Visualization Techniques of Components for Large Legacy C/C++ software

Máté Cserép, Dániel Krupp

Eötvös Loránd University, Ericsson

`mcserep@caesar.elte.hu, daniel.krupp@ericsson.com`

C and C++ languages are widely used for software development in 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 become inherently complicated by implementing complex features (such as OS kernels or databases), and consisting several millions lines of code. During the extended development time, not only the size of the software increases, but a large number (i.e. hundreds) of programmers get involved in the project. Mainly due to these two factors, the maintenance of these software products becomes more and more time consuming and costly.

To handle the above mentioned complexity issue, companies apply source code cross-referencers to help in the navigation and visualization of the legacy code. In our article we present a visualization methodology that assists programmers in the process of comprehending the functional dependencies of artifacts in a C++ source. Our novel graph representation not only reveals the connections between C/C++ implementation files, headers and binaries, but also visualizes the relationships between larger software components – e.g directories –, and provides a method for architecture compliance checking. The applied technique does not require any modification or documentation of the source code, hence it solely relies on the compiler generated *Abstract Syntax Tree* and the build information to analyze the legacy software.

References

[1] Michael Balzer and Oliver Deussen. Level-of-detail visualization of clustered graph layouts. In *Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on*, pages 133–140. IEEE, 2007.

[2] Pierre Caserta and Olivier Zendra. Visualization of the static aspects of software: a survey. *Visualization and Computer Graphics, IEEE Transactions on*, 17(7):913–933, 2011.

[3] Carsten Gutwenger, Michael Jünger, Karsten Klein, Joachim Kupke, Sebastian Leipert, and Petra Mutzel. A new approach for visualizing uml class diagrams. In *Proceedings of the 2003 ACM Symposium on Software Visualization*, SoftVis '03, pages 179–188, New York, NY, USA, 2003. ACM.

[4] Leo Pruijt, Christian Koppe, and Sjaak Brinkkemper. On the accuracy of architecture compliance checking support accuracy of dependency analysis and violation reporting. In *ICPC*, pages 172–181, 2013.