Facultad de Ingeniería en Electricidad y Computación

Report

Software Ingeniering II

Student:

Tenesaca Merchán
Freddy Gilmar

Teacher

Ing. Mercedes Mawyin
Ing. Monica Villavicencio

I PAO II -
2023

REPORT CODE INSPECTION

Introduction

In the world of software development, code quality is essential to ensure optimal application performance. Early detection of errors and adherence to best programming practices are crucial aspects of achieving robust and reliable software. In this context, PMD is presented as a fundamental tool for source code analysis. PMD is a code analyzer that identifies common programming faults such as unused variables, capturing empty blocks, and creating unnecessary objects [1]. In addition to providing thorough code analysis, PMD can also verify that coding conventions and standards are being followed, focusing on defect prevention and detection. In this essay, we will explore how PMD is integrated into the development environment, how it is used to inspect source code, and how custom rule sets can be created to fit the needs of the project.

Development

Installing PMD in the Development Environment

To use PMD in the development environment, a correct installation is essential. First of all, you need to make sure that the Maven integration for Eclipse is configured. Most Eclipse distributions already include Maven tools, but if not, you can follow the steps outlined in Part 1 to successfully install Maven. Next, the Eclipse PMD plugin must be installed following the steps in Part 2. These installations allow PMD to be effectively integrated into the development environment.

Using PMD for Code Inspection

Once PMD is properly installed in Eclipse, code inspections can be performed easily. Part 3 describes how to configure a specific project to enable PMD. The tool is not activated by default, so the "PMD enabled" option must be checked in the project properties. Additionally, you can explore the rule sets that come with PMD, although it is possible to configure custom rule sets. Part 4 explains how to use PMD to verify code. By right-clicking on the project and selecting “PMD → Check Code”, PMD performs an analysis and displays all detected violations. Each violation is represented with a color indicating its priority, making it easy to identify critical issues and warnings.

Creating Custom Rule Sets

One of the most powerful features of PMD is the ability to create custom rule sets to fit the needs of a specific project. Part 5 describes how to create a custom PMD rule set in XML format. These rule sets can include entire rules or select specific rules from other rule sets. Additionally, it is possible to provide additional parameters to certain rules to customize their behavior. This capability allows developers to focus on specific aspects of their project and ensure that the code meets specific guidelines and standards.

Conclusions

Using PMD in software development is an essential practice to ensure code quality and efficiency. The PMD tool integrates seamlessly into the Eclipse development environment and adapts to projects of any size. Early detection of errors, adherence to best practices, and creation of custom rule sets are key advantages of PMD. The tool significantly contributes to improving code quality and preventing potential problems in software development.

Recommendations

It is recommended to use PMD regularly during the software development process. Setting up custom rule sets is especially useful for focusing on critical areas of the project. Additionally, it is essential that developers become familiar with PMD rules and priorities to get the most out of this tool.

References

[1] PMD Source Code Analizer: https://pmd.github.io/

[2]UsingRule Sets: https://maven.apache.org/plugins/maven-pmd-plugin/examples/usingRuleSets.html