Multi-Version Program Analysis

– Dagstuhl Seminar Summary –

Thomas Ball · Stephan Diehl · David Notkin · Andreas Zeller

Public Outreach

Change is an inevitable part of successful software systems. Software changes induce costs, as they force people to repeat earlier assessments. On the other hand, knowing about software changes can also bring benefits, as changes are artifacts that can be analyzed.

In the last years, researchers have begun to analyze software together with its change history. There is a huge amount of historical information that can be extracted, abstracted, and leveraged:

- Knowing about earlier versions and their properties can lead to *incremental assessments.*
- Analyzing the history of a product can tell how changes in software are *related* to other changes and features.
- Relating properties to changes can help *focusing* on changes that cause specific properties.

In this Dagstuhl seminar, researchers that analyze software and its history have met and discussed for a full week, exchanging their ideas, and combining and integrating the techniques to build a greater whole. Clearly, understanding history can play a major role when it comes to understand software systems.

Scientific Highlights

The main concern of the seminar was the *synergy* of the individual approaches. Themes that emerged during the seminar included:

- The use of *bug databases* to judge whether changes were beneficial or not;
- The use of *advanced visualization techniques* that integrate program analysis and history; and
- The use of *version histories* to conduct empirical research, as in the study of clone genealogies.

Dagstuhl Seminar Proceedings 05261
Multi-Version Program Analysis
http://drops.dagstuhl.de/opus/volltexte/2006/559

The latter point – leveraging version histories to conduct empirical research – was maybe the strongest highlight of the seminar. As a direct result of the seminar, a mining challenge was introduced at the Workshop of Mining Software Repositories.

All in all, Software engineering is full of anecdotical evidence, often relying on insufficient or proprietary data. Publicly available change and bug histories may change this, providing reproducible benchmarks for empirical research, and allowing anyone to assess hypotheses about what works in software engineering and what does not.

Perspectives

The analysis of programs across multiple versions has a bright future. The workshop on mining software repositories has never been more active; researchers begin to recognize the great potential of software history to understand development processes. With this Dagstuhl Seminar, we are very happy to have contributed to this momentum. We also look forward to see this research being applied in practice, as it is already the case with IBM and Microsoft.

Key words

software engineering, data mining, software processes, software archives, version control, bug database, experimentation, measurement, verification.