ALIGNMENT OF OPEN SOURCE TOOLS WITH THE NEW ISO 25010 STANDARD
Focus on Maintainability

Emanuel Irrazábal, Javier Garzás
University Rey Juan Carlos, Madrid, Spain
Kybele Consulting S. L., Madrid, Spain

Esperanza Marcos
University Rey Juan Carlos, Madrid, Spain

Keywords: ISO 25010, Maintainability, Tools, Product metrics.

Abstract: Nowadays, quality and especially software product quality is becoming a hot topic in the context of Software Engineering. In this context, measures provide the basis for the improvement and, in particular, code measures provide direct visibility of product quality. Nevertheless, some organizations cannot afford these tools. Several studies have revealed that applying quality models in Small and Medium-Sized enterprises (SMEs) is a very challenging task. Related to this, open-source tools emerge as the answer to provide with the technical support to collect the information needed to assess the quality of software assets. In this work, we review how existing open-source tools fulfill the needs for quality measures raised when you want to assess product quality according to the new ISO/IEC 25010 standard, which has introduced changes in the characteristics of model quality over the previous model. We have focused on the characteristic of maintainability because of its historical significance and its direct impact on total costs.

1 INTRODUCTION

Quality is currently recognized by companies as a major asset to improve their competitiveness. At the same time it has become an essential requirement to ensure that the processes and products developed meet the needs of the customer. Although software quality can be described according to several points of view, when it comes to software development, quality is traditionally related to product quality and process quality (Ebert 2009).

As a response, different reference models for product quality assessment have appeared. In particular, the International Organization for Standardization (ISO) works in a new quality model, the ISO/IEC 25010 (ISO/IEC JTC 1 2011) that revises the standard ISO/IE 9126:2001, Software engineering – Product quality (ISO 2001).

This new standard is currently published (since March 2011) and defines two quality models: a quality in use model and a product quality model. Basically, the product quality model identifies quality characteristics and how they are decomposed into subcharacteristics. In this work we focus on one of them: maintainability, since it has been historically recognised as one of the most relevant issues given its direct impact over the cost of software development and maintenance. Previous studies point out at maintenance as the most expensive phase along the product lifecycle, involving twice the development costs (Pressman, 2002); (Saiedian and Carr 1997).

Measures provide the basis for the improvement and, in particular, code measures provide direct visibility of product quality. Taking these assumptions into consideration, the most suitable approach seems to be the use of commercial tools for the automatic acquisition of such measures (Pagano, 2006). Nevertheless, some organizations cannot afford these tools. Several studies have revealed that applying quality models in Small and Medium-Sized enterprises (SMEs) is a very challenging task (Saiedian and Carr 1997); (Staples et al. 2007) since it implies huge investments in terms of money, time and human resources. This type of organizations needs from adapted Software Engineering practices to fit their size and the nature of their business (Dyba, 2005).

All this given, this work reviews the technical support that existing open-source tools for code analysis offer to the metrics that need to be collected to measure the maintainability subcharacteristics specified by the ISO 25010 standard. This work is partially carried out based on another work (Irrazábal, 2010) made on the basis of ISO 9126 standard and taking into account only a selection of tools found in an open source software repository.

The remainder of the paper is organized as follows. Section 2 presents a summary of the ISO 25010 standard, section 3 presents a summary of the basic metrics proposed by the main measurement models that can be found currently. Section 4 details the tools chosen to measure the basic metrics and explains the selection criteria. In Section 5, it has been performed a qualitative analysis of the found tools. Finally, Section 6 summarizes the main findings of the study.

2 THE MAINTAINABILITY CHARACTERISTIC IN ISO 25010

Before going deeply into this work, we first provide with a brief overview of the maintainability characteristic in ISO 25010 product quality model, the importance of this characteristic on the quality in use model and the amendments respect of the ISO 9126 maintainability characteristic.

The main objective of ISO 25010 is to provide a framework for defining and evaluating the quality of software. To accomplish this, the standard defines two models.

The first is the product quality model that provides a set of quality characteristics relevant and related to static properties of software and dynamic properties of the computer system. To that end, it identifies eight characteristics that compose models to assess product quality (listed above).

One of these is the maintainability, the most interesting for this paper. Here we want to know how easy it would be to modify the software product. In some sense, it is a measure of the effort required to correct a defect or make a change that preserves software functionality. Obviously, software is maintainable if it is easy to understand and test (Heitlager, Kuipers & Visser 2007). Each characteristic in the ISO 25010 product quality model is composed of a set of related subcharacteristic. In this case, the maintainability is composed of the following four subcharacteristics: analyzability, modifiability, testability and reusability.

To summarize, the value for the maintainability subcharacteristics (affecting both models described above) should be obtained by computing a set of measurement functions over the capability of the software. And this capability is determined by a set of static internal properties that can be measured. Nevertheless, nothing it is said about how these functions should be defined, which ones should be the elements to compute and how the values of them have to be gathered.

3 CHOICE OF BASIC MAINTAINABILITY METRICS

This section summarizes the most important measurement models because of the impossibility of finding a single measurement model sufficiently recognized and that details the basic recommended set of metrics to measure maintainability.

Various measurement models have been collected. We have added an acronym after each reference so that it can easily be named in Table 1.
One of the studies that can be considered more interesting in this regard is (Riaz 2009)[M1]. This is a systematic review that collects evidence on software maintainability prediction and metrics. The search has been made again. Regarding the search, string has made customizations, adding two new words: reusability and modifiability. This has been one of the sources that will yield maintainability measurement models. Another source has been the direct search in online databases like ACM Digital Library, Springer Link, IEEE Xplore, Science Direct, etc.

The encountered models relate a set of quality metrics obtained from source code with the maintainability subcharacteristics. We are not taken into account those models directly related to the characteristic of maintainability. This made impossible to relate the metrics to the different subcharacteristic. Yet, many of these metrics can be found in Table 1, named by other models.

Main models found were as follows: Rüdiger Lincke suggests a software quality model for the maintainability characteristic and indicate some basic metrics associated. (Luijten, Visser & Zaidman 2010) [M2]. In (Alshayeb 2009) [M3] reusability is associated with a subset of metrics. Another complete maintainability measurement model is described in (Heitlager, Kuipers & Visser 2007) [M4]. In this paper, authors mapped maintainability subcharacteristics (based on ISO 9126) onto source code properties, such as volume, complexity, duplication, unit length and number of modules.

In (Mouchawrab, Briand & Labiche 2005) [M5] a generic measurement framework for object-oriented software testability has been presented. Ioannis Samoladas presents a hierarchical quality model that evaluates source code and community processes (Samoladas et al. 2008) [M6].

Results are summarized in Table 1, which presents the basic set of internal metrics that are related to the subcharacteristics: analyzability, modifiability, testability and reusability. It is noteworthy to see how many of the measurement models associated with maintainability do not directly consider the reusability. That is why we have opted to seek metric articles (Barnard 1998)[M7](Washizaki, Yamamoto & Fukazawa 2003)[M8] dealing directly with basic reusability metrics.

Metrics acronyms have been added to facilitate the construction of the subsequent tables.

4 CHOICE OF TOOLS

Given the summary of the basic metrics obtained in the previous section, we have chosen the open-source tools that can analyze the software product code. Once found the open source tools related to the static analysis of source code and related to the maintainability we analyze the degree of contribution of the metrics that are obtained.

It has followed the approach used in (Wangenheim 2009), reviewing the repositories of existing open source projects and selecting the main repository to be examined. The search was supplemented by two other sources. The former is the selection of tools shown in similar studies and recommendations derived from quality managers. The latter source has been obtained from interviews with a group of companies involved in a project to improve its development process, including measurement of indicators of the software product.

4.1 Primary Tool Selection

According to previous studies (Wangenheim 2009) the SourceForge repository currently hosts more open source project compared to Codeplex, Google Code or Kenai. Then, we have identified the typical

Table 1: Summary of internal metrics related with the maintainability characteristic according to ISO 25010.

Internal metric	Analyzability	Modifiability	Reusability	Testability
Cyclomatic complexity (CC)	M6	M2,M4	M3	M2,M4,M5, M6
Lines of code (LOC)	M2,M4		M3,M7	M2,M4,M5
Average size of statements (SS)	M6	M2,M6		M2,M5
Comments frecuency(CF)	M6		M7	
Weighted methods per class (WMC)	M6		M8	
Number of base classes (NBC)	M6			
Number of unconditionals jumps (NUJ)		M6		M6
Number of nested level (NNL)		M6		M6
Coupling between objects (CBO)	M1	M1,M6	M7, M8	M1,M5
Lack of cohesion of methods (LCOM)	M1	M1,M6	M7,M8	M1,M5
Depth of inheritance tree (DIT)	M1	M1,M6	M7,M8	M1,M5,M6
Directly called components (DCC)		M6		
Number of children (NOC)	M1	M1,M6		M1,M5,
Number of exits of conditional structs (ECS)				M6
Response for a class (RFC)				M5,M6
Number of methods (NOM)	M1	M1	M7	M1,M5
Number of attributes (NOA)			M7	
Unit test coverage (UTC)				M4,M5
Unit test errors (UTE)				M4,M5
Cyclic dependencies (CDC)	M6	M2		
Efferent coupling (Ce)		M2		
Afferent coupling, (Ca)		M2		
Duplicated blocks (DB)	M2,M4	M2,M4		

words which conduct the search in the repository and obtain the measurement tools most valued by users. According to the reading of the ISO 25010 standard, a set of keywords used for searching the Web site www.sourceforge.net has been adopted. Tools have been identified associated with the languages. NET (VB.NET and C # adding), Java and C / C + +, since these are the most popular development frameworks according to the TIOBE Programming Community Index, updated in January 2011. To complete the previous selection, we include additional groups of tools. First, we consider also those freeware tools for .NET static code analysis boosted by Microsoft (FxCop and StyleCop). Even so, it is clear that they are not currently used separately and that they are already included in the .Net IDE.

Coupling and cyclomatic complexity have been widely acknowledged as the most valuable metrics to assess maintainability of object-oriented systems (Li, S. 1993). Thus, we were also interested in reviewing open-source projects that support the extraction of such metrics. Although some of the previous tools are able of returning the values for such metrics, they are not focused on analyzing coupling and cyclomatic complexity. In particular, we looked at JavaNCSS y JDepend as the most representative. However, we could have included any other similar tools instead of these ones.

Finally, we include other free tools analyzed in other studies (Lincke, Lundberg & Löwe 2008)(Plösch et al. 2009). Table 3 lists all tools selected and it specifies the basic metrics that can be obtained on this basis. In this way, with these open source tools, companies (especially SMEs) can measure aspects of maintainability according to ISO 25010.

4.2 Assessments by Experts

Between the second half of 2009 and during 2010, we have conducted initial assessments in the development process in more than 20 software companies. The aim of these reviews has been providing companies with a snapshot of the current status of their processes, practices and tools used in the development process. At the end of the evaluations we have informed the companies about the missing features and activities needed to improve its development process.

Quality managers and software developers, as part of the company evaluation, have been interviewed to collect information about code static analysis and testing tools used on its projects. Results show that there is no extensive use of this kind of tools. The data collection has taken into account the tool list presented in Table 3.

According to the obtained results, there are two main types of tools that companies implement.

First, those associated to the analysis of the source code and the search for potential problems (PMD, FxCop). However, the style analysis tools closely related to the ones mentioned above are not used widespread. This is due to the big amount of default rules of this type of tools. That forces companies to invest a great amount of time and resources tailoring their own rule set. Likewise, did not have a standard code convention manual, what made the successful implementation of this practice very complex (even in an automated way).

The second type of more used tools has been those associated to unit tests. However, unit test practice was not completely widely used. Only some specific tests were executed, according to some client needs. Although coverage was taken into account, neither a deep analysis was applied nor any decisions were taken.

5 QUALITATIVE ALIGNMENT OF THE TOOLS

This section aims at putting forward clarifications and contributions from some of the tools mentioned in the previous section. To this end it is concluded that some of the tools and metrics could measure additional subcharacteristics already discussed.

5.1 Java NCSS

JavaNCSS supports tree different measures gathered over JAVA source code, namely the Non-Commenting Source Statements (NCSS), the Lines Of Code (LOC) and the Cyclomatic Complexity (CC). We can look at these measures as primary support for all subcharacteristics. So this is an indispensable tool for measurement of maintainability. NCSS provides a good indicator on how easy it would be to understand or follow the source code under evaluation. As well, NCSS is related with testability since the number of non-commented statements can be used as a clue towards understanding if the code is easy to test. On the other hand, the relationship of analyzability with CC is out of doubt. In fact, the number of cycles has been traditionally accepted as a measure to assess the complexity of any software component. Therefore, the lower the value of CC, the higher the analyzability of the component (Heitlager, Kuipers & Visser 2007).

5.2 PMD

We have analysed how the different rulesets that drive the static analysis that PMD carries out are aligned with ISO 25010 maintainability subcharacteristics. This analysis is based on the study of the content of each rule and our own experience using the tool. Just 5 to 10 per cent of the possible violations are directly related with modifiability or testability, we have considered that the tool does present partial alignment with the corresponding subcharacteristic.

In particular, by studying the PMD documentation, we can argue that 16 out of the 20 rulesets are directly related with evaluating the clarity of the code under review. Thus, we can state that PMD is clearly aligned with analyzability.

Finally, one the rulesset is directly related with the testability subcharacteristic. In particular the one that deals with the different problems that can occur with JUnit tests

5.3 Junit, CPPUnit, NUnit and EMMA

Here we group together the tools focused on testing. In particular, Junit, CPPUnit and NUnit perform unit tests while EMMA focuses in code coverage.

Obviously, being focused on testing, all these tools are fully aligned with the testability subcharacteristic. Furthermore, since unit tests have been traditionally acknowledged as an efficient way of documenting source code and assessing its level of stability (Heitlager, Kuipers & Visser 2007), we can state that these tools are partially aligned with analyzability and modifiability.

6 CONCLUSIONS

After analyzing the results it has been concluded that current open source tools serve as a good starting point towards achieving the indicators needed to support different measurement model that implement the product quality model according to the new ISO 25010 standard. In fact these tools, especially their implementations as IDE plugins can improve maintenance task.

Not all tools obtain the same results (Lincke, Lundberg & Löwe 2008) and are equally reliable in terms of false positives and false negatives when looking for violations in the source code (Plösch et al. 2009), making indispensable the task of tool selection and analysis. It stands for CodePro AnalytiX tool, which assesses source code written in Java, as one of the most complete in terms of number of basic metrics obtained. Still, it is advisable to check before accuracy of these tools in real cases, especially those that perform more complex calculations.

Maintainability is important both from the point of view of the software product and maintenance tasks during the development phase. Even though, tools that assist diagnosis are not used. However, companies carry out maintenance tasks based on developers' experience (whenever there is time).

As we have tried to demonstrate in this work, open source tools have the capacity to implement maintainability measurement models. Therefore, these tools will help to the software product maintenance.

ACKNOWLEDGEMENTS

This work has been partially funded by the Centre for Technological-Industrial Development (CDTI) (IDI-20090175), dependent on the Spanish Ministry of Science and Innovation and the MODEL-CAOS project (TIN2008-03582/TIN) financed by the Spanish Ministry of Education and Science (TIN2005-00010/).

REFERENCES

Alshayeb, M. 2009, "Empirical investigation of refactoring effect on software quality", *Information and Software Technology,* vol. 51, no. 9, pp. 1319-1326.

Barnard, J. 1998, "A new reusability metric for object-oriented software", *Software Quality Journal,* vol. 7, no. 1, pp. 35-50.

Dyba, T. 2005, "An empirical investigation of the key factors for success in software process improvement", *Software Engineering, IEEE Transactions on,* vol. 31, no. 5, pp. 410-424.

Ebert, C. 2009, "Guest Editor's Introduction: How Open Source Tools Can Benefit Industry", *IEEE Software,* vol. 26, pp. 50-51.

Feitelson, D. G., Heller, G. Z. & Schach, S. R. 2006, "An empirically-based criterion for determining the success of an open-source project", *Software Engineering Conference, 2006. Australian,*IEEE, pp. 6.

Heitlager, I., Kuipers, T. & Visser, J. 2007, "A Practical

Model for Measuring Maintainability", *Quality of Information and Communications Technology, 2007. QUATIC 2007.*IEEE, Lisbon, pp. 30-39.

Irrazábal E, Vara JM, Garzás J, Marcos E. 2010,. "Alignment of Open Source Tools with ISO norms for software product quality", Software Measurement European Forum (SMEF), Rom - Italy, pp. NA.

ISO 2001, *ISO/IEC Standard 9126 Software Product Evaluation–Quality Characteristics and Guidelines for their Use.* International Organization for Standarization.

ISO/IEC JTC 1 2011, *ISO/IEC FDIS 25010 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality model and guide.* International Organization for Standarization.

Lincke, R., Lundberg, J. & Löwe, W. 2008, "Comparing software metrics tools", *Proceedings of the 2008 international symposium on Software testing and analysis*, ACM, pp. 131.

Luijten, B., Visser, J. & Zaidman, A. 2010, "Faster defect resolution with higher technical quality of software", *Proc. of the 4th International Workshop on Software Quality and Maintainability (SQM'10), pp. NA*.

Mouchawrab, S., Briand, L.C. & Labiche, Y. 2005, "A measurement framework for object-oriented software testability", *Information and Software Technology,* vol. 47, no. 15, pp. 979-997.

Pagano, J. 2006. "Benefits and challenges of developing a public sector metrics program using commercial tools". In *Proceedings of the 1st international workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage information (CAMA '06). ACM, New York, NY, USA, 5-8.*

Plösch, R., Mayr, A., Pomberger, G. & Saft, M. 2009, "An Approach for a Method and a Tool Supporting the Evaluation of the Quality of Static Code Analysis Tools", *Proceedings of SQMB 2009 Workshop, held in conjunction with SE 2009 conference, March 3rd,* pp. NA.

Pressman, R. 2002, *Ingeniería del Software: un enfoque práctico,* McGraw-Hill, Madrid.

Riaz, M., Mendes, E. and Tempero, E. 2009, "A Systematic Review of Software Maintainability Prediction and Metrics", *3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009)*, pp. 367 - 377.

Saiedian, H. and Carr, N. 1997. "Characterizing a software process maturity model for small organizations". SIGICE Bull. Vol. 23, no. 1, pp. 2-11.

Samoladas, I., Gousios, G., Spinellis, D. & Stamelos, I. 2008, "The SQO-OSS quality model: Measurement based open source software evaluation", Open Source Development, Communities and Quality, Vol. 275, pp. 237-248.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P. & Murphy, R. 2007, "An exploratory study of why organizations do not adopt CMMI", *Journal of Systems and Software,* vol. 80, no. 6, pp. 883-895.

Wangenheim, C. G. v. 2009, "Enhancing Open Source Software in Alignment with *CMMI-DEV". IEEE Softw.* Vol. 26, no 2, pp. 59-67.

Washizaki, H., Yamamoto, H. & Fukazawa, Y. 2003, "A metrics suite for measuring reusability of software components", *Software Metrics Symposium, 2003. Proceedings. Ninth International*, ed. IEEE Computer Society, IEEE, Sydney, Australia, pp. 211.