Automating Traceability in Agile Software Development

THESIS SUMMARY
RICHARD SIMKO

Supervizor: Lars Bendix (LTH), Emil Sjödin (RefinedWiki)
Examiner: Ulf Asklund (LTH)

Keeping track of why changes are made in software development projects becomes harder and harder as sofware grows more complex, projects become larger and less centralized and more and more companies shift to agile. This leads to developers wasting a lot of time manually documenting why their changes are made. This Master's thesis explores the ability to automate these tasks.

The Current State of Traceability

When developing software today most functions of traceability are manual. For example when tracing changes in code to requirements, change requests, bug reports or similar, this is done through the developer writing the ID of the work item a particular commit connects to in the commit message. One can easily guess that this is prone to errors since IDs can grow long and complex which increases the risk of a developer inputing the wrong data. This then leads to data which can not be trusted which in many cases can be worse than not having any data at all.

Previous research shows that this is the case as well as documents the need for a more automated approach to generating traceability links to code. In addition to that a survey and a series of interviews are performed as part of the thesis, further strengthening the view that there is a need for automation in this area.

Solution

Instead of implementing a traceability solution which should fit all types of projects and work with every tool, as has been tried previously, this thesis was quickly narrowed down. The focus was to develop a prototype of a tool that can present the developer with suggestions regarding what work items are most likely to have been worked on in his current commit.

This is done by analyzing previous commits by the same developer as well as the state of tickets in the project/issue tracking system. This data is then aggregated and a few best guesses are generated for the developer to choose from. This completely eliminates the process of manually entering numbers and gets rid of the risk that a developer should enter incorrect numbers.

Results

The prototype proved hard to test due to technical constraints. As such no good data regarding its performance could be provided. Instead analysis of its performance was made on a more abstract level, through demonstrations and interviews with developers and configuration management experts. Overall the response was positive and it was commonly said that people wanted a tool like this. However it was interesting to note that there were several, especially CM professionals, persons who could not at all see the need for a tool like this. This proved to be related to how well traceability generation worked in their organization today, since they had good guidelines in place which were followed by everyone the need for a tool was non-existant.

Traceability Automation Framework

In addition to developing a prototype the thesis also creates a framework for analyzing other types of traceability and the possibility for automation. This was purely theoretical and the idea was to uncover which areas would be most interesting to focus on for future researchers. One thing which stood out was traceability links between failed test cases and issue reports, eg. being able to run a smoke test when an issue has been fixed greatly speeds up development. However in order to be able to do this today someone has to manually tie issue reports to failed test cases.