

Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2023)

13ο Πανελλήνιο και Διεθνές Συνέδριο «Οι ΤΠΕ στην Εκπαίδευση»



Assessment of JAVA code quality: Examining the efficacy of an automated tool as a teacher's assistant

Dimitrios Margounakis, Dimitrios Sofronas, Maria Rigou, Theodore Pachidis

Assessment of JAVA code quality: Examining the efficacy of an automated tool as a teacher's assistant

Dimitrios Margounakis¹, Dimitrios Sofronas², Maria Rigou², Theodore Pachidis³
dmargoun@sw.duth.gr, std131050@ac.eap.gr, rigkou.maria@ac.eap.gr, pated@cs.ihu.gr

¹ Democritus University of Thrace, Komotini, Greece

² Hellenic Open University, Patras, Greece

³ International Hellenic University, Kavala, Greece

Abstract

Several contemporary software tools are available for measuring a range of object-oriented software quality metrics. In this study, we examine the efficacy of our automated tool, SQMetrics, which has been developed for educational purposes and can calculate the most popular metrics that have been proposed over time, giving special importance to object-oriented metrics. The tool is tested experimentally for its assistance in the evaluation of students' Java programming assignments through statistical analysis. The results showed a positive correlation between the instructor's rating and the overall quality index extracted from the software, indicating that the software tool could be a reliable assistant to instructor's grading under certain circumstances.

Keywords: software quality, code quality, JAVA code assessment, object-oriented software metrics, metrics tool

Introduction

Object-oriented software quality metrics have recently gained significant attention in the software industry in recent years. As software development has become more complex, the need for effective software quality metrics has grown. Researchers have identified several metrics that can be used to evaluate object-oriented software quality, including cohesion, coupling, complexity, inheritance, and polymorphism (Briand et. al., 1999). However, selecting the appropriate metrics can be challenging, as different metrics may be appropriate for different software development scenarios.

Several contemporary software tools are available for measuring object-oriented software quality metrics. These tools provide developers with a range of metrics that can be used to evaluate software quality (Lanza & Marinescu, 2007). Examples of such tools include SonarQube, Understand, Eclipse Metrics Plugin, CAST, etc. These tools can help developers identify potential problems in their software and improve its overall quality. However, when it comes to education, such tools may not be suitable for students and teachers of Software Engineering courses, since they were not made for educational purposes. Actually, most of them are commercial and rather complicated for a student to use.

To overcome these shortcomings, we have developed SQMetrics, an object-oriented software quality metrics tool, especially for educational purposes considering various pedagogical issues (tool design, educational objectives, pedagogical strategies, etc.). The tool provides a simple GUI, which is suitable for inexperienced users (such as Software Engineering students), together with a fully parametrized environment in order to support different methods of computation for certain metrics. The tool could be useful to teachers and students by providing a quantitative measure of the quality of their code. Students can benefit from such measures since they can identify areas for improvement and track their progress

over time. Additionally, the use of the tool can help students develop critical thinking skills and a deeper understanding of object-oriented principles by analyzing and improving the quality of their Java code.

The tool can also help teachers assess students' performance. In this study, we examined the quality of code written by Master Degree (MD) students in a Software Quality course on a JAVA programming assignment. Together with all other metrics supported, each student submission's total quality index was calculated by SQMetrics. The instructor also graded these submissions using his own grading criteria and statistical analysis was performed to compare these two outcomes. The objective of this study was to examine the efficacy of automated tools such as SQMetrics as a teacher's assistant in determining the JAVA code quality of student assignments.

The paper is organized as follows. The next section provides a short theoretical framework about our tool (SQMetrics) and the metrics it supports. This is followed by research methodology of the study, results, discussion, and conclusion.

The SQMetrics Tool

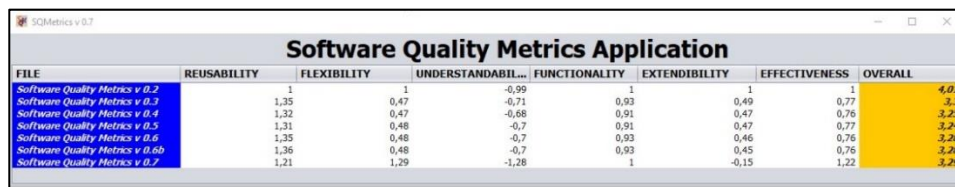
The developed *SQMetrics* (Software Quality Metrics) tool is an easy-to-use open-source tool capable of calculating a complete set of quality metrics for code written in Java, with an emphasis on object-oriented metrics, to support both students and teachers involved in a Software Engineering academic course.

Our tool is capable of calculating the most common code size metrics (Lines of Code-*LOC*, Logical Lines of Code-*LLOC*, and Lines of Comments-*LC*), all metrics proposed by Chidamber & Kemerer (1991), all metrics included in the 3rd level of the hierarchical Quality Model for Object-Oriented Design (*QMOOD*) model, as proposed by Bansiya & Davis (2002), as well as the quality characteristics of the 1st level of this model. *QMOOD* metrics were included in the tool as they allow for overall high-level quality indices to be calculated (Figure 1).

Namely, our tool calculates the 3rd-level metrics of *QMOOD* (DSC, NOH, ANA, DAM, DCC, CAM, MOA, MFA, NOP, CIS, NOM), the 1st-level quality characteristics of *QMOOD* (REUSABILITY, FLEXIBILITY, UNDERSTANDABILITY, FUNCTIONALITY, EXTENDIBILITY, EFFECTIVENESS), the common 'Lines of Code' metrics (PLOC, LLOC, LC) and the most popular object-oriented metrics that are commonly used in research studies (DIT, NOC, CBO, WMC, LCOM1, LCOM2, LCOM3, RFC). Several research studies identify these metrics' popularity (Barkmann et. al, 2009; Basili et. al., 1996).

From the above metrics, *SQMetrics* can produce overall quality indices (*OQIs*), which can prove useful in several educational tasks, e.g. comparison between projects, improvement of code quality, projects' assessment, etc.

The interested reader can find more about the *SQMetrics* tool in (Σωφρονάς, 2020).



FILE	REUSABILITY	FLEXIBILITY	UNDERSTANDABIL...	FUNCTIONALITY	EXTENDIBILITY	EFFECTIVENESS	OVERALL
Software Quality Metrics v 0.2	1	1	-0,99	1	1	1	4,01
Software Quality Metrics v 0.3	1,35	0,47	-0,71	0,93	0,49	0,77	3,7
Software Quality Metrics v 0.4	1,32	0,47	-0,68	0,91	0,47	0,76	3,25
Software Quality Metrics v 0.5	1,31	0,48	-0,7	0,91	0,47	0,77	3,24
Software Quality Metrics v 0.6	1,35	0,48	-0,7	0,93	0,46	0,76	3,28
Software Quality Metrics v 0.6b	1,36	0,48	-0,7	0,93	0,45	0,76	3,28
Software Quality Metrics v 0.7	1,21	1,29	-1,28	1	-0,15	1,22	3,29

Figure 1. Metrics calculated from the *SQMetrics* tool.

Testing SQMetrics assistance in the evaluation of student's projects

The SQMetrics tool was tested experimentally to see if it can be of assistance to Software Quality instructors in order to assess their students' projects. This generic research goal was broken into three research questions, aligned with the objective of this study:

- **RQ1.** *Concerning the same Java code assignment for all students, is the overall quality index indicative of the best project?*
- **RQ2.** *Concerning different assignments, is the overall quality index indicative of the best project?*
- **RQ3.** *Which design properties are indicative of the best projects?*

Data Collection

This study focused on the analysis of three Java assignments that were given to MD students in a Software Design and Management course. Each project was the assignment of a certain year course and the data were collected for three consecutive academic years. Each year's students' task was to implement the system classes of a specific real-world case study in the Java programming language. The domain analysis (including the creation of class diagrams) was the aim of previous assignments throughout the academic year. Students were asked to provide fully-functional Java source code to implement the project's requirements, written in any Java software development environment. This assignment was given during the last course of the MD program, when students were already familiar with Java Programming, in the context of the cognitive subject "Software Management and Quality". Students' age ranged from 22 to 54.

Data collected from the three consecutive academic terms were divided into the respective three data sets. Only complete and functional projects were included in our sample. Considering that the specific assignment was not mandatory for completing the course requirements (but gave extra points to those who completed it), only a part of the students submitted a complete Java project. More specifically:

- **Dataset 1:** 5 out of 17 submissions were considered, as the rest 12 were not qualified (either as incomplete or faulty). 4 of them (80%) were male and 1 of them (20%) was female in this sample.
- **Dataset 2:** 9 out of 11 submissions were considered, as the rest 2 were not qualified (either as incomplete or faulty). 8 of them (89%) were male and 1 of them (11%) was female in this sample.
- **Dataset 3:** 8 out of 14 submissions were considered, as the rest 6 were not qualified (either as incomplete or faulty). 7 of them (88%) were male and 1 of them (12%) was female in this sample.

All the assignments were manually graded by the same instructor, who used predefined standards that evaluated code logic, syntax, style, completeness, and functionality. The instructor's manual grade for each assignment was considered in our statistical analysis, conducted in SPSS Statistics.

Data Analysis and Results

SQMetrics was used to successfully analyze the 22 Java projects submitted by the students. SQMetrics can extract and compute all useful metrics of the QMOOD model: the 11 metrics of Level 3 (which correspond to the 11 design properties of Level 2), as well as the 6 high-quality attributes of Level 1. Integrated into a single output, an overall quality index (*OQI*) for each project is derived from the sum of the quality attributes:

$$OQI = \text{Reusability} + \text{Flexibility} + \text{Understandability} + \text{Functionality} + \text{Extendibility} + \text{Effectiveness} \quad (1)$$

SQMetrics also uses another way of normalization, suggested by Bansiya & Davis (2002), in order to compare different projects. In order to compare each term's projects, the sum of the normalized values for the 11 design properties can be used as follows (*TDPI* stands for Total Design Properties Index):

$$TDPI = \sum_{i=1}^{11} DPi \quad (2)$$

To further normalize *TDPIs* for all projects (independent from the term), each *TDPI* value is divided by the total amount n_{term} of analyzed submissions of the term it belongs, in order to get a normalized value *TNDPI* ϵ (1..11) for each project:

$$TNDPI = \frac{TDPI}{n_{term}} \quad (3)$$

Finally, the manual instructor-based score to measure the code quality of each project was used as the Instructor Index (*II*), to determine whether an automated code quality analyzer tool such as SQMetrics could provide code quality measures that would be assistive to manual grading by instructors. *II* is actually the instructor assessment's grade on a scale of 0-10. Table 1 shows the descriptive statistics for *II*, *OQI*, and *TNDPI* values.

Table 1. Descriptive statistics for *II*, *OQI* and *TNDPI* values.

	N	Minimum	Maximum	Mean	Std. Deviation
<i>II</i>	22	4,30	9,80	7,3727	1,56363
<i>OQI</i>	22	4,38	13,15	6,5623	2,02829
<i>TNDPI</i>	22	5,60	10,13	8,3606	1,05802
Valid N (listwise)	22				

RQ1

By examining the highest *II*, *OQI*, and *TNDPI* values (which indicate the best project) for each dataset, we can observe that most of the time the indices agree with the highest value, but this is not always the case. More particularly:

- The best assignment of dataset 1 (according to *II*) has also the greatest value in *OQI*, however, it is ranked 2nd according to *TNDPI*.
- *II*, *OQI*, and *TNDPI* are in total agreement concerning the best assignment of dataset 2.
- The best assignment of dataset 3 (according to *II*) has also the greatest value in *TNDPI*, however it is ranked in a relatively low position according to *OQI*.

To statistically answer *RQ1*, linear correlation analysis via Pearson's coefficient (r) was conducted on the data of each dataset. This kind of analysis concerns the verification of the existence of a correlation between two scale/ordinal variables. It shows whether there is statistical evidence that the correlation exists as well as the direction and intensity of the relationship between the variables.

By applying correlation analysis to each dataset separately, it appears that only dataset 2 has a statistically significant finding, therefore we cannot generalize the relationship for all datasets.

RQ2

However, by applying correlation analysis to the entire sample, it appears that the *TNDPI* indicators have a statistically significant correlation with the instructor's rating. In particular, *there was a significant positive relationship between II and TNDPI, [$r(22) = .435, p = .043$] (Table 2)*. This specific finding shows that *TNDPI* could be used as an auxiliary indicator in the grading of such projects by the instructor. Graphs showing the *Grade (II)*-*OQI* and *Grade (II)* - *TNDPI* value-pairs for all assignments can be seen in Figures 2 and 3 respectively. Correlation between *II* and *TNDPI* can also intuitively be observed in the corresponding graph.

On the other hand, the analysis showed that there is no correlation between the variables *II* and *OQI*.

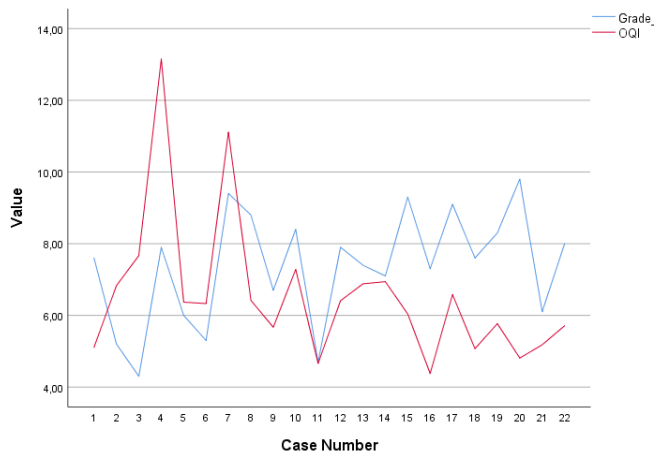


Figure 2. Grade (II) vs OQI values for all 22 assignments.

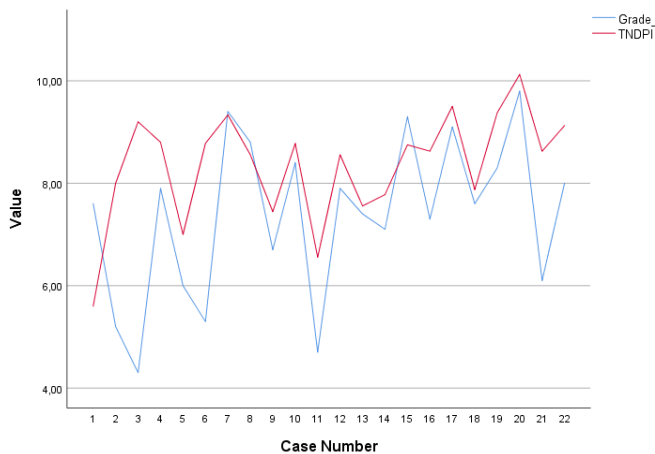


Figure 3. Grade (II) vs TNDPI values for all 22 assignments.

Table 2. Pearson correlation analysis in SPSS.

		II	OQI	TNDPI
<i>II</i>	Pearson Correlation	1	0,154	0,435*
	Sig. (2-tailed)		0,494	0,043
	N	22	22	22
<i>OQI</i>	Pearson Correlation	0,154	1	0,253
	Sig. (2-tailed)	0,494		0,256
	N	22	22	22
<i>TNDPI</i>	Pearson Correlation	0,435*	0,253	1
	Sig. (2-tailed)	0,043	0,256	
	N	22	22	22

*. Correlation is significant at the 0.05 level (2-tailed).

To further examine this relation, we performed another statistical comparison between the grades of the instructor and the quality index *TNDPI*. Since they do not have the same arithmetical limits, the non-parametric Wilcoxon signed-rank test is suitable for this type of comparison. The test was performed to compare the two paired samples and the results are presented in Table 3.

Table 3. Paired Samples Test between Instructor's Grade and TNDPI

		TNDPI - Grade
Z		-1,120 ^a
Asymp. Sig. (2-tailed)		0,263

a. Based on positive ranks.

As the statistical analysis shows, there is no significant difference between the two grading methods. This provides further evidence that *TNDPIs* could be used as a reliable and valid assistant for the instructor's grades.

RQ3

In order to check which design properties best describe the highest graded projects by the instructor in the whole dataset (if any), we need to normalize the values of all 11 design properties (*DP*), calculated by *SQMetrics*. The new variables are:

$$DPNi = \frac{DPi * 10}{n_{term}} \quad (4)$$

and all get a normalized value $DPN \in (0..10)$.

Correlation analysis of total data with reduction of rankings per dataset showed that four variables have a statistically significant correlation with the instructor's rating. In particular:

- there was a significant positive relationship between *II* and *DESIGN SIZE*, [$r(22) = .485$, $p = .022$]
- there was a significant negative relationship between *II* and *INHERITANCE*, [$r(22) = -.503$, $p = .017$]
- there was a significant positive relationship between *II* and *MESSAGING*, [$r(22) = .510$, $p = .015$]
- there was a significant positive relationship between *II* and *COMPLEXITY*, [$r(22) = .623$, $p = .002$]

These findings could serve in two ways:

1. reveal the inner code quality metrics that mostly affect the instructor's grading, and
2. lead to weight modification of each variable for calculating the total quality indices of a project.

One of the biggest advantages of this model is that it can easily accept parameterizations to adapt to new goals and incentives. At the lowest level, the metrics can be changed and a different set of quality attributes can be used. In addition, as previously stated, the weights on the quality attributes can be modified, and the attributes themselves can be replaced by others if desired to improve the overall indicators. Finally, a different normalization approach can be used, e.g. (Chawla & Chhabra, 2013). Correlation analysis between *DPNs* and *II* for each dataset separately showed no consistency among datasets in the statistical results.

Discussion

In the results of the presented statistical analysis, a positive correlation between the instructor rating and the overall *TNDPI* quality index extracted from the *SQMetrics* software is shown. Furthermore, no statistically significant difference is presented between these two measurements after performing a non-parametric paired samples test. These two observations give a first indication that the overall quality indices derived from the *QMOOD* model (and calculated by the *SQMetrics* tool) can be a useful aid for the evaluation of Java code submitted as an academic project in a related course under certain conditions.

A key requirement is that the way the instructor scores (i.e. his own criteria and their relevant weights) is close to the quality metrics calculated by *SQMetrics* so that there is convergence in the calibration. This is not always the case (especially if the measurable criteria are not clearly defined), as different instructors may grade code with different qualitative and quantitative criteria. Note that some instructors may even grade in a completely intuitive manner. Still, the quality criteria defined by *QMOOD* is an objective measure of object-oriented software quality and a useful reference to assess and compare code.

Since the number of assignments used for the evaluation is small, a second condition is to conduct additional research with more student projects and different instructors, so that, if valid, the efficacy of the *SQMetrics* tool as a teacher's assistant can be generalized. It should be noted here that for many instructors code grading does not necessarily mean code quality grading. Let's also not forget that for the study presented, only projects that were "running" and satisfied the functional requirements of the assignment were considered. It is not clear that projects that were not functional would have lower quality metrics values in *SQMetrics* (although some metrics are actually expected to have rather lower values, e.g. *LOC*). In such cases, it seems that the instructor's judgment on the grade is decisive, e.g. a project that performs only 2 of the 4 requested functions would receive a mediocre grade from the instructor, even though some quality characteristics (e.g. *understandability*) would have a higher ranking in *SQMetrics* than other complete and fully-functional submissions.

SQMetrics tends to give all the quality components the same weight. However, instructors usually provide different weights to individual code metrics based on the importance of that component. To identify which factors are the best predictors of the instructor's grade, multiple linear regression analysis could be used in a future study to model the relationship between the instructor's grade and the 1st level quality characteristics calculated by the *SQMetrics*. Moreover, depending on the components' selection different reports may be produced for both the student's and the teacher's perspective (Cipriano et. al., 2022).

Although we have indications that *SQMetrics* can be a reliable assistant to instructor grading, which can save time and resources while maintaining grading consistency, further research is necessary to determine the extent of the software's applicability and whether it can

replace instructor grading in all contexts (Gikandi et. al., 2011). It should be also pointed out that there is ongoing research in the field of automatic grading of student projects, such as using machine learning algorithms to grade Java projects and creating intelligent tutoring systems for Java programming (Al-Shawwa et. al., 2019). This research can provide educators with new opportunities to assess students' Java code in a more efficient and objective manner.

Conclusions

In this exploratory study, we examined the efficacy of the automated tool SQMetrics in determining JAVA code quality. The results showed a positive correlation between instructor rating and the overall quality index extracted from the software, indicating that the software tool can be a reliable and accurate assistant to instructor grading. The absence of a statistically significant difference between the two measurements further supports this statement. However, this research is a preliminary study and has its limitations.

Automated tools, like SQMetrics, are useful since they can help students become better coders by providing them with an easy mechanism to check the quality of their codes before assignment submission. The use of technology in assessment can increase student engagement and provide immediate feedback, which can improve their learning outcomes.

Overall, the findings suggest that the software tool can be a promising assistant to instructor grading, but additional research is needed to determine the full extent of its applicability and potential limitations. Further evaluation of the software's reliability, validity, and bias, as well as its effectiveness in different contexts, can help to inform decisions about the use of the software tool for assessment.

References

- Al-Shawwa, M. O., Alshawwa, I. A., & Abu-Naser, S. S. (2019). An intelligent tutoring system for learning java, *International Journal of Academic Information Systems Research (IJAIRS)*, 3(1), 1-6.
- Bansiya, J., & Davis, C. G. (2002). Hierarchical model for object-oriented design quality assessment. In *IEEE Transactions on Software Engineering*, SE 2002, 28(1), 4-17.
- Basili, V. R., Briand, L. C., & Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, 22(10), 751-761.
- Barkmann, H., Lincke, R., & Löwe, W. (2009). Quantitative evaluation of software quality metrics in open-source projects. In *2009 International Conference on Advanced Information Networking and Applications Workshops* (pp. 1067-1072). IEEE.
- Briand, L., Daly, J., & Wüst, J. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, 25(1), 91-121.
- Chawla, M. K., & Chhabra, I. (2013). Capturing OO Software metrics to attain quality attributes—a case study. *International Journal of Scientific & Engineering Research*, 4(6), 359-363.
- Chidamber, S. R., & Kemerer, C. F. (1991). Towards a metrics suite for object oriented design. In *Proceedings of the 6th ACM Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA)*, (pp. 197-211), Phoenix AZ.
- Cipriano, B. P., Fachada, N., & Alves, P. (2022). Drop Project: An automatic assessment tool for programming assignments. *SoftwareX*, 18, 101079.
- Gikandi, J. W., Morrow, D., & Davis, N. E. (2011). Online formative assessment in higher education: A review of the literature. *Computers & Education*, 57(4), 2333-2351.
- Lanza, M., & Marinescu, R. (2007). *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media.
- Σωφρονάς, Δ. (2020). Σχεδιασμός και Ανάπτυξη Εφαρμογής Υπολογισμού Μετρικών Ποιότητας Λογισμικού. Μεταπτυχιακή Εργασία, Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα (ΠΛΣ), Ελληνικό Ανοικτό Πανεπιστήμιο (ΕΑΠ). <https://apothesis.eap.gr/archive/item/78153>