Developing a smart versioning system

Prof. Natasha Sharygina and Grigory Fedyukovich

During its lifetime any program evolves. A developer can fix a bug, apply optimizations, or add a new functionality to the existent code. After such changes the developer should be sure that the new code is correct, i.e., it does not break any old functionality which is supposed to be preserved in the new version. In order to do it efficiently, a tool called eVolCheck [1] has been developed by researchers of Formal Verification and Security group at USI. eVolCheck formally verifies each version of a program and reuses some efforts (namely, function summaries) between consequent runs. For example, if just one function is changed in the program, it may be enough to re-check only the new code of this function, and do not touch the (preserved) rest of the program.

In the GUI of eVolCheck integrated with Eclipse IDE [2], the process is visualized as follows. First, the syntactic difference between the current version and the previous one is shown. Depending on the amount of the modified code, the user decides to run the upgrade checking tool. Then the Eclipse IDE internally runs eVolCheck, it configures the tool automatically and keeps the settings in a subsidiary storage, so the user does not need to adjust the environment for every check. After eVolCheck completes its tasks, the positive/negative result is returned to the user. In the former case, the change impact (namely, the number of re-checked summaries) is displayed - it represents the semantic influence of the change on the whole program. In the latter case, if a bug is found, the Eclipse IDE shows the trace to the counter-example.

Software versioning is the way to distinguish between the program versions [3]. Software versioning systems (e.g., SVN [4] or Git [5]), integrated to different IDEs, are widely used by software developers in industry and academia. But since it is always up to the user, whether to make a new revision, the existent versioning systems cannot give guarantees that the correspondent program version is correct. We propose to extend one of the existent software versioning systems to support program verification by eVolCheck. We will call it a “smart versioning system”.

The proposed smart versioning system should work as follows. If a current version of the program is proven correct, it might be a good justification to create a new revision in the versioning system. Otherwise,

if at least one assertion in the version is violated (i.e., a counter-example is given), the user should analyze the counter-example, then fix it, and finally run eVolCheck once again. So the revisions in the versioning system will become trustworthy. In our proposed smart versioning system, every new revision will be confirmed safe. We propose the student to implement such technology on the top of the existent tools.

We are sure the proposed technology can be easily implemented because its main ingredients already exist: the Eclipse IDE with available source code and APIs, eVolCheck, and eVolCheck plugin to Eclipse. The student should investigate how to use software versioning systems, and how these systems are being handled by Eclipse IDE. We believe such an integration will make easier to apply formal verification in most software development projects, which are currently unfortunately still not able to use this powerful technology. And this will ultimately increase the quality of software being developed.

There are two main benefits for a student who is going to implement the proposed technology. First, the student will become familiar with formal verification and state-of-the-art verification systems. It will give the young researcher a taste of what kind of research is being done in the modern Computer Science. No theoretical background in Formal Verification is required. Second, the student will earn an important experience in software development. Eclipse IDE is known in developers community, and experience in working with its source code will indicate the student is familiar with modern trends in software development. For this, initial experience in Java programming language is required.

References:

[1] G. Fedyukovich, O. Sery, and N. Sharygina. eVolCheck: Incremental Upgrade Checker for C. In *TACAS'13*
[2] http://eclipse.org/
[3] http://en.wikipedia.org/wiki/Software_versioning
[4] http://tortoisesvn.net/
[5] http://git-scm.com/