Assessing the Impact of Measurement Tools on Software Mantainability Evaluation

Lerina Aversano and Maria Tortorella
Department of Engineering, University of Sannio, Benevento, Italy

Keywords: Software Metrics, Software Quality, Measurement, Evaluation.

Abstract: A relevant aspect of development and maintenance tasks is the evaluation of the software system quality. Measurement tools facilitate the measurement of software metrics and application of the quality models. However, differences and commonalities exist among the evaluation results obtained by the adoption of different measurement tools. This does not allow an objective and unambiguous evaluation of a software product quality level. In this direction, this paper proposes a preliminary investigation on the impact of measurement tools on the evaluation of the software maintainability metrics. Specifically, metrics values have been computed by using different software analysis tools for three software systems of different size. Measurements show that the considered measurement tools provide different values of metrics evaluated for the same software system.

1 INTRODUCTION

The software quality concept has evolved over time, and includes many important requirements for a correct product implementation, use and maintenance. Therefore, the good quality of a software products is an important requirement for adopting and/or maintaining it. In this direction, the availability of a quality models supporting software engineer during their evaluation activities is very important, in particular if they permit an objective evaluation of the quality level of a software product. In this context, the software analysis tools make easier the difficult task of software metrics evaluation and the quality models application. For this purpose, many software measurement tools have been developed. They have different characteristics with reference to the programming languages they analyze and software metrics they evaluate, and the evaluator is often confused to identify the tool that better addresses his/her needs. In addition, an objective evaluation of some quality attributes is difficult to obtain, even when a automatic measurement tool is adopted. Indeed, differences and commonalities exist among the evaluation results obtained by the adoption of different measurement tools. In this direction, this paper proposes a preliminary investigation of the impact of a set of measurement tools on the assessment of software maintainability analysing diversity existing among the different tools. Specifically, different software measurement tools have been analysed with the aim of understanding if their use brings to an equivalent evaluation of the maintenability characteristics. In particular, the analysis involved the assessment of three different size software systems. Overall, the results shows that the considered measurement tools provide different values of the metrics evaluated for the same software system, bringing different maintenability evaluations on the basis.

Next section of the paper describes some related works. Section 3 illustrates the plan of the study executed. Section 4 discusses the evaluation results, and final considerations are given in the last section.

2 RELATED WORKS

In literature many software metrics are defined for assessing the quality of software systems. Metrics can be used for addressing different software management tasks, such as software quality assessment, software process improvement, and so on. They can be measured by analyzing software artefacts, such as source code.

Examples of the most popular metrics are; number of lines of code (LOC), that is the simplest

Aversano, L. and Tortorella, M.
Assessing the Impact of Measurement Tools on Software Mantainability Evaluation.
DOI: 10.5220/0006793003920397
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 392-397
ISBN: 978-989-758-300-1

source code metric; the cyclomatic complexity, representing an internal complexity of software modules, and it is a good indicator to assume for identifying the presence of buggy modules; those ones of the CK metrics suite (Chindamber and Kemerer, 1994), which indicate features of object-oriented systems.

However, many widely used software metrics have not a complete and/or unique definition. For example, metric WMC (Weighted Methods for Class), which is part of the CK metrics suite, represents the weighted sum of the class methods, but its definition does not suggest how methods are weighted. The incompleteness of the definitions of metrics has involuntarily impact on their evaluation.

Several research papers analyze the behaviour of software metrics measurement tools. An analysis of tools evaluating OO metrics has been discussed in (Lincke, 2007), (Lincke et al., 2008). The authors analysed the tools supporting the CK metrics and concluded that the analysed measurement tools output different values for the same metrics. This was due to the difference in interpretation of the metric. A similar study has been proposed in (Codesido, 2011), where the authors observed that the metrics supported by the tools complement each other. In (Rutar et al., 2004) five tools making static analysis of Java source code have been compared, concluding that the usability of the results is difficult. A further comparison has been presented in (Bakar and Boughton, 2012) where the authors again observed that different tools provide different metric values for the same software system. In addition, the values obtained with the manual calculation were different from those obtained through the use of tools, as well. Moreover, in (Tomas, 2013), an analysis of open source tools analysing the Java and evaluate the supported metrics is discussed, but the authors do not provide any empirical validation.

The aim of the comparative study proposed in this paper is to further investigate the behaviour of a set of selected software measurement tools and related features, with the aim of understanding if the evaluation they perform regarding the metrics interpret and evaluate the same metrics and by applying the same strategy. Differently from the previous papers, the presented study focuses on a wider set of software metric tools.

3 PLAN OF THE STUDY

The execution of the presented study, required a planning of the activities to be executed. The main steps are the following:

- **Scope Definition**. The aim is to investigate the impact of the software measurement tools on the evaluation of software quality metrics, with the goal of verifying if they induce to different maintenability evaluations. The task required the selection of the software measurement tools to analyse. Tools have been compared on the basis of the metrics they consider and the measurement they perform. In particular, the paper investigates on the following question: Do the software measurement tools impact of the software maintainability assessment? If yes, what is the kind of impact they have?
- **Metrics Selection**. The aim of this step is to select a comprehensive set of metrics useful for the evaluation of the software maintenability characteristics. Its execution has required the analysis both standards and quality models, and selected tools. The selected metrics have been analyzed with reference of the chosen tools for understanding their impact on the maintainability measurement.
- **Selection of the Software Systems.** The step aimed at choosing the set of software systems to be analyzed for assessing their maintenability by using the selected measurement tools. Open source software systems have been considered. Their selection had to take into account the license kind, as many tools are just partially open source and their source code is not always available. In addition, just Java software systems were considered.
- **Metric Evaluation**. This steps has entailed the measurement of the chosen metrics by assessing the considered software analysis tools for evaluating the selected software systems.
- **Analysis of the Results**. This step has compared the values of the software maintenability characteristics evaluated on a software system by using the different software measurement tools. The aim was to verify how similarly the evaluation tools evaluate the metrics concerning a characteristics and apply the same rules for evaluating the same metric.

The following subsections describe with a greater details the process applied for performing the selection of the considered software measurement tools and the selected metrics.

3.1 Metric Selection

The study of standards and evaluation models helped in the identification of metrics, features and sub-features useful for evaluating by the software maintenability by using the considered software analysis tools.

The considered metrics have been selected by taking into account which metrics the chosen software measurement tools could evaluate. In particular, the considered metrics can be grouped as it follows:

- Dimensional Metrics: LOC (Lines of Code), TLOC (Total Lines of Code), NOP (Number of Packages), NOM (Number Of Methods), MLOC (Medium LOC per method), NOA (Number Of Attributes), etc.
- Object Oriented Metrics used are the object oriented metrics proposed by Chidamber and Kermerer in 1994 (Chidamber and Kemerer, 1994), called CK Metrics, are considered. Some examples are: WMC (Weighted Methods for Class), CBO (Coupling between Objects), RFC (Response For Class), LCOM (Lack of Cohesion of Methods), DIT (Depth of Inheritance Tree), NOC (Number of Children) (Henderson-Sellers, 1996).

Table 1: Maintainability metrics.

Analyzability	LOC	Stability	LOC
	Class& Interface		D
	CC		
	NOA		
Changeability	NOC	Testability	NOC
	LCOM		LOC
	CC		DIT

3.2 Measurement Tools

The software analysis tools to be considered were chosen among the most used open source systems used for measuring software metrics. Open source and freeware analysis tools were considered for permitting their adoption without spending limits

In addition, the tools were chosen also on the basis of the programming language they could analyse and evaluate. In particular, as the results of the measurements to be performed have to be compared, all the chosen tools need to analyze the software systems written by using the same programming languages.

The considered software systems perform a scan of the code and identify eventual errors in the code in an automatic way. They also allow the analysis of the code and automatic evaluation of a large number of metrics. The search of a suitable set of software tools was executed by making a free search on the internet. More than forty software analysis tools were identified in the site SourceForge.net. In order to compare them, only the tools analysing Java software code were taken in consideration. Their recorded characteristics were: Name, home page link, license type, availability, supported programming languages, operating supported system/environment and evaluated metrics. In the end of this preliminary analysis, nine software analysis tools were selected and they are:

- Eclipse Metrics Plugin 1.3.6 A metrics calculation and dependency analyzer Eclipse plugin for (http://easyeclipse.org/site-1.0.2/plugins/metrics.html)
- CCCC A command-line tool. It analyzes C++ and Java files and generates reports on various metrics. (http://cccc.sourceforge.net/)
- Understand A reverse engineering, code exploration and evaluation metrics tool for different programming languages. It provides a collection of metrics. (https://scitools.com/)
- JArchitect A static analysis tool for Java evaluating numerous code metrics, and allowing for some metric visualization. (http://www.jarchitect.com/)
- Stan4j An Eclipse plug-in that allows for analysis of the dependencies between classes and packages, and evaluates code metrics. (http://stan4j.com/)
- CodePro Analytix An Eclipse plug-in, offered by Google and regarding software quality improvement and reduction of development costs. It provides support for code analysis, dependency analysis and metric measurement. (https://marketplace.eclipse.org/content/codepro-analytix)
- LocMetrics A freeware simple tool, used to measure the size of a software program by counting the number of lines in the source code. (http://www.locmetrics.com/)
- SourceMonitor a tool for code exploration, including the measurement of a set of metrics related to the identification of complexity. (http://www.campwoodsw.com/sourcemonitor)
- CodeAnalyzer A Java application for C, C++, Java, Assembly, Html. It calculates metrics across multiple source trees as one project. (http://www.codeanalyzer.teel.ws/)

4 EVALUATION

The metric values evaluated using the measurement tools have been used to analyze the Maintenance feature. For such analysis, SimMetrics 1.0 software has initially been considered. In particular, as it can be seen from Figures 1 and 2 and Table 2, LocMetrics and CodeAnalyzer allow the measure of a lower number of metrics respect the others.

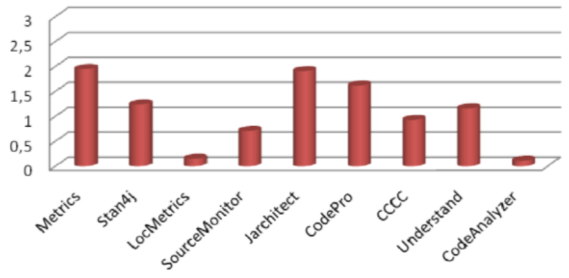

Figure 1: Maintainability of SimMetrics 1.0.

On the other hand there are tools that provide Maintainability values of not too distant, such as Metrics, JArchitect, Stan4j and CodePro. The first pair gives a value around 2, and the other two tools instead a value around 1.7. As it can be seen from Figure 2, the values of Metrics and JArchitect are quite similar.

To further investigate the differences in the metric values of the various tools of measurement, the analysis has been focused only on the tools that provide four or more metrics together, that are Metrics, Stan4j, JArchitect, CodePro Analityx and Understand. The results of the for SimMetrics 1.0 are in Table 2.

As Figure 1 indicate, Metrics and Stan4j bring to a Maintainability value near to 2.0, JArchitect brings to a value of 2.77 while Understand and CodePro Analytix provide a value near to 1.60.

In particular, JArchitect supplies the highest maintainability index because it has a Stability Index higher than others, this is due to a value of the LOC metric much lower than others, which positively influences the evaluation of the characteristic being considered.

Indeed, as already mentioned the LOC metric obtained with Metrics is equal to 2038 while for JArchitect it is 1191, therefore this conduct to a different evaluations of the Analyzability and of the Maintainability.

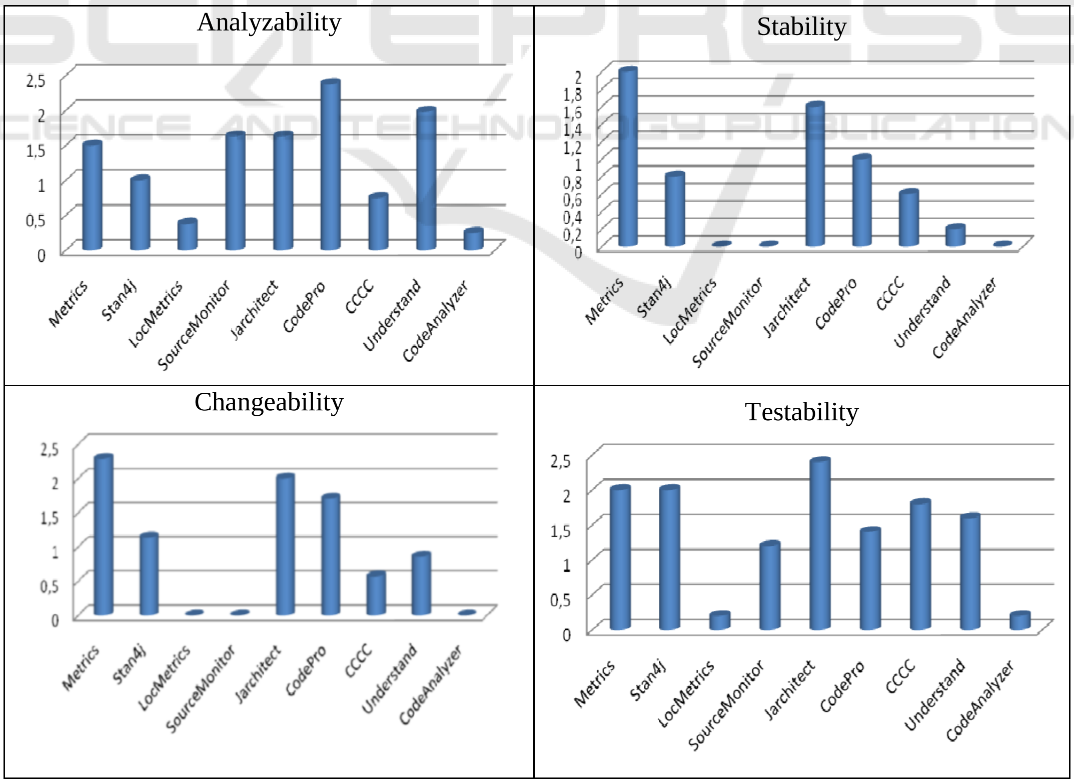

Figure 2: Overall maintainability metrics for SimMetrics 1.0.

Table 2: metric values of the maintainability metrics.

Tools/ Metrics	Metrics	Stan4j	Jarchitect	CodePro	Understand
Analyzability					
LOC	2238	2467	1191	2238	2237
Class& Interface	47	47	47	47	47
CC	1,63	1,36	1,86	1,55	1,53
NOA	2,77	3,64	3,69	2,5	n/a
Changeability					
NOC	0,61	0,53	0,55	0,74	0,61
LCOM	0,28	5,49	0,28	n/a	46,46
CC	1,63	1,36	1,86	1,55	1,53
Stability					
D	0,38	0,42	0,14	0,19	n/a
LCOM	0,28	5,49	0,28	n/a	46,46
Testability					
NOC	6,05	6,79	6,72	5,23	6,39
LOC	2238	2467	1191	2238	2237
NOC	0,61	0,53	0,55	0,74	0,61
DIT	1,71	1,47	1,49	2,48	1,68

For Tool Metrics and Stan4j, it can be observed that they assume a quite similar to Maintainability value. This is due to the fact that Metrics has assume Analyzability and Stability values higher than Stan4j, but it reports a lower Changeability value respect to Stan4j.
In case, instead of the CodePro and Understand tools, they assume Maintainability value quite similar because they have a almost equal values for the metrics. The difference is due to the two metrics that are not evaluated by Understand, D and NOA, and by CodePro which is LCOM. Conversely, Understand provides a very high value in the case of LCOM metric that weighs unmatched metrics.

A deeper evaluation has been performed on 20 software systems to understand if emerged differences related to SimMetrics.
Observing the indexes it is possible to observe that the Maintainability obtained with Metrics and Stan4j, report the same differences emerged with SimMetrics assessment.
Only in the case of JGraph software evaluation the values of the two tools produce an almost similar index (Metrics 2.08, Stan4j 2.06).
Instead, comparing the results obtained using Metrics and JArchitect tools in can be observed in the first graph of Figure 3 that the evaluated software do not always have different values, but going to observe the second graph of Figure 3 it may be noticed that some software has almost the same indices as iReport, EasyMock and Judo software with a value of 2.21.

5 CONCLUSIONS

Have been discussed and reasoned about for years, but only few metrics have even been experimentally validated.
Nowadays, software engineering managers always more often needs to deal with quantitative data regarding the quality of a software system.
Indeed, a number of metrics are generally adopted and measured during maintenance and evolution processes to predict effort for maintenance activities and identify parts of the software system needing attention.

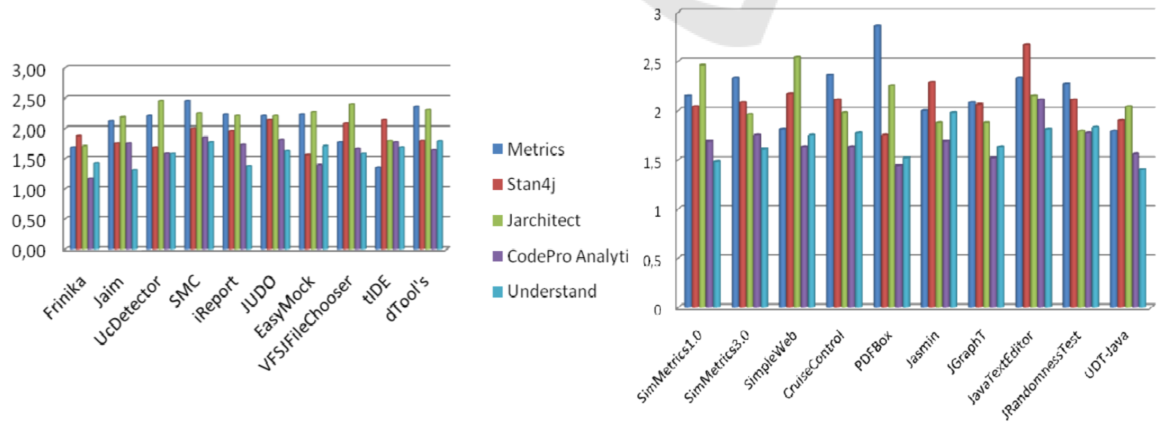

Figure 3: Maintainability of 20 assessed software systems.

However, a lot of metrics have been discussed and reasoned about for years, but only few metrics have been experimentally validated.

Numerous software metrics tools exist that are used to evaluate the software metrics, however, in order to use them in practice, it would be necessary their validation for knowing how they behave and their evaluation have to be interpreted. The evaluation presented in this paper showed that differences exist among the software metrics tools, at least among those ones that have been investigated.

The evaluation highlighted that the tools delivered similar results just for certain metrics. In the large part of the cases, each tool provides a different value for each common metric, and this difference is more evident with the increasing of the size of the analysed software system. This depends on the fact that each tool interprets differently the

metrics, calculates them by applying different rules, and very often do not implement the evaluation by applying the intended definition. the work also analyzed how the existing differences in the values of the metrics evaluated with different tools influences the evaluation of higher level characteristics, such as the maintainability. In fact, the obtained results have highlighted the very different maintenability values obtained by applying the different measurement tools.

A better definition the evaluation process will be formalized in the future works, which will also aim at performing a more extensive evaluation by applying the assessment process to higher number of case studies.

REFERENCES

S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

B. Henderson-Sellers. *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

W. Li and S. Henry. Maintenance Metrics for the Object Oriented Paradigm. *In IEEE Proc. of the 1st Int. Sw. Metrics Symposium*, pages 52–60, May 1993.

R. Lincke. *Validation of a Standard- and Metric-Based Software Quality Model – Creating the Prerequisites for Experimentation*. Licentiate thesis, MSI, V ̈axj ̈o, University, Sweden, Apr 2007.

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

N. Rutar, C. B. Almazan, J. S. Foster, A comparison of bug finding tools for Java, *In proceedings of the IEEE 15th International Symposium o Software Engineering*, ISSRE, 2004.

I. Lamas Codesido, Comparación de analizadores estáticos para código java, 2011.

N. S. Bakar, C. V. Boughton, Validation of measurement tools to extract metrics from open source projects, *IEEE Conference on Open Systems (ICOS),* IEEE, 2012.

E. H. Alikacem, H. Sahraoui, Generic metric extraction framework, *Proceedings of the 16th International Workshop on Software Measurement and Metrik Kongress (IWSM/MetriKon*). 2006.

P. Tomas, M. J. Escalona, M. Mejias, Open source tools for measuring the Internal Quality of Java software products. A survey, Computer Standards & Interfaces 36(1): 244-255, 2013.