Journal of Computer Sciences

Original Research Paper

Comparative Study of the Quality Assessment Tools Based on a Model: Sonar, Squale, EvalMetrics

Zineb Bougroun, Adil Zeearaoui and Toumi Bouchentouf

Laboratoire des Systmes Electronique, Informatique et Images LSE2I, National School of Applied Sciences, Oujda, Morocco

Article history
Received: 02-10-2015
Revised: 20-11-2015
Accepted: 05-03-2016

Corresponding Author:
Zineb Bougroun
Laboratoire des Systmes Electronique, Informatique et Images LSE2I, National School of Applied Sciences, Oujda, Morocco
Email:bougroun.zineb@gmail.com

Abstract: To build software, the customer is looking for a company that develops the product in record time with minimal cost and good quality. To measure the productivity and the software quality, several indicators and metrics have emerged, which have been the subject of various research fields and which are highly demanded by enterprises and software development teams. In order to assess the software quality and ensure of the quality of the product many tools have been developed and used. The work presented in this study is focused on tools measuring software quality, so we present the open source tools developed in java, then we compare it, according to some criteria defined in this study.

Keywords: Software Product, Quality, Model, Metrics, Sonar, Squale, EvalMetrics

Introduction

Quality takes more interest in the software development life. This is the reason why there are different aspects that seek to improve software quality, among these, there is the aspect of processes and methodologies aimed at organize work to reduce time of product and improve quality, that we discussed in previous articles (Bougroun *et al*., 2014; 2015) and the aspect of product quality using metrics (Bougroun *et al*., 2013; 2012a; 2012b) to measure the capability of the product to meet certain factors, in this study we continue to study the second axis, focusing on the evaluation tools software.

Metrics are a way to measure, monitor and predict the quality of a software product. The customers become more and more interested to know the quality measurement of their development during their building; than a tool that shows the progressively developing the available quality is mandatory.

In this area and to predict the software quality, a lot of tools are appeared that collects, calculates and presents the results of the metrics. Among these tools there are those who only collect the metrics and displays their results, there are also those that analysis the result of metrics by setting a minimum and maximum threshold to judge quality and there are also tools that relies on quality models analyzing each factor and criterion then analyze product quality.

In this study, we will make a comparative study between the open source assessments tools developed with java, based on quality models collecting and calculating metric and presenting their analyses to evaluate java software.

This work will be presented as follow:

In the first section we present the related works, we explain the context and the method used to verify selected tools, in the next section we present tools and study their model and plugins. After that we present the comparison between those tools and finish the paper by discussion and conclusion.

Related Work

Many studies have treated the software assessment tools among which we quote:

The first study that was mentioned there is the work done by Thomas *et al*. (2013); they made a study of the state of art of open source tools developed under java and they compared them using the following criteria: Internal quality models supported; metrics implemented; the year of the first version and the latest version and features functional covered.

The study of Rutar *et al*. (2004) compare five bug detection tools (Bandera, ESC/Java 2, FindBugs, JLint and PMD) using static analysis of Java code. The result of this study is that although there are some overlaps among the types of errors detected, most of

them are different. They also say that the use of these tools is very difficult due to the number of results they generate.

Lamas compares two tools (Codesido, 2011), FindBugs and PMD that are complementary in terms of bugs detected despite the fact that they are some overlaps among them.

Ayewah *et al*. (2007) discuss the warnings found by FindBugs tool and classify them by kinds, positives (warnings that aren't really defects), trivial bugs (true defects with minimal impact) and serious bugs (defects with significant impact).

Van Emden and Moonen (2002) present an approach for the automatic detection and visualization of code smells with jCOSMO and discuss how this approach can be used in the design of a software inspection tool.

In the articles above, we find research studying the different tools and show their weakness and strength we find also a study of one tool at a unique point of view. In this article we will discuss the tools that support a quality model (thus the same strategy) but we'll compare them according to the nature of the metric, the code smells and the presentation of the result.

Context and Method

As we have stated in the previous paragraph this article is allocated to study open source tools developed under java that supports a quality model and aimed to analyze and evaluate the quality of software. This study is based on the following criteria:

- Nature of the metrics implemented in the tool
- Code smells detected
- The results presentation method

To select tools of this study we have use the following source of information:

- Google Scholar and science direct were used to find the related work according to the following keywords: Open source tools java, quality metrics, code smells, design smells. Among the most relevant articles that based on our survey they are four articles (Tomas *et al*., 2013; van Emden and Moonen, 2002; Spinellis *et al*., 2009; Wagner *et al*., 2005)
- Java Power Tools book (Smart, 2009). Chapter "Quality metrics tools"
- ISO/IEC 25000 portal (ISO/IEC, 2015). Section "Open Source Measurement Tools"

As a result we found among sixteen tools only three (Sonar, Sonar Plugins and Squale) that implements a quality model (ISO 9126 SIG and SQUALE).

Tools Characteristics

Sonar

Presentation

Sonar (http://www.sonarsource.com/) is an open platform to manage code quality in a continuous way developed and supported by Sonar Source. It aims to analyze the quality of components and report them with a web server, it stores metrics in a database and presents them. Each new release of component triggers a complete analysis. The developer can also trigger an analysis during the development phase to anticipate the quality and correct it before the Release. Sonar follows the ISO/IEC 9126 to assess the quality of the projects under evaluation and provides as core functionality code analyzers, defects hunting tools, reporting tools and a time machine (Veiga and Frade, 2010). Sonar is a very recent tool (it appeared in 2009), but it has already more than forty plugins available. However only four plugins have the ability to view report of results (Fig. 1).

Model

The quality model in ISO/IEC 9126 was developed during 2001 to 2004, it comprises two sub-models: The internal and external quality model and the quality in use model. The quality model was inspired from McCall's and Boehm's models. The model is divided in 6 characteristics: Functionality, reliability, usability, efficiency, maintainability and portability; which are further subdivided into 27 sub-characteristics (ISO/IEC, 2001; 2003a; 2003b; 2004).

Plugins

Sonar groups a set of well-known code analyzers such as Cobertura, PMD, FindBugs, Squid, CheckStyle and Clover (Arapidis, 2012).

By using these plugins, it is able to cover all categories: Comment size (Density of comment lines and some other related), duplicated code (Density of duplicated lines), complexity (Average complexity by method, Average complexity by class, Average complexity by file …), coding rules (Violations of Sun code conventions), dependencies (Package cycles, Package dependencies, File dependencies …), Unit tests (is refer the number of successful or failed tests, it also takes into account parts of the code not covered by the tests) (Fig. 2), Potential bugs (this criterion refers to the various security vulnerabilities or bugs that may be present in source).

Squale

Presentation

Squale is a web application which asses projects by presenting the result of metrics, it use a batch process

developed in Java that performs the analysis of source code (Squalix) by means of a database that stores the metrics (Fig. 3 present the interface of projects analysis list).

Model

SQUALE model has been developed and validated over 2008-2009 in an industrial setting with Air France-KLM and PSA Peugeot-Citroen. It use the ISO 9126 model, which promotes a three-level model of quality (factors, criteria and metrics) and add practices as an intermediate level between metrics and criteria (Mordal-Manet *et al*., 2009). In terms of analysis and presentation of data, it shows 3 out of 6 factors of SQUALE Quality Model: Maintainability, evolutivity and reuse capacity, discarding analysis, functionality, architecture and reliability (Tomas *et al*., 2013).

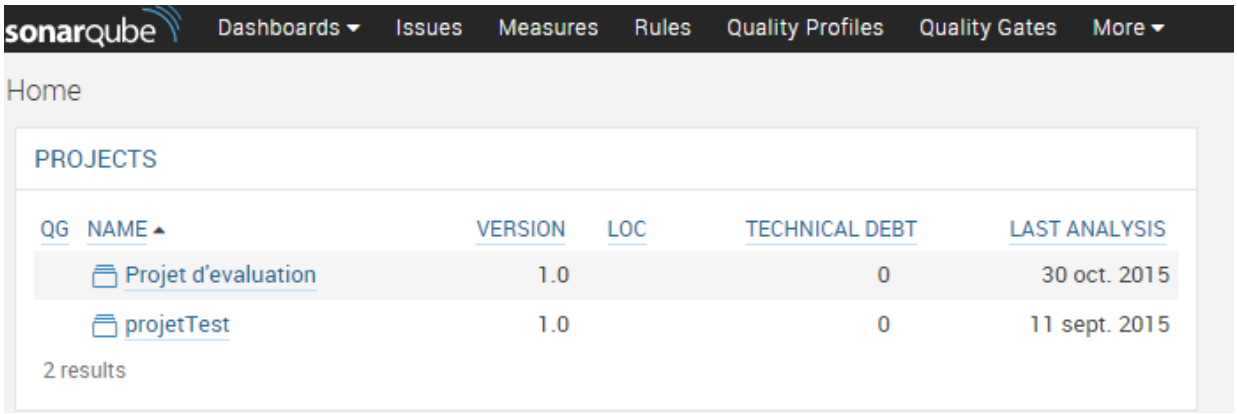

Fig. 1. Sonar interface for list evaluation project

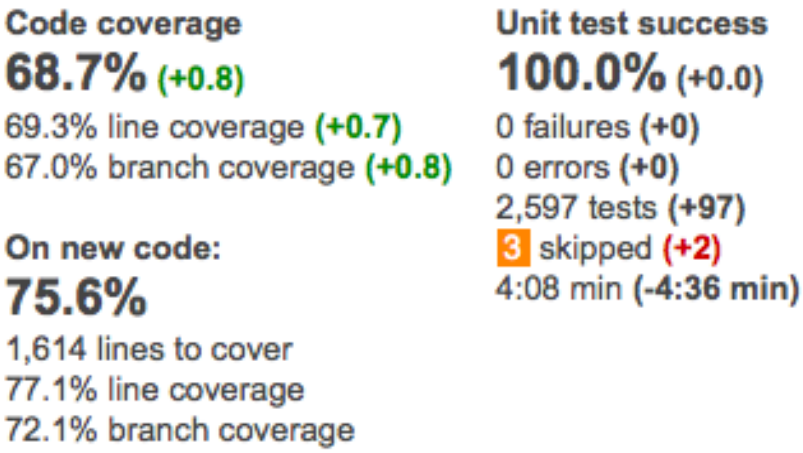

Fig. 2. Sonar result of test evaluation

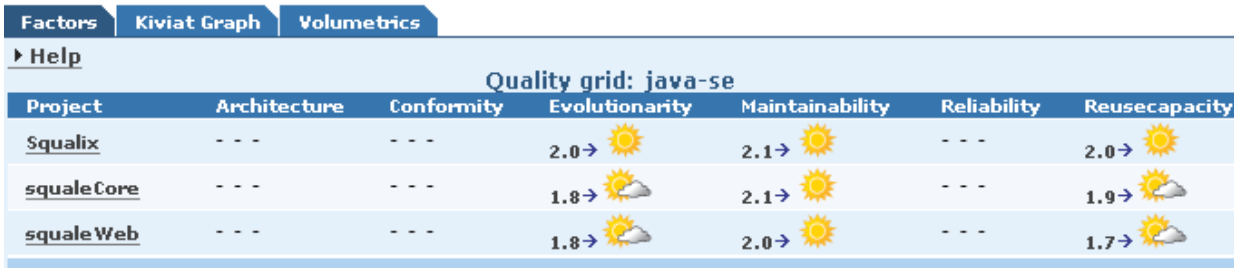

Fig. 3. Squale interface for list evaluation project

Plugins

Squalix invokes the following plugins for static analysis: Checkstyle, JavaNCSS, CKJM, PMD/CPD and Jdepend (Tomas *et al*., 2013). By using these plugins it covers: Code size (number of lines by method, by class…), comment size (Density of comment lines), duplicated code (number of duplicated lines) complexity (Average complexity by method, average complexity by class, average complexity by file …), dependencies (class dependencies packages dependencies…), coding rules (Violations of Sun code conventions) and code smells.

Eval Metric

Presentation

EvalMetric is an open source tool developed by the team of ENSAO to assess software during its development (https://sourceforge.net/projects/evalmetrics/). It is a web application that uses metrics to judge a project and analyze its quality. The developer can also trigger an analysis during the development phase to allow them to anticipate the quality and correct it before the Release. Once we have a new release the customer can trigger an analysis that is saved in the database and in this way we will have the entire history of the application. EvalMetric also offers the possibility of extracting report and graphs to facilitate analysis. This tool is based on the standard ISO 25000 (ISO/IEC, 2015) (the Fig. 4 present the interface of a list of evaluated projects).

Model

EvalMetric is based on the standard ISO 25000 which is an evolution of ISO 9126; it was developed between 2012 and 2014. The quality model is divided into two parts, quality in use and product quality, this tool use the second one which is structured to three levels the first one contains eight characteristics, the second level contains thirty one sub-characteristics and the last one concerns the measures. The EvalMetrics has added to the model another level which contains properties of quality and design and insert it between the metrics and the sub-characteristics level.

Plugins

The tool does not use a plugins to calculate metrics the entire product has been developed internally. The tool covers: Code size (number of lines by method, by class…), comment size (Density of comment lines), duplicated code (number of duplicated lines) complexity (Average complexity by method, Average complexity by class, Average complexity by file …), dependencies (class dependencies packages dependencies…), coding rules (Violations of Sun code conventions) and detection of design patterns.

Comparison between Sonar, Squale and EvalMetric

Benchmarks

In this study we try to make a comparison between open source java software evaluating software developed in Java. As we have seen in the previous parts this study will be reduced to three software views the criterion that was given from the beginning (tools based on a model): Sonar Squale and EvalMetric.

The study will be of interest to meet the following questions:

- What are the metrics implemented and in which category it belongs?
- What are the codes smells that can be detected?
- What are the rules of coding that can be detected?
- How much the software can give us an overall vision quality?

Q1: What are the Metrics Implemented and in which Category it Belongs?

In this issue we will classify implemented metrics tools according to its categories to have a general vision on the level of completeness of this tool. **Here are metrics categories taken in consideration**:

Complexity; inheritance; code size; coupling; Encapsulation ; Cohesion; package architecture; package dependencies; package cohesion; package size and test.

In the study presented in the Table 1, Squale covers the majority metric categories, regarding the code, except encapsulation, Squale shows low on tests, using this tool you cannot know the tests done and the untested code (Fig. 5 to know how squale presents the results metrics). Sonar is more oriented to test cases done/not done condition covered/ not covered ... (Fig. 2) in terms of code metrics it focuses on the size criterion: The comment size, file size, the package size class ... and some metric dependencies: Dependencies between files, packages (Fig. 8 shows results of metrics). EvalMetric also covers all categories previously mentioned except the test category. Between EvalMetric and Squale, the last one implements several metrics in each category while EvalMetric has only a few metrics in each category (Fig. 6 shows the result of size metrics and Fig. 7 shows the result of characteristics indication).

Q2: What are the Codes Smells that can be Detected?

The three tools use the PMD plugins which is a powerful tool to detect smells code. This analysis tool scans Java source code and looks for potential problems like possible bugs, dead code, suboptimal

code and overcomplicated expressions, for instance [7]. It is based on sets of validation rules or rulesets. Each ruleset comprises a set of rules and every rule corresponds to a code checking. The rules of PMD look for bad coding practices to avoid potential errors resulting from experience. PMD includes a module known as CPD "Copy Paste Detector", which can detect the duplicated code existing in the program and therefore measure the number of blocks, lines and duplicated tokens.

Fig. 4. Squale interface for list evaluation project

Metric	Value
Max value of the cyclomatic complexity	1.0
Sum of all cyclomatic complexity	1.0
Number of methods	1.0
Number of javadoc	2.0
Number of classes	0.0
Number of source lines	10.0
Lack of cohesion in methods	1.0
Afferent coupling	5.0
Weighted methods per class	2.0
Number of Children	0.0

Fig. 5. Squale interface for results metrics

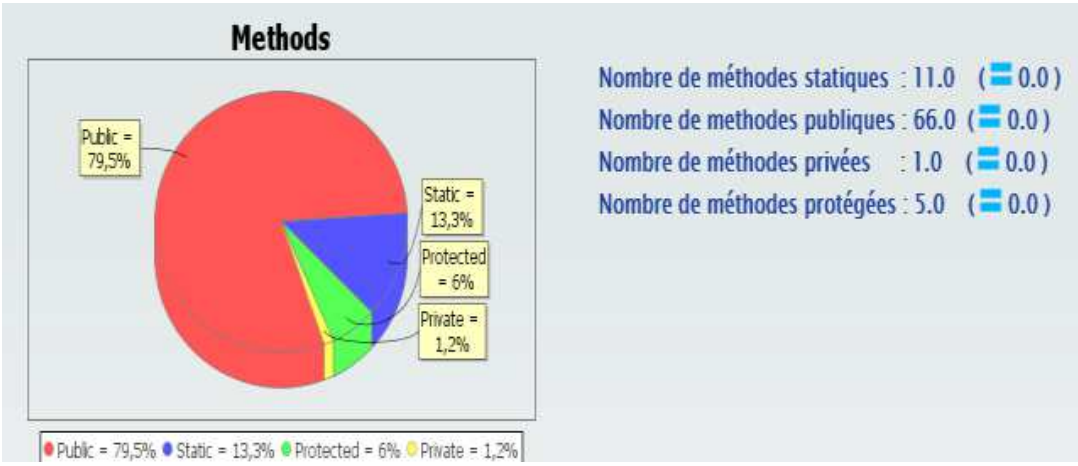

Fig. 6. EvalMetrics interface for methods metrics

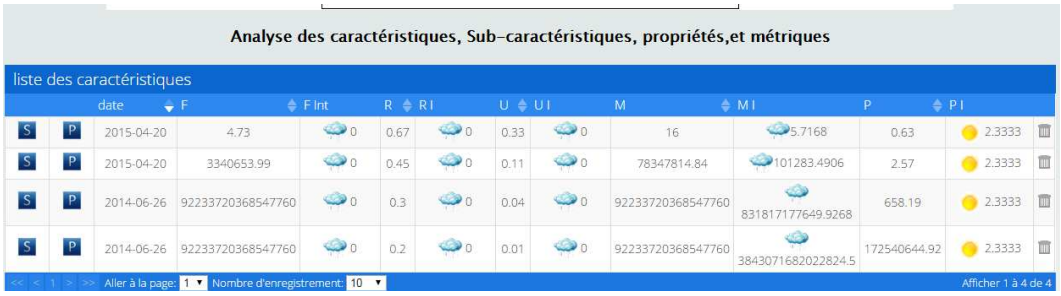

Fig. 7. EvalMetrics interface for characteristics indication

Fig. 8. Sonar interface for results

Table 1. Comparative study of Squale Sonar and EvalMetric by class of theirs metrics

Categories of metrics											
	Inheritance	Code size	Coupling	Encapsulation	Cohesion	Pack architecture	Pack dependencies	Pack cohesion	Pack size	Test	
Squale	WMC V(G) eV(G)	DIT NHL NOC NIM NRM SIX	SLOC LOC NOM CLOC	CBO COF RFC	-------	LCOM1 LCOM2 LCOM3 LCOM4 TCC	D I A Ca Ce Out-port Classes In-port Classes Hidden Classes	SPC SCC PP CP PRIS SRIP NPCD NPD	CPC Locality Happiness	NCP	-------
Sonar	WMC V(G)	-------	NOM NOC NOP LOC	-------	-------	-------	-------	Package cycles Package dependencies to cut Package tangle index NPD	-------	-------	Condition coverage. Conditions by line. Covered conditions by line. Line coverage. Lines to cover. Skipped unit tests. Uncovered conditions. Uncovered lines. Uncovered lines. Unit tests. Unit tests duration. Unit test errors. Unit test failures. Unit test success density.
Eval Metrics	WMC V(G)	DIT NOC NIM	NOM NOC LOC CLOC	RFC CBO	AHF APF MHF MPC MPF	TCC	D I A Ca Ce	PP CP NPD	CPC	NPC	-------

Q3: What are the Rules of Encodings that can be Detected?

The three tools are based on the Checkstyle tool that can detect up to 2228 issues that are java code conventions and standards, for example: Missing javadoc, related to code beautification, declaring field as final…Its operation is based on validation rules, which are equivalent in most cases to coding conventions, so rule violations allow measuring coding conventions violations. Even though this is its main functionality, since version 3 it can identify class design problems, duplicated code, or bug patterns (Tomas *et al*., 2013). There is no difference between the three tools because the tools are based on the same tool. =

Q4: How Much the Software can Give us an Overall Vision Quality?

Sonar is a very useful tool if you want to make a microscopic analysis on a project using a quality model. This means that although it is based on ISO 9126 that this model does not appear in the project analysis. The tool focuses primarily on metrics and coding rules; it presents them in general (example: The percentage commentary throughout the project) and presents them in a specific way (the user comment for any class, method ...).

With EvalMetric and Squale tools we can do a macroscopic and microscopic analysis. EvalMetrics and Squale are based on the ISO 25000 and SQUALE model therefore project analysis comes in hierarchy by following the model used, so you can view the quality factors of your project model and see their satisfaction which can make you to decide and evaluate the quality of the whole project. These tools give you also the possibility to analysis each package, class method by showing the metrics related to each level.

Discussion

The study was done in this article focuses on open source tools developed with java, based on a quality model, that evaluates java software. This study was reduced to three tools Sonar Squale and EvalMetrics. it was done on functional criteria and according to these criteria presented in the previous section we can say that sonar is richer than the other tools in regard to the presentation of tests done/not done ... but it is lower in the implementation of the metric, for it does not cover all quality properties and it does not put value in the model on which it is based (no presentation of the model in its Results).

Squale and EvalMetrics are similar in regard to functionality that they cover, it is true that Squale does not cover the encapsulation property but it implements many metrics in the other properties contrary to EvalMetrics. The main advantages of EvalMetrics compared to this tools is that it has a part to detect designs patterns as well as anti patterns and it traces the history of each quality factor each metric for your project from your first test (Fig. 9 and 10).

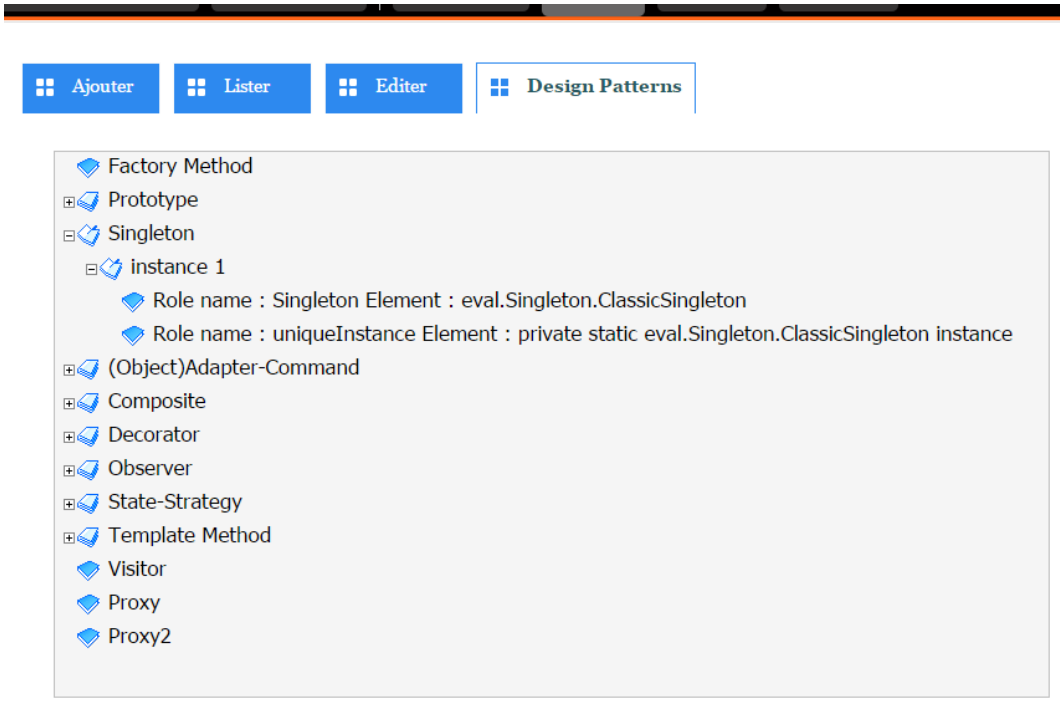

Fig. 9. EvalMetrics interface for design pattern result

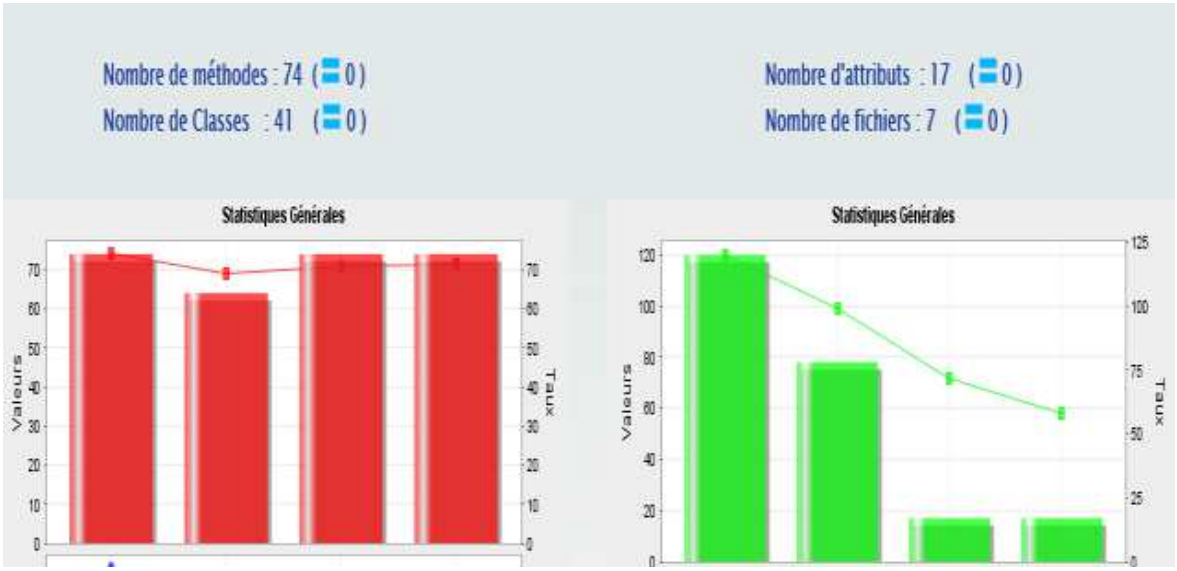

Fig. 10. EvalMetrics interface for metric history

Conclusion

The comparative study we did in this study has focused on open source Java tools based on model that evaluate java software, this work was based on the survey done on open source tools for measuring of the quality. View the first criterion demanded (software which is based on a quality model) our study was limited in three software (Squale, Sonar, EvalMetrics) and as a conclusion of this study Sonar shows a force in the presentation of the unit test (Unit test errors, test time, test done, condition covred ...) while the other two tools did not. Squale and EvalMetrics exceed Sonar in presenting the model that it implements and between the two tools EvalMetric exceeds Squale view that it shows the patterns and the anti patterns of the project.

Funding Information

The authors have no support or funding to report.

Author's Contributions

Zineb Bougroun: Participet in all experiments, cordenates and the writing of the work.

Adil Zeearaoui: Cordinated the work.

Toumi Bouchentouf: Orgnized the work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Arapidis, C., 2012. Sonar Code Quality Testing Essentials. 1st Edn., Packt Publishing, Birmingham, ISBN-10: 1849517878, pp: 318.

Ayewah, N., W. Pugh, J.D. Morgenthaler, J. Penix and Y.Q. Zhou, 2007. Evaluating static analysis defect warnings on production software. Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, Jun. 13-14, ACM, USA., pp: 1-8. DOI: 10.1145/1251535.1251536

Bougroun, Z., A. Zeaaraoui and T. Bouchentouf, 2014. The projection of the specific practices of the third level of CMMI model in agile methods: Scrum, XP and Kanban. Proceedings of the 3rd IEEE International Colloquium in Information Science and Technology, Oct. 20-22, IEEE Xplore Press, Tetouan, pp: 174-179. DOI: 10.1109/CIST.2014.7016614

Bougroun, Z., A. Zeaaraoui and T. Bouchentouf, 2015. Scrumban/ XP: A new approach to cover the third level of CMMI model. Proceedings of the Mediterranean Conference on Information and Communication Technologies, (ICT' 15).

Bougroun, Z., A. Zeaaraoui, M. Saber and T. Bouchentouf, 2013. Enhancement of the taxonomy of Metrics by ISO model using design quality properties. Proceedings of the International Syposium on Securiy and Safety of Complex Systems, (SCS' 13) Agadir Morocco.

Bougroun, Z., A. Zeaaraoui, M.G. Belkasmi and T. Bouchentouf, 2012a. Joining ISO model with metrics using design quality properties. J. Inform. Syst. Manage., 2: 184-195.

Bougroun, Z., A. Zeaaraoui, M.G. Belkasmi and T. Bouchentouf, 2012b. Classification of the metric in ISO model by oriented object properties. Proceedings of the 2nd International Conference on Innovative Computing Technology, Sept. 18-20, IEEE Xplore Press, Casablanca, pp: 128-132. DOI: 10.1109/INTECH.2012.6457816

Codesido, I.L., 2011. Comparación de analizadores estáticos para código Java.

http://www.sonarsource.com/

https://sourceforge.net/projects/evalmetrics/

ISO/IEC, 2001. ISO/IEC TR 9126-1: Software engineering-product quality-part 1: Quality model. International Organization for Standardization.

ISO/IEC, 2003a. ISO/IEC TR 9126-2: Software engineering-product quality-part 2: External metrics. International Organization for Standardization.

ISO/IEC, 2003b. ISO/IEC TR 9126-3: Software engineering-product quality-part 3: Internal metrics. International Organization for Standardization.

ISO/IEC, 2004. ISO/IEC TR 9126-4: Software engineering-product quality-part 4: Quality in use metrics. International Organization for Standardization.

ISO/IEC, 2015. ISO/IEC 25000 portal.

Mordal-Manet, K., F. Balmas, S. Denier and S. Ducasse, 2009. The squale model-a practice-based industrial quality model. Proceedings of the IEEE International Conference on Software Maintenance, Sept. 20-26, IEEE Xplore Press, Edmonton, AB., pp: 531-534. DOI: 10.1109/ICSM.2009.5306381

Rutar, N., C.B. Almazan and J.S. Foster, 2004. A comparison of bug finding tools for Java. Proceedings of the 15th International Symposium on Software Reliability Engineering, Nov. 2-5, IEEE Xplore Press, pp: 245-256. DOI: 10.1109/ISSRE.2004.1

Smart, J.F., 2009. Java power tools.

Spinellis, D., G. Gousios, V. Karakoidas and P. Louridas, 2009. Evaluating the quality of open source software. Electr. Notes Theoretical Comput. Sci., 233: 5-28. DOI: 10.1016/j.entcs.2009.02.058

Tomas, P., M.J. Escalona and M. Mejia, 2013. Open source tools for measuring the internal quality of Java software products. A survey. Comput. Standards Interfaces, 36: 244-255. DOI: 10.1016/j.csi.2013.08.006

van Emden, E. and L. Moonen, 2002. Java quality assurance by detecting code smells. Proceedings of the 9th Working Conference on Reverse Engineering, Nov. 29-Dec. 1, IEEE Xplore Press, pp: 97-106. DOI: 10.1109/WCRE.2002.1173068

Veiga, J.M. and M.J. Frade, 2010. TreeCycle: A Sonar plugin for design quality assessment of Java programs. Techn. Report CROSS-10.07-1.

Wagner, S., J. Jürjens, C. Koller and P. Trischberger, 2005. Comparing bug finding tools with reviews and tests. Proceedings of the 17th IFIP TC6/WG 6.1 International Conference on Testing of Communicating Systems, (TCS' 05), Springer, Montreal, Canada, pp: 40-55. DOI: 10.1007/11430230_4