Zoltan Porkolab and his team:

Tibor Brunner:: Code comprehension with CodeCompass

Bugfixing or new feature development requires a confident understanding of all details and consequences of the planned changes. For long existing large telecom systems, where the code base have been developed and maintained for decades by fluctuating teams, original intentions are lost, the documentation is untrustworthy or missing, the only reliable information is the code itself. Code comprehension of such large software systems is an essential, but usually very challenging task. As the method of comprehension is fundamentally different from writing new code, development tools are not performing well. 4 During the years, different programs have been developed with various complexity and feature set for code comprehension but none of them fulfilled our special requirements. CodeCompass is an open source LLVM/Clang based tool developed by Ericsson and the Eotvos Lorand University, Budapest, to help understanding large legacy software systems. Based on the LLVM/Clang compiler infrastructure, CodeCompass gives exact information on complex C/C++ language elements like overloading, inheritance, the (read or write) usage of variables, possible calls on function pointers and the virtual functions - features that various existing tools support only partially. The wide range of interactive visualizations extends further than the usual class and function call diagrams; architectural, component and interface diagrams are a few of the implemented graphs. To make comprehension more extensive, CodeCompass is not restricted to the source code. It also utilizes build information to explore the system architecture as well as version control information when available: git commit history and blame view are also visualized. Clang based static analysis results are also integrated to CodeCompass. Although the tool focuses mainly on C and C++, it also supports Java and Python languages. Having a web-based, pluginable, extensible architecture, the CodeCompass framework can be an open platform to further code comprehension, static analysis and software metrics efforts.
https://github.com/ericsson/codeCompass

Gabor Horvath: Static Analysis with CodeChecker

Static analysis allows code optimisation, large-scale refactoring, code metrics and visualization in addition to discovering potential code defects. Static analysis involves the analysis of software without executing it, in almost all cases via an automated tool. This approach has its limitations, but it has been proven practical, and there are numerous issues and ideas leaving it an active field of research. The main idea behind static analysis' industrial application is cost-effectiveness. It has been statistically proven that the earlier an issue with the code is found, the easier, and cheaper it is to fix, while early triaging also enables easier long-term maintenance and code quality. For the C language family, the international open-source project LLVM/Clang
offers an easy to use, object-oriented syntax model, upon which the Clang Static Analyzer has been built. This tool allows the analysis of software code based on "checks", which are rulesets in the analyzer for finding call and data flows, control structures, and various other patterns that show potential defects in the code.

CodeChecker is an open-source defect visualizing effort backed by Ericsson Hungary, and Eötvös Loránd University Hungary, which allows users to easily integrate the analysis, defect report collection, and visualization of their large-scale projects. With a build command served as input, CodeChecker automates the configuration and execution of the analyzer, and the subsequent collection of analysis steps via which the analyzer found a particular defect, showing it in a Web application-based triaging interface.

In our lesson, we will investigate the basics of Static Analysis, familiarize ourselves with some lesser known features of the compilers, and learn the usage of the CodeChecker tool.