Caroline Rieder WIND RIVER

Department of Computer Science
University of Salzburg
Mentor at the University: Univ.-Prof. Dr. Wolfgang Pree

Obsolete C/C++ Symbol Detection

Wind River Systems is an embedded device software development company.
Along with several embedded operating systems, the company offers an IDE – called *Wind River Workbench*, which is based on the Eclipse-SDK. This IDE reuses several open source projects of Eclipse, like CDT. The major part of *Workbench* UI development is done in Salzburg.

As diploma thesis at the University of Salzburg, I would like to contribute a new feature to *Workbench*, which gives the user the possibility to identify and remove unreferenced symbols from an existing code base by analysing the whole software system.

In this context, *symbol* is used as the collective term for:

- Functions
- Variables

Unreferenced symbols are not desirable because they potentially bloat the size of the binary, waste memory, increase the maintenance effort or even cause bugs in the future, when making use of these symbols.

One possible scenario, where people really benefit from such a feature is the following:
Somebody gets in charge of maintenance or development of an existing code base and does not know the content of the project as a whole. This usually implies that he does not have a clear overview of the necessity of the code elements. Without such a feature, he will potentially be wasting hours or days by maintaining code, which is not used any more. As office-hours are expensive, using the planed feature could save lots of money for the company.

Within my diploma thesis the detection of following code elements is planed:

- Unreferenced symbols
- Unreferenced symbol–trees
- Unreferenced source files
- Unreferenced libraries

Different approaches of getting the necessary information will be covered within the paper, and potential risks and restrictions will be identified.

The feature will be implemented as part of *Wind River Workbench* and will offer the user following functionality:

- Analysis of the code base
- Visual demonstration of the findings
- Possibility for the user to decide, which code to remove
- Removal of the identified code

This kind of code-analysis is a long-awaited feature and would complement the range of functionality offered in *Wind River Workbench*.