SCIENCES &
BIOENGINEERING SCIENCES

The Research Group

Software Languages Lab

has the honour to invite you to the public defence of the PhD thesis of

Reinout Stevens

to obtain the degree of Doctor of Sciences

Title of the PhD thesis:

A Declarative Foundation for Querying the History of Software Projects

Promotor:

Prof. dr. Coen De Roover

Prof. dr. Viviane Jonckers

The defence will take place on

Friday, May 5, 2017 at 17:00

in Auditorium D.0.02 at the Campus Humanities, Sciences and Engineering of the Vrije Universiteit Brussel, Pleinlaan 2 - 1050 Elsene, and will be followed by a reception.

Members of the jury

Prof. Dr. Em. Theo D'hondt
Prof. Dr. Jan Hidders
Prof. Dr. Ann Dooms
Prof Dr. Peter Vranckx
Dr. Julia Lawall (Sorbonne Université Paris)
Prof. Dr. Xavier Blanc (Université de Bordeaux)

Curriculum vitae

Reinout Stevens (°1988), obtained a Master in Computer Science at the VUB in 2011. Supported by an IWT grant, he started to pursue declarative techniques for querying the history of software projects. In the course of the next 4 years, he co-authored 1 journal article, 2 full-length conference papers, and 5 short conference papers. He presented his results at several international workshops and conferences. For most of the past year, he contributed several prototypes to an IWT SBO project on change-centric quality assurance.

Abstract of the PhD research

The use of a Version Control System (VCS) is an industry best practice for developing software projects. VCS's enable developers to share and integrate changes made to the source code. As a side-effect, a VCS stores the entire development history of the versioned software project. This history is of interest to several stakeholders. Developers can use it to find answers to questions that might arise during the development. Examples of such questions are "Who introduced this piece of code?" or "Who has contributed to this failing class?". Researchers in the domain of mining software repositories can leverage the same history to analyze, spot trends, and retrieve actionable information about the development of software projects.
Although very insightful, software histories are too large to inspect manually. Support from a general-purpose history querying tool that satisfies the needs of the different stakeholders is in order. Such a tool automatically identifies the elements from a project's history that exhibit the characteristics specified in a given history query. We identify several criteria for such a general-purpose history querying tool. Among others, history querying tools should support the following characteristics in their queries:

- Version Characteristics concern the different elements of a particular version of a project. Examples of these characteristics are the author, commit message, or the state of the source code of a particular version.
- Temporal Characteristics concern the temporal quantification over the versions of the software project. Examples include quantifiers such as "for every version" or "after the first version that…".
- Change Characteristics concern the fine-grained edits made between two versions of the source code. Examples of these are the source code elements that have been inserted, moved, updated or removed.
- Evolution Characteristics concern the effect of applying a (sub)sequence of changes to source code. Examples are all change sequences that have the same effect as a known refactoring or earlier transformation of the code.

We propose a declarative foundation to history querying that supports these different characteristics. Characteristics are expressed in logic queries, while a logic proof procedure identifies history elements exhibiting the specified characteristics. We create and design the general-purpose history querying tool QwalKeko that supports the characteristics in a uniform declarative language. We validate our work through different means. We validate the support for version and temporal characteristics through several usage scenarios stemming from the aforementioned stakeholders. We validate the support for change characteristics by conducting a large-scale empirical study in which we investigate the co-evolution of functional tests with the system under test. We conduct this study twice; once using our approach to history querying and once using a general-purpose programming language. This enables us to compare both implementations on the different concerns of the study. Finally, we evaluate the support for evolution characteristics by identifying instances of refactorings in different change sequences stemming from open-source projects, and ensuring that the identified change sequences are correct, minimal and executable.