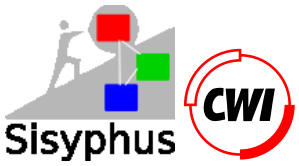
Sisyphus: Continuous Integration of Component-Based Product Lines

"Pacing the Heartbeat of Software Development"

Tijs van der Storm, storm@cwi.nl
Centrum voor Wiskunde en Informatica (CWI)
http://www.cwi.nl/~storm



Research Abstract

Continuous integration is considered to be an important practice during software development. By building and testing a software product on a frequent basis, bugs are detected early and a recent working version of the product is always available. However, in product line approaches, when many components and code-lines are maintained in parallel, building every variant of the product soon becomes impractical. Nevertheless, especially in such cases, the feedback that continuous integration generates may prove invaluable.

The first question in this research is how to make continuous integration in such multi-component, multi-project situations manageable, both in terms of software configuration management (SCM) and in terms of performance. Secondly, I want to investigate how continuous integration can be adapted to support the software delivery and customer support processes. This way, continuous integration can be seen as the first step towards continuous release.

The Sisyphus build system represents first steps in this research direction. It improves upon other continuous integration systems by maintaining a database that keeps track of which version of which component has been built in what project against which dependencies with what result. Formal reasoning on this database about both the evolution history and the build history of a component including its dependencies, facilitates the optimization of continuous integration.

The database helps in answering the following questions. Does this component have to be rebuilt at this moment? Which components are affected by commit X? Are there successful builds for the dependencies of this component? Which set of successful builds do we choose if there are multiple options? Such and other questions help to increase the average build frequency by applying the following principles:

- Only perform builds if there are affecting changes.
- Share builds across product variants and projects.
- Backtrack to earlier builds if the latest ones have failed.

Combining these three key ingredients allows Sisyphus to trade some of the currency of completed builds for increased success rates, thereby reducing time to market. Furthermore, because the Sisyphus database ensures traceability of built artifacts to the originating sources, build results can actually be used to support the software delivery and customer feedback processes. Exploring these directions further constitutes ongoing and future work.