Integration and visualization of software metrics for Java in VS Code

Graduate

Raphael Geiser

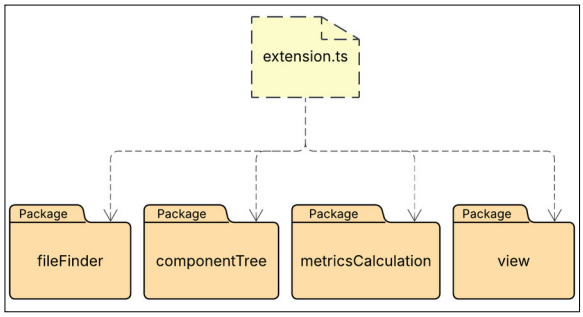
Initial Situation: In modern software development, the overview of the source code is quickly lost as project size and distributed teams increase. Growing complexity and lack of transparency lead to higher error susceptibility, rising maintenance costs, and more difficult quality assurance. Code metrics provide an objective approach to make software measurable in terms of structure, readability, and maintainability, and to initiate targeted improvements. They enable the early identification of potential problem areas, the formulation of verifiable guidelines, and support for refactorings. Many existing tools, however, are only available as standalone applications and are not integrated into the development process, which hinders their regular use. Visual Studio Code (VSC) is a widely used editor but currently does not offer an integrated solution for calculating metrics for Java projects. A corresponding extension therefore promises significant practical benefits.

Approach / Technology: The goal of this work was the development of a VSC extension that automatically calculates common software metrics and visualizes them directly in the editor. The implementation was carried out entirely in TypeScript using the VSC API and the efficient Tree-sitter parser, which converts Java source code into syntax trees. The composite pattern was used for the internal data structure to represent the hierarchical organization of packages, classes, and methods. The modular, extensible calculation of metrics is based on the visitor pattern, supported by generics for flexible return type design. The architecture comprises four main steps:

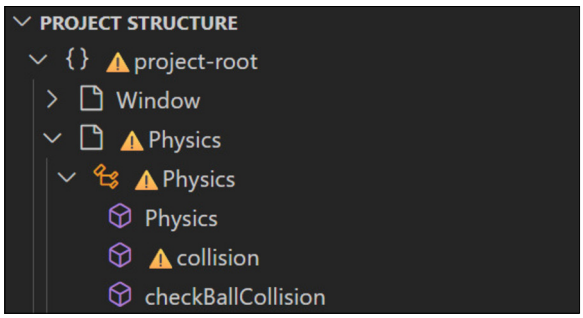
- 1 Locating and parsing the relevant files
- 2 Building the component tree
- 3 Calculating metrics using specialized visitors
- 4 Displaying the results via the VSC tree-view interface

Result: Within the scope of this work, four central metrics were implemented: Lines of Code (LOC), Cyclomatic Complexity (CC), Lack of Cohesion of Methods (LCOM), and selected Halstead metrics. The results are presented in an interactive, tree-like view that allows both a representation of the project structure and direct navigation to relevant code sections. Critical values are visually highlighted and propagated to parent elements, enabling efficient localization of problem areas. Through configuration, users can select which metrics should be calculated, allowing the solution to be flexibly adapted to project-specific requirements. The developed extension integrates seamlessly into the development process, reduces analysis effort, increases transparency, and supports developers in targeted refactorings. The implemented architecture provides a solid foundation for future extensions, such as project-specific thresholds, additional metrics, or enhanced visualizations, including temporal metric trends. Thus, the work contributes to the sustainable improvement of code quality in Java projects and opens up diverse opportunities for research and further development.

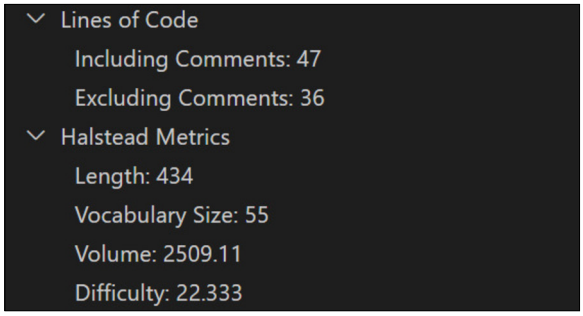
Overview of the key components of the developed software.
Own presentment



Interactive representation of the project structure. The exclamation mark indicates a potential problem area.
Own presentment



Display of the metric results for the selected component in the above view.
Own presentment



Advisor
Prof. Dr. Norbert Frei

Co-Examiner
Adrian Kretz

Subject Area
Computer Science