TAV 23 München "Special Track":

Neuere Entwicklungen im Testbereich der SIEMENS AG - CT SE 1

An Industrial Perspective on Model Based Testing

Andreas Ulrich

The presentation gives an overview of the current practice in software development projects in industry and discusses the problems that formal and model-based testing techniques face when they are attempted in an industrial setting. One of the main observations is that software is developed in evolutionary steps. Software is seldom developed from scratch and is never a one-time solution. Instead it evolves in increments and product versions. As a consequence of the iterative development, automated test execution must be supported to deal with the validation of daily builds and small releases. In this context, the selection of the right set of regression test cases and the definition of suitable test stop criteria become essential. Both issues are typically answered by experience taking into account the number and distribution of bugs discovered in earlier iterations and releases and later test phases, but are rarely addressed in formal, model-based testing approaches.

Moreover, model-based techniques face further challenges if they are applied in the different development phases of requirement analysis, design and modelling, coding and system integration, and testing. First of all, requirements of a software product are typically incomplete, late, changing over development time, and even outdated. It is therefore very unlikely that a complete specification of the system can be made available for model-based testing. Graphical modelling languages, e.g. UML, if used, turn out to be frequently unproductive because of the complexity of related tools and scalability issues. Users get easily distracted from their original modelling task because they spend much time on getting the graphical layout of their model right. Another aspect relates to the configuration management of specifications using a graphical language. Comparing differences between two graphical models is not an easy task that complicates any tooling.

In the coding phase one is faced with the fact that most parts of the software product are reused from previous versions or come in from third-party components. For this reason, system integration and validation becomes the cornerstone of a successful product quality assurance. Due to the use of model-based design approaches, e.g. MDA, an increasing percentage of the production code is generated from a model automatically. This aspect has an influence on model-based testing approaches. It might be not desirable to derive tests from the same design model, for example, because they would simply test the correctness of the code generator. Instead, testing techniques should be applied that focuses on the integration of the generated code in its environment.

Last but not least, testing must not concentrate on functional aspects only. Usability aspects, performance and integration aspects must be considered as well. It is hard to see that any single formal test method can be used to handle all these different requirements together. Consequently, formal test methods should focus on a particular requirement domain, in which they can have their largest impact. In the end, the best gauge for any new method is the degree of productivity it helps to boost.

Secrets of Test Driven Development

Peter Zimmerer

Test-driven development (TDD) is a new approach for software construction in which developers write automated unit tests before writing the code. These automated tests are always rerun after any codes changes. Proponents assert that TDD delivers software that is easier to maintain and of higher quality than using traditional development approaches.

Based on experiences gained from real-world projects employing TDD, Peter Zimmerer shares his view of TDD's advantages and disadvantages and how the TDD concept can be extended to all levels of testing. Learn how to use TDD practices that support preventive testing throughout development and result in new ways of cooperation between developers and testers.

Take away practical approaches and hints for introducing and practicing test-driven development in your organization.

Integration Testing - Looking for a Solution to Testing Concurrent Components and Systems

Andrej Pietschker

In integration testing we often face the problem of determining the verdict of a test run without having sufficient insight into the system. This problem is even more apparent in concurrent or embedded software because the traditional approach of using a debugger has limitations. Many developers create log statements to trace the program flow during development. This technology can be extended for use in integration testing.

However due to the large amount of data and the possibly complex behaviour of systems comprehending these traces can very difficult. We propose to use an automated approach to analysing system properties in traces. In this presentation we will motivate, and introduce two approaches to passive testing.