Efficient Incremental Testing of Variant-Rich Software Systems

Prof. Dr.-Ing. Ina Schaefer
(joint work with Malte Lochau and Sascha Lity)
Institute of Software Engineering and Automotive Informatics, Technische Universität Braunschweig

CREST, London, 19 November 2012
Agenda

• Model-based Testing – Basic Principles

• Efficient Testing of Software Product Lines
  • Combinatorial Testing by Subset Selection
  • Regression-based Incremental Testing
Objectives of Testing

A software test is the dynamic verification of a system with a set of test cases against the expected system behavior in order to

- Observe erroneous behavior during system execution,
- Detect conceptual faults in software,
- Correct errors in implementation.
Model-based Testing - Concept

Test Model

conforms?

\[ \text{i}_k, \ldots, \text{i}_2, \text{i}_1 \]

\[ \text{o}_1, \text{o}_2, \ldots, \text{o}_k \]

IUT
State machines as test models

- $S$ is a finite set of states
- $s_0 \in S$ is an initial state
- $L \subseteq \Pi_I \times \Pi_O$ is a set of transition labels
- $T \subseteq S \times L \times S$ is a transition relation
State machine test artifacts:

- **Test case**: \( tc = (t_0, t_1, \ldots, t_k) \in T^* \) is a sequence of \( k \) transitions
- **Test run**: \( \text{exec}(tc, tm) = (l_0, l_1, \ldots, l_k) \in L^* \)
- **Test result**: \( \text{iut passes} \; \; tc \iff \text{exec}(tc, \text{iut}) \approx_{te} \text{exec}(tc, tm) \)
State Machine Test Suite

- Infinite set of all valid test cases: $TC(tm) \subseteq T^*$
- Derive finite test suites: $ts \subseteq TC(tm)$
- Coverage criteria $C$: finite sets $tg = C(tm)$ of test goals such that $\forall g \in tg : \exists tc \in ts : covers(tc, g)$
Advantages of Model-based Testing

- Systematic and automatic test case generation
- Usable in early development phases (Model-in-the-Loop)
- Automatization of test execution
- Regression planning by change impact analysis on models
We use a combination of two approaches:

**Combinatorial Testing:**

Selection of Representative Subsets from a large set of possible variants

**Incremental Regression-based Testing:**

Reuse Test Cases and Test Results in order to efficiently test the selected variants
Efficient Testing of Feature Interactions

Subset Selection Heuristics (Lochau/Oster et al., Perrouin et al.):

- Start from Set of Valid Feature Combinations (Feature Model)
- Select Representative Set of Product Variants Covering all Pairs of Features (using Set Coverage Algorithms)
Incremental Model-based Testing

Evolution/Variation

Test Model TM → Test Model TM’

Test Goals TG → Test Goals TG’

Test Suite TS → Test Suite TS’

Test Plan TP → Test Plan TP’
Delta-Modeling of Variant-Rich Systems

- Product for valid feature configuration.
- Developed with Standard Techniques

- Modifications of Core Product.
- Application conditions over product features.
- Partial ordering for conflict resolution.
Delta-Modeling - Background

Instances of Delta-Languages:
- Software Architectures (Delta-MontiArc)
- Programming Languages (Delta-Java)
- Modeling Languages (Delta-Simulink, Delta-State Machines)

Advantages of Delta-Modeling
- Modular and flexible description of change
- Intuitively understandable and well-structured
- Traceability of Changes and Extensions
- Support for proactive, reactive and extractive SPLE
Delta-oriented Test Models (Examples)

Adding a State to a State Machine:

Changing the transition labels:
Delta-oriented Test Modeling

Feature-Konfiguration

Feature-Modell

Delta 1 Add

Delta 2 Rem

Delta 3 Mod

Kernprodukt T1: e1/e2

T2: e3/

T3: e2/

T4: e5/e6

T5: e4/e6

Kern + Delta 1

Kern + Delta 1 + Delta 2

Kern + Delta 1 + Delta 2 + Delta 3

Produkt

S1

S2

S3

S4

S5

S6

S7

T1: e1/e2

T2: e3/

T3: e2/

T4: e5/e6

T5: e4/e6

T6: e3/

T7: e1/

T8: e3/

T9: e2/

T10: e5/e6

T11: e4/e6
State Machine Regression Delta

\[
\begin{align*}
\delta_{tm}^{-1} &= \{\text{add } t1, \text{add } t3, \\
&\quad \text{rem } t4, \text{rem } t5\} \\
\delta_{tm} &= \{\text{rem } t1, \text{rem } t3, \\
&\quad \text{add } t4, \text{add } t5\} \\
\delta_{tm',tm'} &= \{\text{add } t3, \text{rem } t4\} \\
\delta_{tm'} &= \{\text{rem } t1, \text{add } t5\}
\end{align*}
\]
Classification of Test Cases by Delta-Analysis

Test cases V1

Retest V2

Test cases V2

Test cases Vn

Variant 1

Variant 2

Variant n

Invalid V2

\[ \Delta \]
Delta Testing - Procedure

0. Fully test first product variant

1. Generate test cases for subsequent variants
   • Still valid and reuseable test cases?
   • Invalid test cases?
   • New test cases?

2. Selection of test cases by delta analysis:
   • Always test new test cases
   • Select subset of reuseable test cases for re-test

3. Optionally minimize resulting test suite by redundancy elimination
Delta-Testing Strategy
Case Study – Body Comfort System

28 Features, 11161 Product Variants, 1 Core Product, 40 Deltas
16 Products for Pair-Wise Feature Coverage
Case Study BCM – Delta-Testing Results
Case Study BCM – Delta-Testing Results (2)
Requirements-based Delta-Testing - Classification

Pilot Study in Automotive Domain (Relative Figures)
Requirements-based Delta-Testing – Estimated Test Effort

Pilot Study in Automotive Domain (Relative Figures)
Possible Strategies for Re-Test Selection

• Manually by Test Engineer

• (Semi-)Automatical Classification of Test Cases into Variants

• Formulation of Requirements in Delta-Sets with Linking of Test Cases to Requirements

• Model-based Impact Analysis of Changes by Delta Analysis
Conclusion

• **Model-based testing** allows systematic test case generation and automatic test case execution.

• **Subset selection heuristics** reduce number of product variants to a representative subset which is tested.

• **Incremental delta-testing** reduces test effort between product variants by test case and test result reuse.