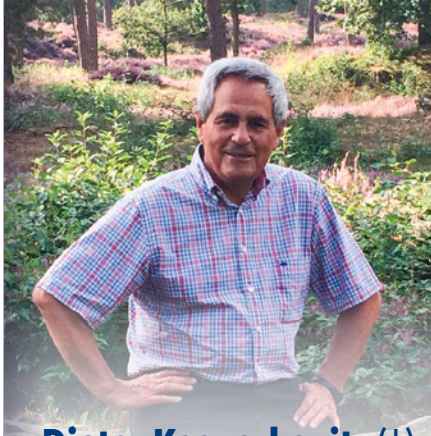
5 14 September - 12 October 2018

B&C

At the heart of high tech | bits-chips.nl

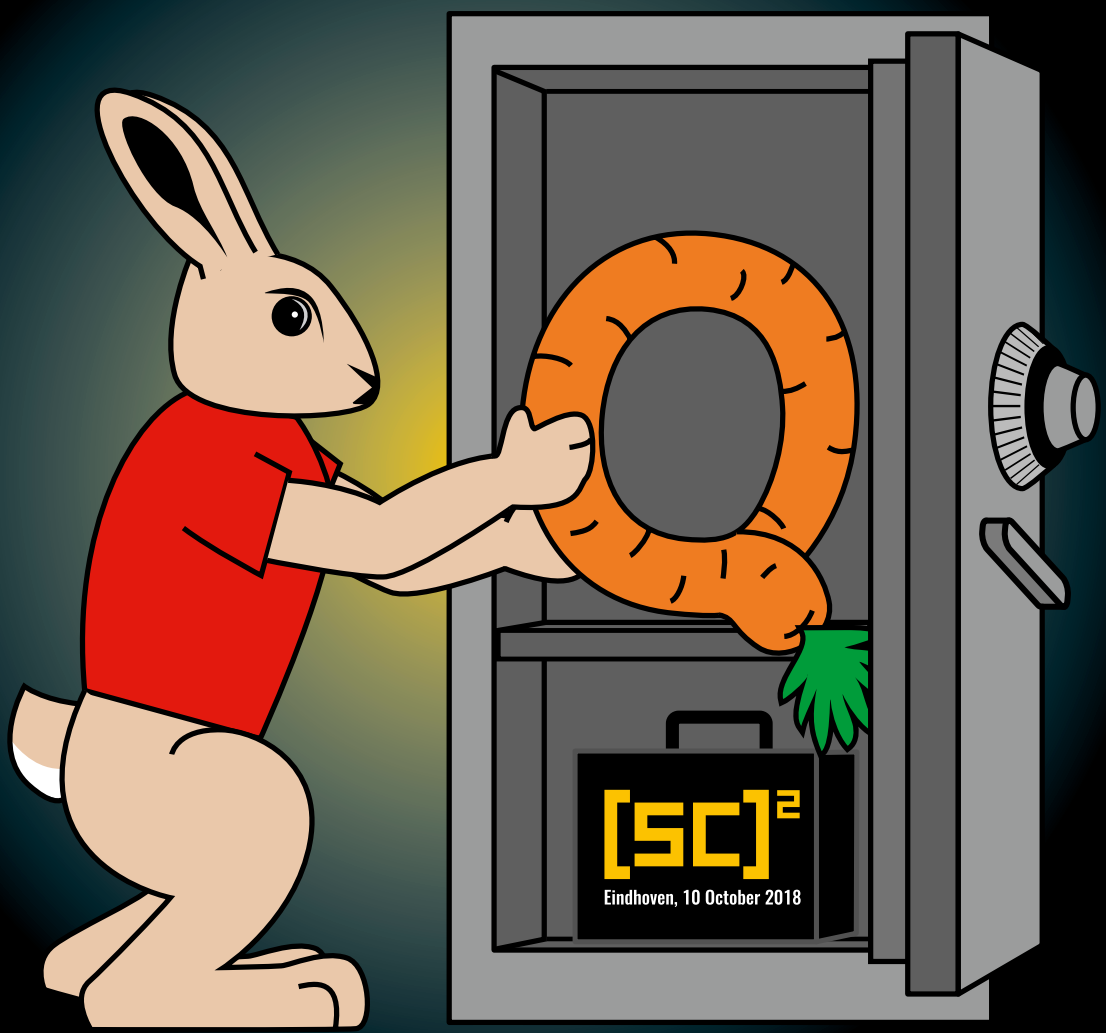
Single-chip radar

Waalwijk's Senz2 picks up Omniradar



Dieter Kasperkovitz (†)

Physicist and inventor to his dying breath



ENSURING QUALITY WHEN SCALING AGILE

HOW SOFTWARE ARCHITECTURE BECOMES A GAME CHANGER

Software architectures lay out the foundations for maintainable and sustainable development. In the past, an architecture model's obsolescence imposed a risk on both the model and the project's success. With hierarchical reflexion models we can continuously align the models and their implementation, so that we can leverage them at the highest level of significance throughout the system's lifecycle.

Thomas Eisenbarth
Daniel Simon

Over the decades software development process models have evolved from waterfall to agile. The more agile approaches promise a more flexible, faster and user-oriented path to success. As the number of iterative cycles in software development has increased, the idea of refactoring software has gained ground. Refactoring facilitates future development by improving the software's internal quality without changing its functionality for the end user.

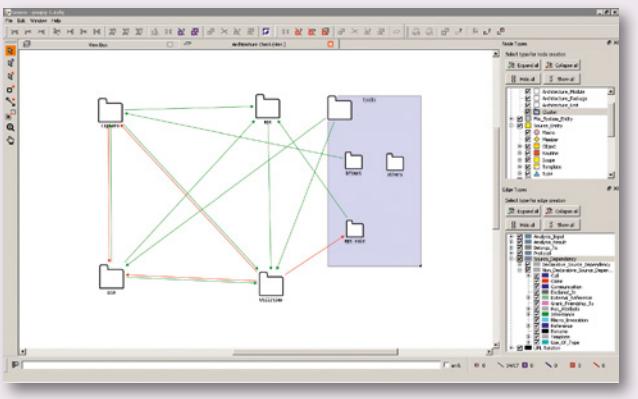
With refactoring as a remedy, the thinking goes, we can let software complexity grow and accept an increase in software erosion (also referred to as 'technical debt' or 'software decay') for a certain amount of time or a certain number of iterations and then explicitly plan a cleanup iteration, typically shortly before or after a release to production. While this idea of 'software gardening' (inspired by the idea of 'grow and prune') seems intriguing, it doesn't work out in practice, for reasons of time ('we don't have the time to clean up now'), risk ('we don't want to introduce new errors') or budget ('why should we change things when the end user won't even notice?').

The solution to overcoming these arguments is to adopt a continuous refactoring approach. Continuous refactoring on a daily schedule makes time, risk and budget much easier to assess and also more acceptable. Instead of waiting for software to erode and then implementing reactive fixes, you take control and act before erosion takes its toll. Experience and research show that an up-to-date, erosion-free software architecture is a huge lever for productivity. Ideally, an architecture is subject to continuous refactoring.

Reflexion analysis

A software architecture spans multiple development cycles as it gives strategic guidance to developers. But there's a downside: code changes will inevitably affect the software architecture sooner or later, teams must continuously verify that the implementation follows the architectural guidance and architecture creation requires significant investments. Hierarchical reflexion models make it possible to keep the software architecture and its implementation in continuous sync and to immediately detect deviations.

Hierarchical reflexion analysis compares the software architecture (as specified eg in a UML model) and its implementation (usually in source code) and gives instant feedback. As such, it facilitates communication in the development teams and completely eliminates misconceptions and misunderstandings. Reflexion analysis gives 'life' to models that formerly ran out of date in no time – it changes the game of software architecture-based development from obsolete models and divergent implementations to always up-to-date abstractions and tangible, comprehensible implementations.



Axivion Bauhaus Suite highlights architecture conformance continuously and lets teams resolve deviations immediately.

Thomas Eisenbarth is a seasoned expert in modelling software architectures, a technology evangelist and the cofounder of Axivion. Daniel Simon co-authored the research on hierarchical reflexion models with University of Bremen professor Rainer Koschke, and is now head of professional services at Axivion.

Edited by Nieke Roos