.

To manage large quality models in practice, a modularization concept was introduced. Quality models can be split into modules, having explicit dependencies between them. The editor realizes the modules as separate files and thus enables the concurrent work of different users on a single quality model. The modularization concept further enables inheritance and overwriting concepts for model elements. This way, a general module can define commonly known metrics. A specific model for a programming language can overwrite these metrics and attach concrete tools to them.

Furthermore, the editor contains a context-sensitive online help. When selecting a model element in one of the views, the help window shows the corresponding help text. Through this help system, the description of the quality meta model and a modeling guideline is integrated.

Quality Assessment. ConQAT is a toolkit for the creation of quality dashboards that allows to model the analysis configuration with a graphical domain-specific language (DSL). It provides diverse quality analyses out of the box and integrates with state-of-the-art code analysis tools like Findbugs, PMD, FxCop or Gendarme. The results of the quality analyses are aggregated and visualized in configurable HTML dashboards. The configuration DSL contains blocks and processors, which are connected with edges to denote the data flow. Processors are ConQAT's atomic processing units, which are each implemented in a Java class. Blocks

are composite units built from processors and blocks.

Within the Quamoco tool chain, ConQAT is used to perform the quality assessment based on the quality model. The quality model editor allows to automatically generate a ConQAT configuration that contains all the required analyses and aggregations to assess the software system according to the quality model. For each measure defined in the quality model, a corresponding block is generated. For instance, if a quality model refers to a FindBugs rule, the FindBugs block is added to the resulting analysis configuration.

For executing the analysis, settings specific for the analyzed software system have to be provided in a run configuration. Examples include the input directory, external tool reports, and the location of findings discovered in manual inspections. As as result, ConQAT produces a quality assessment report in HTML format that aggregates all measurements and findings according to the quality model. Since the actual quality assessment runs in batch mode, it can be easily integrated in a continuous integration environment to constantly monitor the quality status of the software system. That way, the impact of changes to the system on quality characteristics can be detected early. The continuous evaluation also allows for convenient fine-tuning of the quality model, since the effect of adaptations can be directly inspected in the assessment result. Trend charts allow insights about the evolution of the quality status over time.

4. EXPERIENCE

The presented tooling is developed and used in the research project Quamoco. The project partners (Technische Universität München, SAP, Siemens, Capgemini, Fraunhofer IESE, itestra) translate their existing guidelines, rule sets, inspection checklists, etc. to the quality meta model. In a study [6], we analyzed the expressiveness of the meta model and found that it covers the needs of all partners.

Up to now, each partner developed a quality model using the editor, covering a subject area important to him. The different models were then integrated into a single model, by extracting generally accepted quality factors using the overwriting mechanism. The resulting quality model consists of 1073 model elements. The quality model editor proved adequate for managing this large model.

Siemens created a model containing the rules of the static code checker Gendarme for C#. Moreover, a small subset of PCLint rules for C/C++ was incorporated. An existing quality model developed by Technische Universität München and MAN (a truck manufacturer)—containing static code checker rules of FindBugs and PMD for Java—was also introduced. SAP modeled the Web Content Accessibility Guidelines (WCAG), which are not targeted at source code, but at graphical user interfaces. Itestra provided a set of guidelines containing manual inspection rules for software.

Using this quality model, several quality evaluations were conducted with the tool chain. The evaluation of systems developed with the programming languages C/C++, C# and Java is possible. The evaluation of web pages by the accessibility module was operationalized using the results of manual inspections.

For a more thorough test of the quality model and its tool chain, a structured case study was conducted [7]. The ranking of five systems produced by our tool chain was compared to a ranking based on manual quality evaluations published in [9]. Statistical tests showed a significant correlation.

5. CONCLUSION

Over the last decades, multiple quality models and quality modeling approaches have been proposed. Moreover, academia and industry provide a plethora of quality analysis tools as well as quality dashboard tools that create an integrated display of various quality data. However, we lack a tight integration of quality models and quality analysis tools. Such an integration is necessary to effectively and efficiently compare the actual quality characteristics of a software system (provided by the code analysis tools) with the specified quality requirements (provided by the quality model). To address this, we present the integrated tool chain developed in the Quamoco project. The tool chain enables quality engineers to create and maintain quality models with an editor. Such models explicitly describe the relation to the code analysis tools that are used to analyze software systems and the aggregation instructions used to condense the measurement data. The actual assessment of software systems is supported by the quality assessment toolkit ConQAT that interfaces well-known analysis tools like FindBugs and FxCop. Based on the measurement data provided by these tools and the specified aggregations in the quality model, ConQAT generates a quality report that is directly related to the quality model. Hence, the presented tool chain closes the gap between quality models and analysis tools. For the first time, it enables quality engineers to leverage the power of existing quality analysis tools within the structured framework of explicitly defined quality models.

6. REFERENCES

[1] B. W. Boehm. *Characteristics of Software Quality.* North-Holland, 1978.

[2] F. Deissenboeck, E. Juergens, B. Hummel, S. Wagner, M. y Parareda, and M. Pizka. Tool support for continuous quality control. *IEEE Software*, 25(5):60–67, 2008.

[3] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An Activity-Based Quality Model for Maintainability. In *Proc. of the ICSM'07*, 2007.

[4] R. G. Dromey. A model for software product quality. *IEEE Trans. Software Eng.*, 21(2):146–162, 1995.

[5] B. Kitchenham, S. G. Linkman, A. Pasquini, and V. Nanni. The SQUID approach to defining a quality model. *Software Qual. J.*, 6(3):211–233, 1997.

[6] M. Kläs, C. Lampasona, S. Nunnenmacher, S. Wagner, M. Herrmannsdörfer, and K. Lochmann. How to Evaluate Meta-Models for Software Quality? In *Proc. of MetriKon'10*. 2010.

[7] M. Kläs, K. Lochmann, and L. Heinemann. Evaluating a Quality Model for Software Product Assessment – A Case Study. In *Proc. of SQMB'11 (to appear)*, 2011.

[8] C. Marinescu, R. Marinescu, R. F. Mihancea, D. Ratiu, and R. Wettel. iPlasma: An Integrated Platform for Quality Assessment of Object-Oriented Design. In *Proc. of the ICSM'05*. 2005.

[9] R. Plösch. Software-Verkostung. 23.04.2010. `http://www.ipo.jku.at/dokumente/upload/Software%20Verkostung%20Impulsvortrag.pdf`.

[10] H. Schackmann, M. Jansen, and H. Lichter. Tool Support for User-Defined Quality Assessment Models. In *Proc. of MetriKon'09*. 2009.