Model-based Testing of Dynamic Component Systems

Siamak Haschemi
first supervisor: Prof. Dr. Joachim Fischer
second supervisor: Prof. Dr. Jens-Peter Redlich
in METRIK since: 2008-02-01

Problem statement

Modern software systems are not created monolithic anymore. Instead, they are built-up by a collection of reusable components and can evolve (due to new requirements and identified bugs) by replacing one or several components. Some of these systems, called *Dynamic Component Systems* (DCSs), cannot be stopped in order to replace components. Instead, components have to be replaced while the system is running. Thus, components need to be prepared for *dynamic availability* of required functionality provided by other components. Examples of such systems are safety- and mission-critical systems like the SOSEWIN system, where a downtime of the system could lead to a high financial or civil risk. The ability of a components to be tolerant to dynamic availability of required functionality is the central problem I want to focus on in my thesis. To keep the complexity manageable, I assume following characteristics of the systems:

- systems are *reactive* and state changes are caused by *discrete events*
- the number of states of the system is *finite*
- there are *no real-time* requirements
- systems support method- or event-based communication between components

The goals of my thesis are:

- to develop of a systematic test approach for testing the ability of a component to handle dynamic availability of required functionality; testing should be possible at development time
- integrating the aspects of the approach in a metamodel-based description with the long-term goal of platform and tool independence

Approach

My approach is to apply the technique *model-based testing* (MBT) to DCSs. MBT, a variant of testing, defines a process (fig.1) to generate automatically test cases from formal models, also called *test models*. These models contain the expected behavior of (parts of) the system under test. The typically infinite number of possible test cases is reduced by using *test selection criteria*. In the phase of *test execution*, the generated test cases are applied to the SUT and expected and actual behavior is compared to find bugs. Applying MBT to DCSs raises following scientific questions, which need to be answered in my thesis:

- **test models:** How can we consider the dynamic availability of components in the creation of test models?
- **test selection criteria:** Which test selection criteria are suitable to find bugs in DCSs related to dynamic availability of required functionality?
- **test case generation:** Which test generation tool supports (or can be extended to support) DCSs, the considered type of test models, and the required test selection criteria?
- **test execution:** What requirements has the test execution on the component runtime environment?

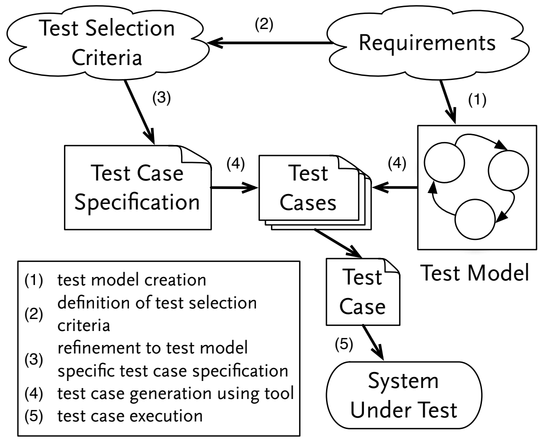

Figure 1. Model-based Testing process.

Current and Future Work

With the characteristics and assumptions made for DCSs, the usage of transition-based notations for test models (i.e. UML Statecharts) seems to be sensible. My approach is to include the component description and the life cycle states of components in the test models to generate test cases containing instructions for the component runtime to activate and deactivate components. This approach has to be formalized and evaluated with the help of case studies.

A novel test selection criteria for parallel Statecharts has been proposed [1]. This and other existing criteria has to be evaluated to show their fault finding properties for bugs related to dynamic availability.

PUBLICATIONS

[1] Haschemi, S. *Model Transformations to Satisfy All-Configurations-Transitions on State-charts*, MoDeVVA 2009 - Model-Driven Engineering, Verification, and Validation: Integrating Verification and Validation in MDE, Denver, Colorado, USA, October 2009.

[2] Haschemi, S. & Wider, A. *An Extensible Language for Service Dependency Management* , Service and Component Based Software Engineering (SCBSE), Patras, Greece, August 2009.

[3] Haschemi, S. & Sadilek, D. *Modeling Component Dependencies*, poster, Fundamentals of Software Engineering (FSEN), Kish, Iran, April 2009.