UDC 004.41:004.052.2

M. K. Milinchuk
V. V. Chopliak

STATIC CODE ANALYSIS AS A TOOL FOR IMPROVING SOFTWARE QUALITY IN MODERN APPLICATIONS

Vinnytsia National Technical University

Анотація

У даній роботі розглянуто статичний аналіз коду як ефективний метод забезпечення якості програмного забезпечення. Проаналізовано принципи роботи інструментів статичного аналізу, їх інтеграцію у процеси розробки та CI/CD-конвеєри. Особливу увагу приділено виявленню вразливостей безпеки, технічного боргу та порушень стандартів кодування на ранніх етапах розробки. Визначено переваги застосування статичного аналізу коду для підвищення надійності та підтримуваності сучасного програмного забезпечення.

Ключові слова: статичний аналіз коду, якість програмного забезпечення, SAST, технічний борг, CI/CD, безпека коду.

Abstract

This paper examines static code analysis as an effective method for ensuring software quality. The principles of static analysis tools and their integration into development processes and CI/CD pipelines are analyzed. Special attention is paid to the early detection of security vulnerabilities, technical debt, and coding standard violations. The advantages of applying static code analysis for improving the reliability and maintainability of modern software are determined.

Key words: static code analysis, software quality, SAST, technical debt, CI/CD, code security.

Introduction

Modern software systems are growing in scale and complexity, making manual quality control increasingly insufficient. Defects, security vulnerabilities, and poor coding practices introduced during development may remain undetected until late stages of the software lifecycle, leading to significant costs and operational risks [1]. Static code analysis has emerged as a powerful approach to addressing these challenges. Unlike dynamic testing, which requires executing a program, static analysis inspects source code without running it, enabling the identification of potential defects at the earliest possible stage of the development process [2]. The purpose of this paper is to analyze the principles, tools, and benefits of static code analysis in modern software engineering.

Research results

Static code analysis tools operate by parsing the source code into an abstract syntax tree (AST) and applying predefined rule sets or pattern-matching algorithms to detect anomalies [3]. These tools can identify a wide range of issues, including unused variables, null pointer dereferences, buffer overflows, SQL injection vulnerabilities, and violations of coding standards such as MISRA or CERT.

Among the most widely used tools are SonarQube, ESLint, Checkstyle, PMD, and Coverity. Each tool targets specific programming languages and provides configurable rule sets tailored to different quality objectives. SonarQube, for instance, supports over twenty-five programming languages and delivers dashboards with metrics on code coverage, code smells, duplications, and security hotspots [4].

One of the most significant advantages of static analysis is its seamless integration into continuous integration pipelines. When configured as a mandatory step in CI/CD workflows, static analysis tools automatically block code with critical violations from being merged into the main branch, enforcing quality gates at scale. This practice shifts quality assurance to the left in the development lifecycle, substantially reducing the cost of identifying and fixing defects compared to later-stage testing.

Static Application Security Testing (SAST) represents a specialized form of static analysis focused on identifying security vulnerabilities such as cross-site scripting (XSS), insecure cryptographic usage, and hard-coded credentials [5]. SAST tools are increasingly incorporated into DevSecOps workflows, ensuring that security considerations are addressed from the earliest stages of development rather than as an afterthought.

Despite their advantages, static analysis tools also present certain limitations. They may generate false positives, flagging code that is technically correct, which can erode developer trust and encourage rule suppression over time. Additionally, static analysis cannot detect runtime-dependent bugs or concurrency issues that only manifest during actual program execution.

Conclusion

In conclusion, static code analysis represents an indispensable tool in modern software quality assurance. By detecting defects, vulnerabilities, and code quality issues at the source level, it enables development teams to build more reliable and secure software while reducing downstream maintenance costs. The integration of static analysis into CI/CD pipelines and DevSecOps workflows is becoming a standard practice in professional software development. Future advances in AI-assisted code analysis are expected to further reduce false positive rates and significantly improve the accuracy of defect detection in complex software systems.

REFERENCES

1. OWASP Static Code Analysis. URL: https://owasp.org/www-community/controls/Static_Code_Analysis.
2. Introduction to Static Analysis / Synopsys. URL: https://www.synopsys.com/glossary/what-is-sast.html.
3. Abstract Syntax Trees and Static Analysis. URL: https://www.geeksforgeeks.org/abstract-syntax-tree-ast-in-java/.
4. SonarQube Documentation. URL: https://docs.sonarqube.org/latest/.
5. SAST – Static Application Security Testing / Gartner. URL: https://www.gartner.com/en/information-technology/glossary/static-application-security-testing-sast.

Мілінчук Максим Костянтинович – студент групи 6ПІ-24б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: maksimmilinchyk@gmail.com.

Науковий керівник: ***Чопляк Вікторія Володимирівна*** – викладач англійської мови, кафедра іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: nikavnuchkova@gmail.com.

Maksym K. Milinchuk – a student of 6PI-24b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: maksimmilinchyk@gmail.com.

Scientific Supervisor: ***Victoria V. Chopliak*** – teacher of English, Foreign Languages Department, Vinnytsia National Technical University, Vinnytsia, e-mail: nikavnuchkova@gmail.com.