Component visualization methods for large legacy software in C/C++

Máté Cserép[a], Dániel Krupp[b]

[a]Eötvös Loránd University
`mcserep@caesar.elte.hu`
[b]Ericsson
`daniel.krupp@ericsson.com`

Abstract

Software development in C and C++ is widely used in the various industries including Information Technology, Telecommunication and Transportation since the 80-ies. Over this four decade, companies have built up a huge software legacy. In many cases these programs, implementing complex features (such as OS kernels, databases) become inherently complicated and consist of millions lines of code. During the many years long development, not only the size of the software increases, but a large number (i.e. hundreds) of programmers get involved. Mainly due to these two factors the maintenance of the software becomes more and more time consuming and costly.

To attack the above mentioned complexity issue, companies apply source code cross-referencers to help in the navigation and visualization of the legacy code. In this article we present a visualization methodology that helps programmers to understand the functional dependencies of artifacts in the C++ code in the form similar to UML component diagrams. Our novel graph representation reveals relations between binaries, C/C++ implementation files and headers. Our technique does not require any modification or documentation of the source code. It solely relies on the compiler generated Abstract Syntax Tree and the build information to analyze the legacy software.

Keywords: code comprehension, software maintenance, static analysis, component visualization, graph representation, functional dependency