Calculating Object Oriented Metrics from Large C++ Programs

István Siket and Rudolf Ferenc
University of Szeged, Department of Software Engineering
{siket|ferenc}@inf.u-szeged.hu

A critical issue in large-scale software development and maintenance is the rapidly growing size and complexity of software systems. It follows from this rapid growth that it is difficult to comprehend the source code of large programs. Applying different program metrics can help us to characterize object oriented programs, which means that we can express certain properties of the program's source code.

A lot of object oriented metrics have been defined which can be grouped according to which properties of the code is taken into consideration. We show only a few examples but it turns out that knowing the metrics about a program can help in finding errors and can be used during development and maintenance. For example, using cohesion metrics it can be checked if a class is designed well or it is too complex so it should be divided into two or more classes. Furthermore, if we want to use only some parts of the program (e.g. some classes) it is important that these parts do not depend too much on other parts because in that case they cannot be used without the rest of the program. These dependencies can be expressed by metrics as well.

However, using the metrics in practice is not so easy because if we want to calculate metrics for a large program we have to collect the necessary data from the source first. Basically a task similar to compilation must be done. The source files must be analyzed one by one and the desired properties of the code must be collected (we used the Columbus/CAN tool for analysis). But real-world systems contain several thousands of source files so doing this by hand is not feasible. We solved this problem by "wrapping" the compiler. This means that if we run the compiler, the wrapper will start instead of it, which executes first the original compiler (with the same parameters so the output remains the same), and second, it invokes Columbus/CAN as well with the same parameters and in the same environment. This works well also in the case of makefiles. This way all we have to do is to build a software system as usual (but with the wrapper switched on) and we get all necessary data.

We tested this approach on two real-world systems (Mozilla and Open Office), and using the collected data we calculated 88 different metrics with the help of Columbus.