DOI https://doi.org/10.30525/978-9934-26-597-6-29

SELECTING METRICS FOR QUANTIFYING SOFTWARE QUALITY

Maiia Liuta
Cherkasy State Business-College, Ukraine
Corresponding author's e-mail: maiialiuta@gmail.com

Abstract

This paper investigates the relationship between quality attributes and software quality metrics to support informed decision-making in software product quality assessment. The study includes an analysis of the interchangeability of metrics within categories and emphasizes their importance in assessing critical attributes such as security, performance, and maintainability. It concludes that these metrics are essential for enhancing software testing, project management, and reducing implementation costs, thereby improving overall software quality.

Keywords: quality attributes, software quality metrics, informed decision-making, code quality, interchangeability of metrics.

Introduction

An automated risk assessment system (ARAS) is a Over the years, hundreds of metrics have been created to evaluate code quality, monitor the performance of both software and development teams, schedule work tasks, etc. There are many reasons to use them for evaluating of the work done. They prevent bugs, help improve overall project planning, encourage process improvement and more thorough security analysis, and more. During testing, metrics improve the relationship between test coverage, risk, and system complexity, and once automated, can increase the profitability of a software product over time.

The traditional classification distinguishes 5 groups of metrics:

1) product metrics quantify the characteristics of a software product;

2) process metrics evaluate the characteristics of software development processes;

3) internal metrics help to measure all the properties that are important for a software developer;

4) external metrics help to assess the properties that are important to the user;

5) project metrics provide a system of indicators for monitoring project progress.

From the point of developers' view, metrics should quantify the quality level of the created software, which is closely related to the quality attributes of the system. The study of such connections is the subject of many scientific publications [1-4], however, a matter of particular interest is the more practical question: which metrics are the most indicative and widely used. In particular, in the context of software quality analysis tools, many aspects can be evaluated: implementation security, performance, efficiency, maintainability, modification, etc.

The purpose of the study was to systematise and build sets of software metrics for assessing the quality of product code in terms of functional and non-functional requirements.

One of the most systematic proposals for selecting subsets of metrics to assess compliance with non-functional requirements is given in [1]. The authors identify sets of key quality attributes for different types of software (web applications, embedded systems, information systems, distributed systems, database applications, and general-purpose applications), while matching them with the stages of the development life cycle. For each quality attribute, a set of indicators is proposed to assess it.

A similar analysis, but with reference to programming paradigms and domain, was conducted in [3]. It is expected that the vast majority of publications on metrics relate to object-oriented code. The authors also showed that the development of new metrics is a continuous process over time, which emphasises the need for up-to-date sets of code quality metrics.

In a broader context, [4] discusses the methodology for selecting and evaluating software quality metrics. Depending on the level of the metric, the authors define the quality metrics that are most commonly found in scientific publications. The following conclusions can be drawn regarding code quality assessment [5]:

– McCabe and Halstead's metrics describe the quality at the method level, and Chidamber and Kemerer's set of metrics represents the quality at the class level;

– MOOD and QMOOD metrics are the most widely used in scientific publications;

– size metrics are mainly related to the assessment of the code's error-prone nature;

– coupling and cohesion metrics are mostly mentioned in publications to measure the level of ensuring such quality attributes as maintainability, understandability and reusability. Metrics related to inheritance and code complexity are also specified, based on which flexibility, understandability and reusability are measured;

– Other design-level metrics include interface density, ease of learning (average time to learn/ master the use of a component), clarity of error

messages, number of customisable metrics for the interface, IPCI (InterPackage Change Impact Index), a group of metrics related to GoF design patterns, IIPE (InterPackage Extension Index), IIPU (InterPackage Usage Index), IIPUD (InterPackage Usage Diversion Index), etc.

Conclusions

Thus, code quality metrics are not only a tool for assessing a certain set of indicators that allow you to control the current quality of a software product throughout its development life cycle, but also suggest ways to improve software quality. Certain groups of metrics also contribute to improving the quality of testing, more efficient software project management, which will further reduce the cost of software implementation and maintenance.

References

[1] A tertiary study on links between source code metrics and external quality attributes / U. Iftikhar та ін. Information and Software Technology. 2024. V. 165. P. 107348. URL: https://doi.org/10.1016/j.infsof.2023.107348.

[2] Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review / R. Jabangwe et al. Empirical Software Engineering. 2014. V. 20, № 3. P. 640–693. URL: https://doi.org/10.1007/s10664-013-9291-7.

[3] A mapping study on design-time quality attributes and metrics / E. M. Arvanitou та ін. Journal of Systems and Software. 2017. V. 127. P. 52–77. URL: https://doi.org/10.1016/j.jss.2017.01.026.

[4] Source code metrics: A systematic mapping study / A. S. Nuñez-Varela та ін. Journal of Systems and Software. 2017. V. 128. P. 164–197. URL: https://doi.org/10.1016/j.jss.2017.03.044.

[5] Software Product Quality Metrics: A Systematic Mapping Study / Colakoglu F. N., Yazici A., Mishra A. IEEE Access. 2021. V. 9. P. 44647–44670. URL: https://doi.org/10.1109/access.2021.3054730.

[6] Trusina, I., Jermolajeva, E., Abramov, V., Gopejenko, V (2024) World development assessment in an invariant coordinate system of energy units: The newly industrialized economies perspectives (2024). Journal of Infrastructure, Policy and Development., 2024, 8(3), 3110. DOI: https://doi.org/10.24294/jipd.v8i3.3110

[7] I Trusina, E Jermolajeva, V Gopejenko (2024) System-dynamic approach to assessing sustainable development: the example of the USA. Economic Science for Rural Development 2024, 259 DOI: 10.22616/ESRD.2024.58.026

[8] Nevliudov, S Novoselov, O Sychova, V Gopejenko, N Kosenko Decentralized information systems in intelligent manufacturing management tasks. Advanced Information Systems 8 (3), 100-110 https://doi.org/10.20998/2522-9052.2024.3.12