Tool Support for User-Defined Quality Assessment Models

Holger Schackmann, Martin Jansen, Horst Lichter

RWTH Aachen University, Research Group Software Construction

{Schackmann|Lichter}@swc.rwth-aachen.de, Martin.Jansen@rwth-aachen.de

Abstract:

Quality assessment based on software metrics is generally founded on an implicitly or explicitly given quality model, that defines how measurement values are aggregated. The quality model itself is often buried in the supporting tools, which limits adaptability and understandability of the quality model.

In this paper we introduce a generic meta-model for quality assessment models that support metric-based quality evaluations. Furthermore we present a corresponding tool for the definition of quality models by the user, and their automatic evaluation based on underlying third-party metric tools.

Keywords

Quality Model, Process Improvement, Software Metrics, Project Controlling

1 Introduction

The usage of software metrics for the evaluation of product or process quality usually entails the need to provide condensed information on quality characteristics of interest. Thus metric results must be aggregated to a higher level. The kind of aggregation is defined in a quality assessment model. The quality assessment model is often not made explicit, but only implicitly contained in tools for metric evaluation or reporting.

As the metrics of interest are usually organization-specific [1] the quality assessment models also have to be tailored to the needs of the organization. If the quality assessment model is only implicitly given in a tool, its adaptability is limited. Moreover the aggregation of metric results may be difficult to understand for the assessor.

In this paper we present a tool that supports the definition of quality assessment models by the user, and their evaluation based on underlying metric tools. It is embedded into the **QMetric** tool suite that supports the evaluation of metrics on software repositories like change request systems or configuration management systems [2]. However the herein presented quality model editor and evaluation tool is intended to be generic. Thus it is open to be applied with other metric tools, e.g. in the context of code quality assessment.

The remainder of the paper is organized as follows. A brief overview of related work is given in section 2. Recurring concepts in quality assessment models are

discussed in section 3. Requirements of the tool, the underlying meta-model for quality assessment models, and the architecture of the tool are described in section 4. First application experiences of the tool are discussed in the last section.

2 Related Work

Very diverse models are commonly termed “quality model”. So we first need to clarify which kind of models we are interested in. Deissenboeck et al. classify quality models according to their purpose [3]:

- **Definition models** describe which characteristics constitute the quality of a product or process (e.g. ISO 9126 [4]).
- **Assessment models** are used to evaluate quality based on metrics.
- **Prediction models** are used to predict quality (e.g. COQUALMO [5]).

In the following we will focus on assessment models.

The most widespread use of software metrics can probably be found in tools and approaches for quality assessment based on source code metrics. Many of the available code assessment tools just present bare metric results and highlight violations of thresholds that can often be redefined by the user. Some tools also support the evaluation based on a default quality model with limited adaptability. For example, *Logiscope*[1] uses a quality model based on a factor - criteria – metric hierarchy similar to the McCall model [6]. *Swat4j*[2] provides a default quality model leaned against ISO 9126, and an evaluation based on linear equations and adjustable weights. Washizaki et al. propose a quality model for the C programming language that is also based on ISO 9126, and uses comparison data from other projects to derive thresholds for acceptable metric results [7]. Plösch et al. present a code assessment tool called *SPQR* that supports the tailoring of quality models based upon ISO 9126 [8]. *SPQR* can import results from different static code analysis tools. Moreover experts can provide annotations to the metric results, e.g. severity ratings or estimated effort to fix a problem.

There are a number of tools that are targeted at integration of metric results from heterogeneous sources, and their visualization [11, 12, 13]. An example is the tool *ConQAT* which provides configurable processors for aggregation and visualization using diagrams, tree maps, and graphs [14]. Thus the tool can be used to implicitly define a quality model.

The *DesCOTS-QM* tool is targeted at defining quality models for COTS components [15]. The quality model is a hierarchy of quality characteristics, subcharacteristics and attributes. The underlying metrics have to be retrieved manually, and it is unclear whether the tool supports the aggregation of these results.

1 http://www.telelogic.com/products/Logiscope/
2 http://www.codeswat.com/

3 Concepts of Quality Assessment Models

Quality assessment models are based on similar concepts, independent of the entity to be assessed. We will describe them in the following, as they build a foundation for the definition of the meta-model for quality assessment models described in the next section. Different approaches to quality modeling usually have their own terminology. For the sake of clarity we will use the terms as defined in ISO 15939 [16].

The entities of measurement considered in a quality model usually have a layered structure with inclusion relationships (e.g. methods, classes, packages).

Each assessment model is ground on some **base measures** that are used to measure an attribute of an entity based on some model (e.g. cyclomatic complexity, or number of defect reports). Base measures are then combined in order to characterize the entity. Typical operations are

- Normalization with a size measure (e.g. defects per line of code)
- Counting the occurrence of problematic patterns, like violations of certain thresholds (e.g. too complex methods, change requests with a late reaction)
- Aggregation in order to characterize entities of measurement on a higher level (e.g. weighted methods per class, median time until a change request is resolved for a component)

If these derived measures can be interpreted based on some decision criteria, they can serve as **quality indicator**. A further step is usually the aggregation to an index number that is used to summarize a number of observed facts. This aggregation can be based on a linear equation (e.g. McCalls quality model [6]), or more complex functions (e.g. the Maintainability Index [17]). Another approach for the aggregation to an index number is the usage of quality levels. Examples are the maturity levels in CMMI, and the Code Quality Index [10]. Quality levels are typically constructed with the following approaches:

- More quality indicators are considered on each quality level. This can be done in order to point out a typical path of improvement, or with consideration of the directness of the effects and the added costs of certain improvements.
- Tightened thresholds are applied for quality indicators in higher quality levels.

The thresholds used for defining a certain quality level, as well as for identifying problematic patterns, are in some cases derived based on empirical comparison data. The comparison data can either be based on a portfolio of similar projects [10], or based on selected projects that are considered to be acceptable from a quality point of view [7]. Measurement results are then classified according to the value distribution of the comparison data, e.g. by comparing to quartiles [10], or some other function derived from quantiles of the value distribution [7].

4 The QMetric Quality Model Editor and Evaluation Tool

The initial motivation for the development of the tool was to extend the QMetric tool suite with a support for user-defined quality models, and their automatic evaluation based on comparison to a set of similar products. The QMetric tool suite provides tool support for evaluating metrics on change request and configuration management data. Its core is a generic metric calculation component that takes a metric definition in a declarative language as input, and returns time series of metric results as output. Thus the quality models must be operational in a sense that each quality assessment is based on the automatic evaluation of software metrics.

Moreover we aimed at keeping the tool flexible enough to be used with other metric tools, for example in the context of metric-based code quality assessment. Thus the underlying meta-model for quality models must take into account the recurring concepts of quality assessment models described in section 3. It must be open to be extended with aggregation functions needed in future applications. Summarizing, the quality model editor must be based on a meta-model for operational, metric-based quality assessment models.

Accordingly the evaluation tool must facilitate the quality evaluation for an entity of measurement defined by the user, and based on a given quality model. Thus the tool must take into account the structure of the entities of measurement. When using metric evaluations on change request data, the entity of measurement is given by specifying a product or component of interest, and the time span and time granularity for the evaluation.

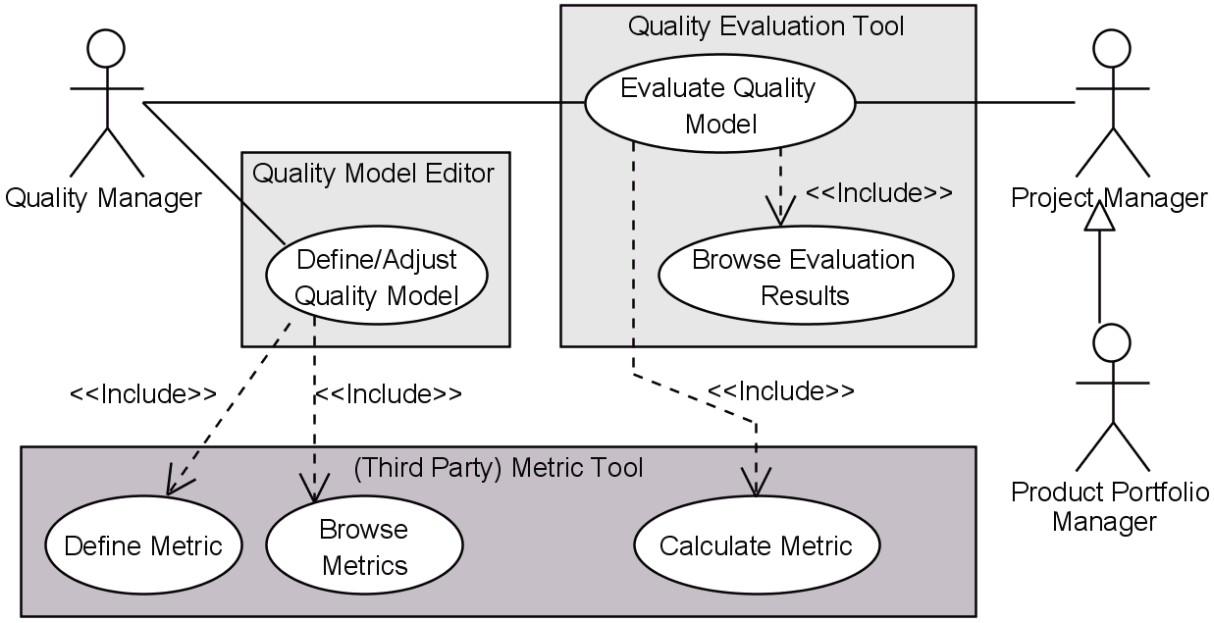

Figure 1: Use Case Diagram of the QMetric Tools

An overview of the user roles, related use cases and the different tools is given in Figure 1. Basically the following user roles and responsibilities can be distinguished:

- The **quality manager** is responsible for defining a quality model. This includes the identification of the quality characteristics of interest and their relationships, and developing and validating corresponding metrics to be used as quality indicators. Moreover the quality manager can tailor quality models to the needs of different projects.
- The **project manager** performs a quality assessment of his project based on a given quality model. The quality model needs to be completed with additional information, like a filter that specifies the project to be evaluated, the evaluation time span, and optionally additional information like empirical comparison data. Depending on the evaluation results, the project manager can decide on counter-measures, or adjust the project plan if necessary.
- The **Product portfolio manager** performs similar quality assessments for a set of related products. He is responsible for decisions related to the overall product or project portfolio, like decision on the resource allocation among the projects or process improvement measures. Both the project manager and the product portfolio manager have to be able to drill down the evaluation results to individual metric values.

The meta-model for quality models is described in the next section. The architecture of the solution is depicted in section 4.2.

4.1 A Meta-Model for Quality Assessment Models

The main concepts of the meta-model are shown in Figure 2. An instance of the meta-model (i.e. a quality model) is a directed acyclic graph (DAG). Source nodes of a quality model represent quality indicators. Inner nodes and sink nodes represent quality characteristics. **Quality Characteristics** represent high-level requirements on the quality. An example of a quality characteristic is planning precision which can be subdivided into the quality characteristics adherence to schedule, adherence to planned effort and process transparency. **Quality Indicators** are used to describe how quantitatively measurable attributes of an entity can be interpreted with respect to a quality characteristic.

Information like a rationale, the range of possible values, and guidelines for interpretation can be attached to each node (i.e. a quality characteristic or a quality indicator). Each node has an associated **ValueSpecificaton** which defines how the value for the node is determined.

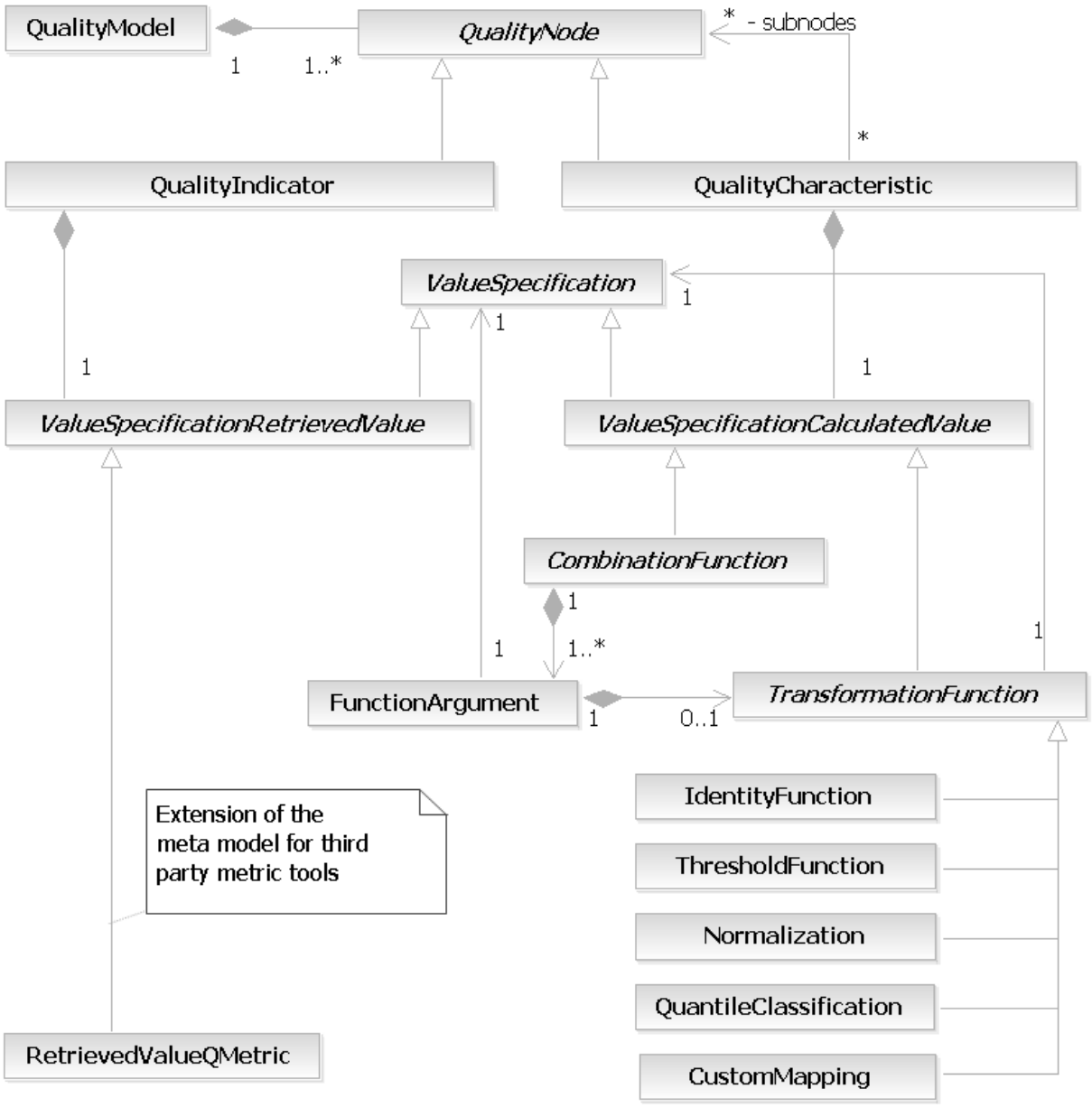

Figure 2: Meta-Model for Quality Assessment Models

A **QualityIndicator** contains a **ValueSpecificationRetrievedValue** that defines how a measurement value is retrieved from a metric tool. The concrete instantiation of a ValueSpecificationRetrievedValue depends on the underlying metric tool. When using metric evaluations on change requests given by the QMetric tool suite, the metric will be given as metric specification in a declarative language.

A **QualityCharacteristic** contains a **ValueSpecificationCalculatedValue** that defines how a value of the quality characteristic is calculated based on the values of the nodes connected by incoming edges as arguments. The ValueSpecification-CalculatedValue is either a unary function called TransformationFunction, or an n-ary CombinationFunction. Available **TransformationFunctions** are the identity function, a threshold function, normalization to a certain value interval, a user-

defined custom mapping, or the assignment of a value based on quantile classification with respect to a set of empiric values. A **CombinationFunction** is applied to combine the inputs from different incoming edges. Typically a linear equation is used to express different weighting of the inputs. Again the quantile classification can be applied for the result of this function.

The available functions enable to express a wide range of quality models. The fulfillment of an envisaged quality level can for example be modeled by using a threshold function at a sink node of a DAG. Different quality levels can then be modeled in sub graphs of the DAG with tightened thresholds in each level. However the approach is open to implement additional functions if required in the quality model.

The meta-model itself does not consider the structure of the entities of measurement. Instead the abstraction level of the entity of measurement is described in the underlying metric definitions of the quality indicators. Moreover they must be considered in the quality evaluation tool, as the concrete entity of measurement is only fully-specified, when the user has configured the evaluation.

4.2 Architecture

Corresponding to the user responsibilities, the tool is split into two main components, the Quality Model Editor and the Quality Evaluation Tool. An overview is given in Figure 3.

Third party metric tools can be integrated by providing two different adapters. The **Metric Definition Adapter** provides a list of available metric definitions. The Metric Calculation Adapter facilitates to trigger metric evaluations and retrieve the results from the third party metric tool.

The **Quality Model Editor** supports the creation of hierarchical quality models with a graphical editor (see Figure 4). A wizard is available that guides the user through the definition of a value specification for quality characteristics and quality indicators. In case of the ValueSpecificationRetrievedValue the user has to select between the available metric definitions given in a metric tool. The quality model is saved in the standardized XMI format. Thus it is open to be used in third-party tools.

The **Quality Evaluation Tool** is used to evaluate the process quality based on software process data with respect to a given quality model. In order to configure an evaluation the user has to specify:

- the quality model to be used,
- the entity to be evaluated,
- and optionally empirical comparison data to be used.

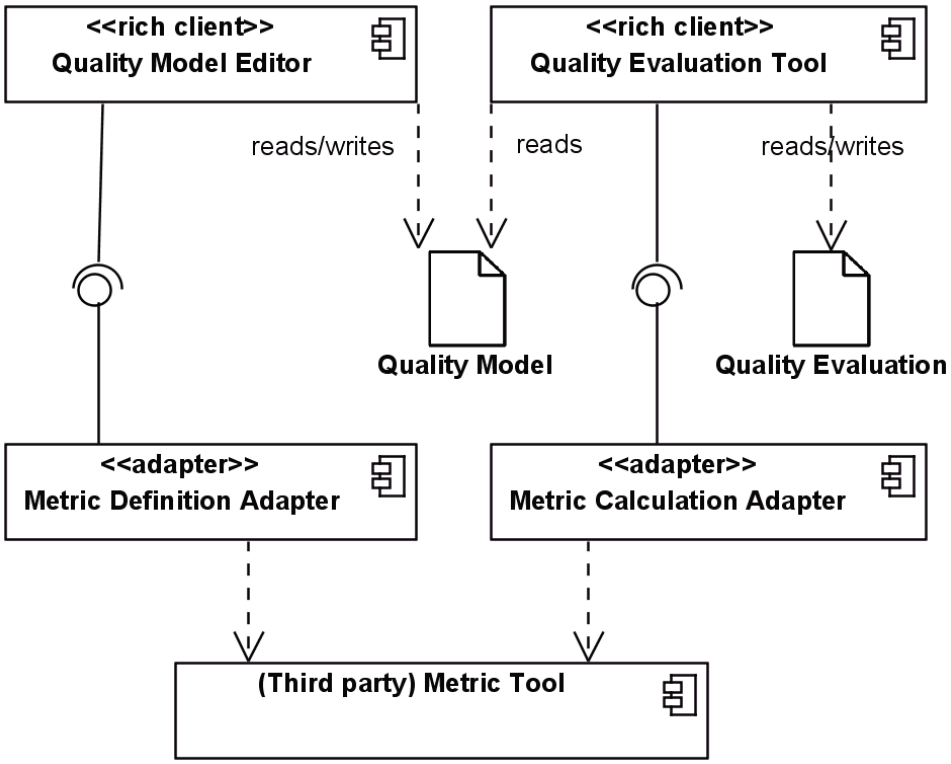

Figure 3: Architectural Overview

The configuration of the entity to be evaluated and the comparison data depend on the underlying metric tool. If evaluations are based on the QMetric tool for evaluation on change request databases, the entity to be evaluated will usually be the change request process of a product or project. It can be defined by specifying filter that matches the set of change requests related to this product or project. Further on the time span and the time granularity (e.g. month or year) for the entity of measurement must be defined.

Optionally the measurement values of the evaluated entity can be interpreted based on the value distribution in a peer group (e.g. a group of similar products). The required comparison data is defined by specifying a filter that describes the products of interest, and the time span and time granularity for comparison.

The Quality Evaluation Tool can automatically trigger all required metric calculations. For this purpose the metric definitions given in the quality model will be completed with information given in the configuration of the evaluation. The fully-specified metric definitions can then be evaluated using the metric tool. The configuration of the evaluation and the retrieved results are saved in a separate XMI file, decoupled from the quality model itself. Results can be browsed in a tree view (see Figure 5), drilled down to individual results for each time interval, and exported to CSV for usage in third-party tools. Resulting values for each quality characteristic and quality indicator can be visualized in line charts.

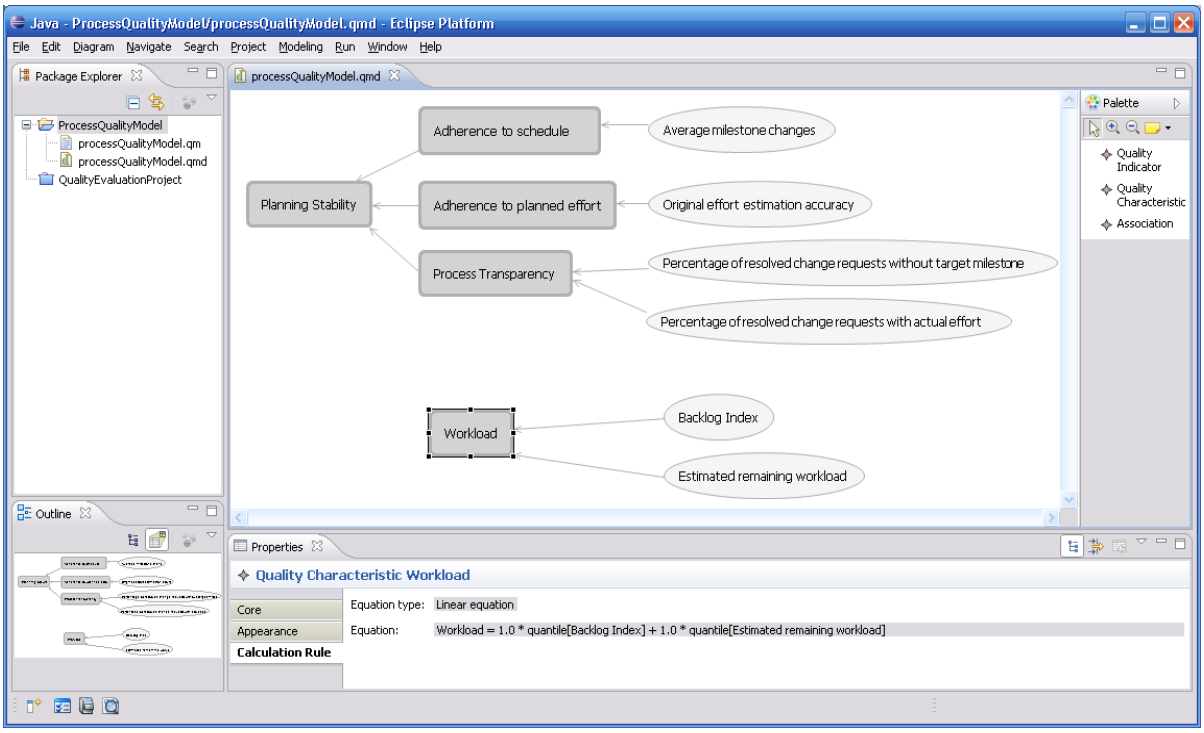

Figure 4: Quality Model Editor

Both the Quality Model Editor and the Quality Evaluation Tool are implemented as plug-ins to Eclipse. This facilitates to bundle these tools accommodated to different user roles (e.g. quality manager or product portfolio manager) as well as to provide both tools in a single rich-client application.

5 Experiences and Outlook

The tool has been applied for the definition of quality models for the assessment of process quality for the product portfolio of the open source projects GNOME [18], and Eclipse [19]. The quality model editor facilitated the intended aggregation of quality characteristics. Thus the expressiveness of the available aggregation functions had been sufficient. However there are some recurring combinations of quality nodes and related aggregation functions which could be made available as predefined building blocks.

The graphical presentation supports understandability of the quality model and the relations between quality characteristics and indicators. The experiences of these case studies helped to improve the usability of the wizards for the definition of value specifications. The evaluation tool simplifies quality assessment based on change request metrics, by automating many steps of the evaluation. Moreover it supports analysis and interpretation, by presenting the evaluation results with drill-down to individual metric values.

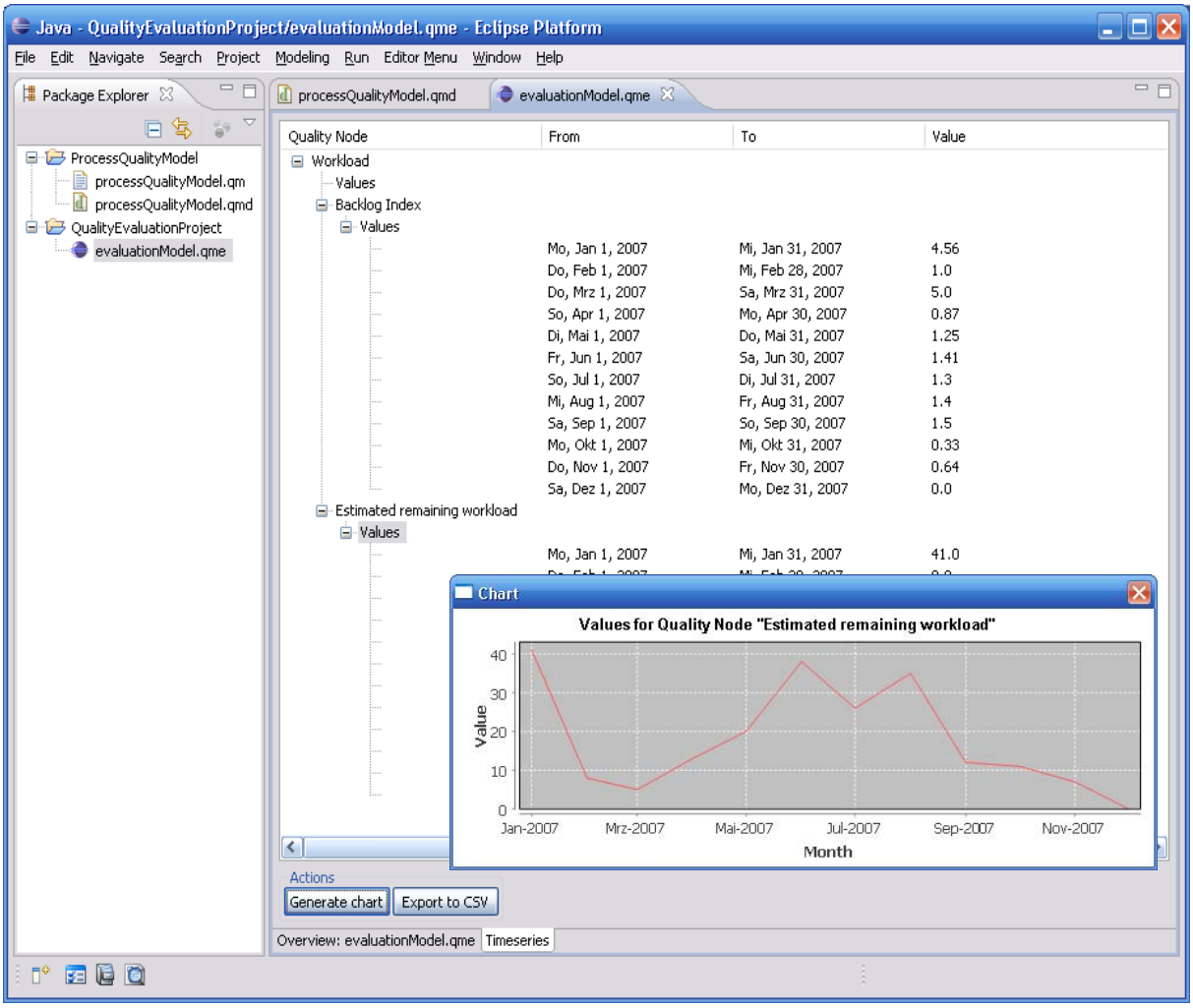

Figure 5: Quality Evaluation Tool – Display of Evaluation Results

Deissenboeck et al. formulate general requirements for quality assessment models and their meta-models [3]. In the following we list the relevant requirements and discuss the fulfillment by the presented meta-model.

- *The model shall not be limited to automatically measurable metrics.* In the first place the presented tool support is targeted at automatic measurements. However, it is possible to provide a Metric Calculation Adapter that retrieves metric results from manually evaluated values (e.g. given in a spreadsheet).
- *Metrics must be ideally supported by a description model.* The definition of metrics depends on the underlying metric tool. The QMetric tool suite uses metric definitions in a declarative language, which facilitates compact and precise definition of the metrics. When integrating third-party tools, the quality model must either include a precise description of the metric itself, or some reference to the metric as it is available in the metric tool.
- *The aggregation of quality characteristics and indicators must to be understandable by the assessor.* Having an explicit quality model with a set of predefined aggregation functions improves understandability of the aggregation.

Nevertheless understandability of a concrete quality model needs sufficient annotation of the quality characteristics with the underlying explanations.

- *An assessment model shall support the specification of different required quality profiles for different parts of the software.* The presented tool explicitly supports the tailoring of quality models. Thus different quality profiles can be defined by adaptations of a template quality model.

In order to fully assess the generality of the proposed meta-model it still remains an open task to integrate other metric tools, like tools for evaluating code quality metrics. The presented quality model editor and evaluation tool are available open source on *www.qmetric.org*.

References

1. Ebert, C., Dumke, R.: *Software Measurement. Establish – Extract – Evaluate - Execute*. Springer, Berlin, Heidelberg, 2007.
2. Schackmann, H, Jansen, M., Lischkowitz, C., Lichter, H.: *QMetric - A Metric Tool Suite for the Evaluation of Software Process Data*. Companion Proceedings of the 31th International Conference on Software Engineering (ICSE'09). Vancouver, Canada, May 16-24, 2009.
3. Deissenboeck, F., Juergens, E., Lochmann, K., Wagner, S.: *Software Quality Models: Purposes, Usage Scenarios and Requirements.* Proc. of the 7th International Workshop on Software Quality (WoSQ '09). Vancouver, Canada, May 16, 2009.
4. *ISO/IEC 9126 - Software Engineering - Product Quality*. International Organization for Standardization (ISO) / International Electrotechnical Commision (IEC). Genf, 2001.
5. Chulani, S., Boehm, B.: *Modeling Software Defect Introduction and Removal: COQUALMO (COnstructive QUALity MOdel)*. Technical Report USC-CSE-99-510, University of Southern California, Center for Software Engineering, 1999.
6. McCall, J.A., Richards, P.K., Walters, G.F.: *Factors in Software Quality, Vols. I-III*. AD/A-049-014/ 015/055. Springfield, VA, National Technical Information Service, 1977.
7. Washizaki, H., Namiki, R., Fukuoka, T., Harada, Y., Watanabe, H.: *A Framework for Measuring and Evaluating Program Source Code Quality*. In Proc. of the 8th international Conference on Product-Focused Software Process Improvement (Riga, Latvia, July 2-4, 2007). J. Münch and P. Abrahamsson, Eds. LNCS, vol. 4589, pp. 284-299, Springer, Berlin, Heidelberg, 2007.
8. Ploesch, R., Gruber, H., Pomberger, G., Saft, M., Schiffer, S.: *Tool Support for Expert-Centred Code Assessments*. In Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation (ICST 2008). Lillehammer, Norway, April 9-11, 2008. pp. 258-267, IEEE Computer Society, Washington, DC, 2008.
9. Lewerentz, C., Simon, F.: *A Product Metrics Tool Integrated into a Software Development Environment*. In Object-Oriented Technology. ECOOP '98 Workshop Reader

(Brussels, Belgium, July 20-24, 1998). S. Demeyer and J. Bosch, Eds. Lecture Notes In Computer Science, vol. 1543, pp. 256-260. Springer, London, 1998.

10. Simon, F., Seng, O., Mohaupt, T.: *Code Quality Management*. Dpunkt Verlag, Heidelberg, Germany, 2006.
11. Staron, M., Meding, W., Nilsson, C.: *A Framework for Developing Measurement Systems and its Industrial Evaluation*. Inf. Softw. Technol. 51, 4 (Apr. 2009), pp. 721-737, 2009.
12. Lawler, J., Kitchenham, B.: *Measurement Modeling Technology*. IEEE Software, vol. 20, no. 3, pp. 68-75, May/June 2003.
13. Kitchenham, B. A., Hughes, R. T., Linkman, S. G.: *Modeling Software Measurement Data*. IEEE Trans. Softw. Eng. 27, 9, pp. 788-804, Sep. 2001.
14. Deissenboeck, F., Juergens, E., Hummel, B., Wagner, S., Mas y Parareda, B., Pizka, M.: *Tool Support for Continuous Quality Control*. IEEE Software 2008, 25 (5), pp. 60-67, 2008.
15. Carvallo, J.P., Franch, X., Grau, G., Quer, C.: *QM: A Tool for Building Software Quality Models*. Proc. of the 12th IEEE International Requirements Engineering Conference (Kyoto, Japan, Sept. 6-10, 2004). pp. 358-359, 2004.
16. *ISO/IEC 15939 - Systems and Software Engineering – Measurement Process*. International Organization for Standardization (ISO) / International Electrotechnical Commision (IEC). Genf, 2007.
17. Coleman, D., Lowther, B., Oman, P.: *The Application of Software Maintainability Models in Industrial Software Systems*. J. Syst. Softw. 29, 1, pp. 3-16, Apr. 1995.
18. Schackmann, H., Lichter, H.: *Evaluating Process Quality in GNOME based on Change Request Data*. Working Conference on Mining Software Repositories (MSR'09). Vancouver, Canada, May 16-17, 2009.
19. Schackmann, H., Schaefer, H., Lichter, H.: *Evaluating Process Quality based on Change Request Data – An Empirical Study of the Eclipse Project*. In: International Conference on Software Measurement, Software Process and Product Measurement (IWSM/Mensura). Amsterdam, Netherlands 18-19, 2009.