УДК 004.4.05

КОНТРОЛЬ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОПОМОГОЮ СТАТИЧНИХ АНАЛІЗАТОРІВ КОДУ

Євтушенко Д.А.
Науковий Керівник – ст. викладач. Терещенко Г.Ю.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
тел.: +38(050) 753-36-36, email: dmytro.ievtushenko@nure.ua

This article examines the importance of static code analyzers in software quality control, addressing their main principles, drawbacks, and limitations. The author highlights the need for appropriate tool selection, integration with version control systems, and development environments, as well as the establishment of code standards to improve software quality. Additionally, the article emphasizes the significance of effective collaboration between developers, architects, and project managers in successfully implementing static code analyzers in the software development process.

The subject of software quality control becomes more pertinent in the contemporary environment of information technology development. The dependability, productivity, and scalability of the software product are significantly influenced by the employment of efficient methodologies and procedures for analyzing software code quality [1]. Static code analysis is one of the most important tools for software quality assurance.

In contrast to dynamic analysis, which is based on program execution, static code analysis includes understanding program code without actually running it. Syntactic, semantic, and metric analysis are the three basic methods used in static code analysis. The quality of the software product may be impacted by common errors and code style violations, which are easier to spot with the use of static code analyzers [2].

Static code analyzers do, however, have some shortcomings and restrictions. False positives in particular are conceivable and can have a detrimental effect on the software development process. The ability of static analyzers to identify context-dependent mistakes may also be limited [3]. Additionally, the constraints for code style can make it difficult to incorporate static analysis in the development process and may require the programmer to alter their approach to writing code.

The technique of automating software code quality control is made possible by the integration of static analyzers with version control systems and development environments. Such integration enables the tracking of code modifications as well as the early identification and correction of errors.

Additionally, the constraints for code style can make it difficult to incorporate static analysis in the development process and may require the programmer to alter their approach to writing code.

It is crucial to select the right tool that takes into account the programming language and project details for the implementation of static analyzers in the development process [4]. Different analyzers have unique characteristics, benefits, and drawbacks, thus the decision should be well-founded.

The technique of automating software code quality control is made possible by the integration of static analyzers with version control systems and development environments. Such integration enables the tracking of code modifications as well as the early identification and correction of errors.

By developing and putting into practice coding standards that follow quality control specifications, high-quality software is made feasible. Code standards facilitate reworking and debugging as well as promote uniformity, readability, and comprehension of the code [1].

Software quality assurance tools like static code analyzers are essential. They enable the detection of widespread errors and violations of code style that affect the reliability and output of programs [2]. However, there are several drawbacks and limitations to static analyzers that should be considered while utilizing them in the development process. To solve these difficulties, the appropriate tool selection, communication with development environments and version control systems, as well as the development and implementation of coding standards that improve software quality, are required [3].

Static code analysis, when used correctly, can considerably raise the quality of software by assisting in the early identification and rectification of faults as well as the preservation of good coding practices [4]. Static analyzers can thus evolve into a crucial instrument for software quality assurance, boosting the effectiveness of the development cycle and enhancing the success of information projects.

References:

1. McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.

2. Fowler, M. (2018). Refactoring: Improving the Design of Existing Code. Addison-Wesley.

3. Naylor, D., & Coleman, M. (2010). Source Code Analysis: A Road Map. 32nd IEEE/ACM International Conference on Software Engineering. doi:10.1109/ICSE-COMPANION.2010.160

4. Spinellis, D. (2003). Code Reading: The Open Source Perspective. Addison-Wesley.