Process Assessment by Evaluating Configuration and Change Request Management Systems

Holger Schackmann, Horst Lichter
RWTH Aachen University
Research Group Software Construction
Ahornstr. 55, 52074 Aachen, Germany
{schackmann,lichter}@swc.rwth-aachen.de

ABSTRACT

This paper presents an approach for assessing process qualities based on evaluating metrics on change request and configuration management systems. It is based on user-defined quality models to enable quality evaluations customized to the information needs of an organization. Further on the concept of declarative metric specifications is introduced, which enables a precise definition of metrics on an appropriate abstraction level. With the corresponding tool support given in the QMetric tool suite, this concept simplifies development and validation of the metrics needed for quality evaluations.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics.*

General Terms

Measurement, Management

Keywords

Quality Modeling, Declarative Metric Specification, Mining Software Repositories, Software Product Management

1. INTRODUCTION

Managing a large portfolio of software products requires continuous monitoring of project status and process quality. Collecting the required data by regularly status reporting can be expensive and intrusive and furthermore ignores the past history of the process [1]. This motivates mining data from routinely collected **software repositories** like change request management (CRM) or software configuration management (SCM) systems. Naturally this data does not provide a holistic view on the development process, but it offers valuable information to assess certain characteristics of the process.

ICSE-WUP'09: The Warm-Up Workshop for ACM/IEEE ICSE 2010, April 1-3, 2009, Strand, Cape Town, South Africa.

Currently the historic data available in these systems is only used in a limited way for evaluating process qualities. Existing change request management systems usually provide a number of fixed metric evaluations [2]. Metrics appropriate for organization-specific information needs must be implemented in custom scripts [3]. There also exists a number of tools for generating metrics or visualizations based on version control systems [4][5][6]. However there is no generalized approach on assessing process quality characteristics on a higher level based on data available in software repositories. In order to provide such an approach one has to face several challenges:

How to relate higher level quality characteristics to metrics? Quality characteristics of interest are in general derived from the objectives of the organization [7]. Change request management systems are typically customized to organization specific needs. Hence appropriate metrics for evaluating quality characteristics depend on the designated process and the data available. Moreover guidance must be provided on how to interpret resulting measurement values with respect to a quality characteristic. Thus a conceptual base is needed on how to model the relationship between quality characteristics and underlying metrics.

How to develop and validate metrics? Approaches like GQM [8] provide a general framework for deriving metrics. However specifying a metric in detail bears many pitfalls due to the complexity of the underlying objects of measurement [9]. Hence a systematic procedure for developing and validating metrics is required.

How to mine software repositories in a flexible way? Apart from the methodological questions appropriate tool support for metric evaluation must be available. In order to offer a general approach that is applicable independent from a specific tool infrastructure, it is necessary to collect metrics from any common CRM and SCM system.

Typically tools are targeted at a single source of information (e.g. a specific change request management system), and provide only a number of fixed metric evaluations with limited adaptability [2]. Metric tools for SCM systems do not consider traceability links between change requests and changes in the source code. Developing custom scripts for the required metric evaluations is time-consuming and costly. Hence a more generic approach for metric evaluation is needed that can adapt to different underlying systems and supports a wide range of metric definitions.

Warm Up Workshop for ACM/IEEE ICSE 2010 (WUP '09), Nenad Medvidovic and Tetsuo Tamai (Eds.). ACM, New York, NY, USA, 37-40
DOI=10.1145/1527033.1527046

This paper presents concepts that are targeted to tackle these questions, and describes a respective approach for assessing process qualities based on mining CRM and SCM systems. Moreover it will be pointed out how these concepts had been implemented in a tool suite called **QMetric**, which provides a generic metric calculation engine, and an editor and evaluation tool for user-defined quality models.

2. USER-DEFINED QUALITY MODELS

In order to relate metrics to improvement goals we use hierarchical quality models that lean on the approach of bidirectional quality models introduced by Simon et. al [10]. This section first introduces the related terminology (see Figure 1). Then it is discussed how individual measurement values can be interpreted and aggregated with respect to higher level quality characteristics.

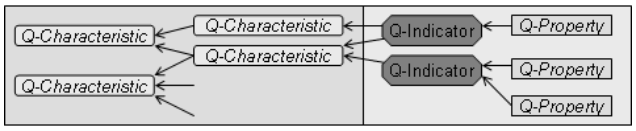

Figure 1. Hierarchical quality model

On the one side the **quality characteristics** reflect high-level requirements on the quality. In terms of the ISO/IEC 15939 standard these quality characteristics correspond with information needs derived from the business, organizational, regulatory, product or project objectives [11]. 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.

On the other side the **quality properties** denote objective attributes of an entity (i.e. product, process, or system), that can be used to distinct between the considered entities, and can be objectively and quantitatively distinguished by automated means. Examples for such quality properties are the total number of defects, the frequency of assignee changes of a change request, or the number of reopened change requests.

The quality properties will be used in a bottom-up fashion to form **quality indicators**. A quality indicator describes how a number of quality properties can be interpreted with respect to a quality characteristic. Hence the quality indicators bridge the gap between the technical view of quality properties and the abstract view of the quality characteristics.

The question is how the measurement values of quality properties can be aggregated to high-level quality characteristics. The target is to provide a general approach for quality models for evaluating process quality based on mining software repository data. Existing approaches in the area of metric-based quality evaluation (e.g. the evaluation of internal software quality based on code metrics) are typically based on a fixed quality model with limited adaptability (e.g. adjusting weights of the quality indicators).

In our view the evaluation of process quality needs a more flexible approach for quality models due to organization-specific improvement goals, and due to the heterogeneity of the underlying metrics. In the following we will describe the concepts for user-defined quality models that are implemented in the QMetric tool suite.

In general the model is a directed acyclic graph (DAG). Source nodes of the quality model represent quality indicators. Quality properties are defined using a declarative metric specification (see section 3), that defines how metric values are retrieved from underlying CRM and SCM systems. Hence a metric specification must be defined for each quality indicator.

Inner nodes and sink nodes of the quality model represent quality characteristics. For each quality characteristic a value specification must be defined that describes how a value of the quality characteristic is calculated based on the values of the nodes connected by incoming edges as arguments. The value specification is defined as follows: A unary function is applied to each argument. Available functions 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. Then a second function 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 quantile classification can be used to guide interpretation of the results according to the value distribution in a peer group of measured entities. 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 model is open to implement additional functions if required. Further development of respective models in our ongoing case studies will show weather the building blocks listed above offer sufficient expressiveness.

An evaluation based on the quality model enables the systematic comparison of the process quality in a certain time interval to earlier time intervals, and the analysis of the development processes in a portfolio of software products.

3. DECLARATIVE METRIC SPECIFICATIONS

One of the basic ideas of our approach is that the developer of a metric should concentrate on the model of a metric, not on the way how the metric is calculated from underlying software repositories. This is realized by a declarative language for metric specifications that abstracts from the way the information is stored in specific CRM or SCM systems [2].

The basic building blocks for these specifications are filters for information fields of a change request (e.g. its severity, status, or target milestone), and events that occur in the history of a change request (e.g. change of the assignee, committing related code, or reopening a resolved request). Filters and events can be combined with Boolean operators.

Each metric specification contains a **base filter** that defines which change requests are considered during the calculation (e.g. only change requests that belong to a certain product). Further on the evaluation time period and the time granularity (e.g. month or year) are defined.

Then one of several predefined **value calculators** can be applied to calculate a value for individual change requests in each time

Warm Up Workshop for ACM/IEEE ICSE 2010 (WUP '09), Nenad Medvidovic and Tetsuo Tamai (Eds.). ACM, New York, NY, USA, 37-40
DOI=10.1145/1527033.1527046

interval according to the given time granularity. Examples of value calculators are the calculation of the length of a time interval between two specified events in the lifecycle of a change request, the calculation of the time a change request resides in a certain state, or the calculation of the number of occurrences of certain events during a time period. In the latter case an optional weight can be applied (e.g. a weighting by the severity of the change request, or by its estimated remaining workload).

The outcome of these value calculators can be combined with operations like sum, maximum, or mean value to calculate a result for a certain time interval. This approach offers a large flexibility for the specification of metrics. Furthermore the metric specification is separated from the way the required information is retrieved.

The concept is implemented in the generic metric calculation engine of the QMetric tool suite [12]. By design of the tool the access to data sources (i.e. concrete CRM or VCS systems) is separated from the metric evaluation algorithm. It operates on abstract fields that are provided by wrappers for the underlying data sources. Currently such wrappers had been implemented for the CRM systems Bugzilla and Mantis, and for CVS and Subversion [13]. Wrappers for CVS and Subversion are based on the Scmbug tool which offers a generic solution to link changes in a software configuration management system to related change requests [14]. This does not only enable to define metrics that consider SCM events (e.g. commit changes to files, add a branch) and size information (e.g. size of a code change) but also metrics that combine information from CRM and SCM systems. Moreover the QMetric tool is designed to enable extensions of the evaluation algorithm, like the evaluation of additional fields in a customized change request tool, or the extension with new value calculators and weights.

4. DEVELOPING AND VALIDATING METRICS

The usage declarative metric specifications with appropriate tool support leverages the level of abstraction when developing metric definitions. Precise description in a declarative language improves communication on the metric definition. Thus experimenting with metrics and adjusting them is faster and easier.

But, first experience has revealed certain pitfalls during the development of metric definitions [2]. Typical examples are considering not all relevant events related to the intended metric, improper interpretation of the status workflow, or deprecated data in a change request system due to inconsistent usage of input fields. This is the motivation for a structured approach for developing metrics on CRM and SCM that had initially been presented together with a case study on process quality in the Eclipse project [9]. We will briefly summarize the steps performed for the development of a metric:

1. Deriving of process **quality characteristics** from the objectives of the organization [7]. These characteristics can be refined stepwise.
2. Improvement goal based identification of corresponding **quality properties**: In order to identify measurable quality properties it is necessary to analyze the way the CRM and SCM systems are used, e.g. it must be examined what is the typical workflow of a change request, and which information is collected on a change request. Then quality properties need to be defined where some relation to the quality characteristics is conjectured. The plausibility of the metric can then be validated by inspecting the results calculated for individual change requests and examining whether the history of a request conforms to the envisaged interpretation.
3. Definition of **quality indicators** that enable comparability between projects: The quality indicator must define how measurement values related to individual change requests (e.g. time until a change request is resolved, or granularity of related changes in the SCM system) can be aggregated. An appropriate quality indicator must eliminate interfering factors like age and size of a project, and it must be ensured that the assignment of the measurement values to time intervals stands in a temporal connection to potential causes in the process in order to prevent misleading interpretations. The QMetric tool suite provides a number of constructs in the metric specifications that facilitate different kinds of normalization of the metric results (e.g. counting the percentage of change request whose residence time in a status of the workflow hits a certain threshold, instead of using the average residence time). Again these indicators can be included as aggregated calculation in a declarative metric specification. The results can then be validated by comparing metric results of projects to expert assessments of the process of these projects.

The proposed procedure facilitates an iterative refinement of metric definitions and enables to detect problems early due to the stepwise validation.

5. EXPERIENCES AND OUTLOOK

Applicability of the approach to quality modeling was evaluated in a case study which analyzes the quality of the change request process in different Eclipse projects [9].

The tool BugzillaMetrics which encompasses the QMetric evaluation engine and a web-based metric query tool that provides wizards for defining metric specifications on a graphical user interface, had been published open source. BugzillaMetrics has found a community of users, which points out usability of declarative specifications, as well as the practical relevance of the approach. The main characteristics of the QMetric tool suite are the following:

- General infrastructure for the evaluation of metrics on software repositories data like CRM and SCM systems.
- Flexible tool support for the definition of quality models and automatic evaluation based on software metrics.

Based on these results a quality model for open source projects is currently being developed that is oriented at typical goals of established open source projects, like user involvement and planning stability. This quality model distinguishes several quality levels of the change request process in an open source environment. Naturally it is not possible to achieve a holistic evaluation, since not every aspect of the process is reflected in the software repositories. Such a model would be complementary to manual approaches for assessing the maturity of open source projects [15][16].

Warm Up Workshop for ACM/IEEE ICSE 2010 (WUP '09), Nenad Medvidovic and Tetsuo Tamai (Eds.). ACM, New York, NY, USA, 37-40
DOI=10.1145/1527033.1527046

In our ongoing work we apply the approach for analyzing the development process of an industrial partner. The main targets are the following:

- Improved transparency in a large software product portfolio in order to support planning and resource allocation.
- Identification of development process weaknesses and assessment of changes in the process.

Hence it needs to be analyzed how the information needs of different roles (e.g. project manager or product portfolio manager) can be reflected in quality models, and how the evaluation results can be visualized in an understandable way. This extended study will help to evaluate the benefits and limitations of the proposed approach in practice.

6. ACKNOWLEDGMENTS

We would like to thank Kisters AG, Aachen for supporting the development of BugzillaMetrics.

7. REFERENCES

[1] Cook, J. E., Votta, L. G., and Wolf, A. L. 1998. Cost-Effective Analysis of In-Place Software Processes. *IEEE Trans. Softw. Eng.* 24, 8 (Aug. 1998), 650-663. DOI= http://dx.doi.org/10.1109/32.707700

[2] Grammel, L., Schackmann, H., and Lichter, H. 2007. BugzillaMetrics: an adaptable tool for evaluating metric specifications on change requests. In *Ninth international Workshop on Principles of Software Evolution: in Conjunction with the 6th ESEC/FSE Joint Meeting* (Dubrovnik, Croatia, September 03 - 04, 2007). IWPSE '07. ACM, New York, NY, 35-38. DOI= http://doi.acm.org/10.1145/1294904.1294909

[3] Kanat-Alexander, M. 2008. The Bugzilla Survey – August 2008. https://wiki.mozilla.org/Bugzilla:Survey

[4] Gall, H. C. and Lanza, M. 2006. Software evolution: analysis and visualization. In *Proceedings of the 28th international Conference on Software Engineering* (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM, New York, NY, 1055-1056. DOI= http://doi.acm.org/10.1145/1134285.1134502

[5] Kagdi, H., Collard, M. L., and Maletic, J. I. 2007. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software Maintenance and Evoution.*. 19, 2 (Mar. 2007), 77-131. DOI= http://dx.doi.org/10.1002/smr.344

[6] Draheim, D. and Pekacki, L. 2003. Process-Centric Analytical Processing of Version Control Data. In *Proceedings of the 6th international Workshop on Principles of Software Evolution* (September 01 - 02, 2003). IWPSE. IEEE Computer Society, Washington, DC, 131.

[7] Ebert, C. and Dumke, R. 2007. Software Measurement: Establish - Extract - Evaluate - Execute. Springer Verlag, Berlin.

[8] Basili, V., Caldiera, G. and Rombach, H.D. 1994. The Goal Question Metric Paradigm. In: Encyclopedia of Software Engineering. John Wiley & Sons, 528-532.

[9] Schackmann, H. and Lichter, H. 2008. Comparison of Process Quality Characteristics Based on Change Request Data. In *Proceedings of the international Conferences on Software Process and Product Measurement* (Munich, Germany, November 18 - 19, 2008). R. R. Dumke et al., Eds. Lecture Notes In Computer Science, vol. 5338. Springer-Verlag, Berlin, Heidelberg, 127-140. DOI= http://dx.doi.org/10.1007/978-3-540-89403-2_12

[10] Simon, F., Seng, O. and Mohaupt, T. 2006. Code Quality Management, Dpunkt-Verlag, Heidelberg.

[11] ISO/IEC 15939 Systems and software engineering – Measurement Process, ISO, Geneva, 2007.

[12] Schackmann, H., Jansen, M., Lischkowitz, C. and Lichter, H. 2009. QMetric - A Metric Tool Suite for the Evaluation of Software Process Data. In *Companion Proceedings of the 31th international Conference on Software Engineering* (Vancouver, Canada, May 16-22, 2009) ICSE'09, ACM, New York, NY.

[13] BugzillaMetrics project, www.bugzillametrics.org

[14] Makris, K., Ryu, K.D. 2005. Scmbug: policy-based integration of software configuration management with bug-tracking. USENIX Annual Technical Conference, USENIX Association, Berkeley, CA, 11-22.

[15] Ciolkowski, M. and Soto, M. 2008. Towards a Comprehensive Approach for Assessing Open Source Projects. In *Proceedings of the international Conferences on Software Process and Product Measurement* (Munich, Germany, November 18 - 19, 2008). R. R. Dumke et al., Eds. Lecture Notes In Computer Science, vol. 5338. Springer-Verlag, Berlin, Heidelberg, 316-330. DOI= http://dx.doi.org/10.1007/978-3-540-89403-2_26

[16] Golden B.: Open Source Maturity Model © Navica, http://www.navicasoft.com/pages/osmmoverview.htm

Warm Up Workshop for ACM/IEEE ICSE 2010 (WUP '09), Nenad Medvidovic and Tetsuo Tamai (Eds.). ACM, New York, NY, USA, 37-40
DOI=10.1145/1527033.1527046