

Title: Code coverage improvement of database-centric applications

Academic team leader(s): Dr. ir. Ana Lucia Varbanescu (UvA), Dr. Clemens Grelck (UvA)

Involved partners: SNS Bank N.V.

Challenge:

When developing software, two aspects are of great importance: time-to-market and quality. Time-to-market influences the success of new features and allows for quick feedback (data) which in turn can be used for continuous improvement. The quality of the software itself is key for high availability as well as ensuring a good user experience. To allow for these aspects to be guaranteed, SNS Bank N.V. focusses on continuous integration/delivery including automating test-cases, as well as automating software quality assessment. Within this process there are two challenges. Firstly, the available instruments used to assess quality are often not accurate enough. Secondly, automation mainly focuses on the execution and re-execution of test-cases. Their construction is a manual process, which is costly.

The goal of this project is twofold: (1) we want to define metrics for code coverage quality assessment for database-centric applications (having complex logic in the embedded queries), and (2) we want to automate the construction of code coverage tests.

Metrics for Quality assessment

Typical code coverage tools are tailored for assessing a specific programming language, e.g. Java or C++. However, in reality, applications often consist of multiple programming languages. In our case the applications contain logic in both a programming language (e.g.: C++) as well as in a database query language (e.g. in SQL).

One of the automated quality indicators is the code coverage of all unit tests for each application.

Measuring code coverage provides insights in a number of metrics: line coverage, branch coverage, function coverage, etc.

Due to the mix of languages, current coverage tools give a too optimistic result: the reported coverage is expressed in terms of statements or branches of the application's main programming language, and ignores the database queries since these are present in an embedded form of which the tool is blind for. Database queries can contain quite complex logic, which is of course should not be left untested.

Therefore, SNS Bank N.V. is looking for a way to improve the current analysis tools with respect to reporting code coverage, to include both the programming as well as the query language. This will allow for a more accurate quality estimation, and the ability to take measures accordingly.

Leveraging automation

Once coverage can be accurately assessed, we like to automate the testing procedures. Because manually constructing test cases is a costly process, we want to complement it with algorithms that can generate test-cases automatically. Recent advances from science [1,2] that may lead to real breakthrough. For example, combined static analysis and evolutionary algorithms have been shown to be able to deliver better coverage than human testers on real world Java classes. However, these Java classes do not contain embedded database queries. Such queries are more challenging as they require tables to be generated, which are much more complex in structure than typical primitive typed values or objects in plain Java classes. We are looking for a solution for this, to help SNS Bank to improve the quality of its software products even further and shorten their time-to-market.

Outcome

At SNS we consider the case a success when having defined the metrics and having a proof-of-concept application demonstrating them, as well as gaining further insights in the concepts of generating testcases.

Relevant literature

[1] Javier Tuya, María José Suárez-Cabal, Claudio de la Riva. Full predicate coverage for testing SQL database queries, in *Journal of Software: Testing, Verification, and Reliability*, Volume 20, Issue 3, pages 237–288, 2010.

[2] U. Rueda, T.E.J. Vos, I.S.W.B. Prasetya, Unit Testing Tool Competition -- Round Three, in *IEEE/ACM 8th International Workshop on Search-Based Software Testing (SBST)*, IEEE, 2015.