Combinatorial Test Design in the TOSCA Testsuite

Lessons Learned and Practical Implications

Rudolf Ramler, Theodorich Kopetzky
Software Competence Center Hagenberg GmbH
Softwarepark 21, A-4232 Hagenberg, Austria
{rudolf.ramler, theodorich.kopetzky}@scch.at

Wolfgang Platz
TRICENTIS Technology & Consulting GmbH
Leonard-Bernstein-Straße 10, A-1220 Vienna, Austria
w.platz@tricentis.com
Overview

- Basics – the mash
- What are the Requirements?
  - The distillate
  - Grouped by: containing the combinatorial explosion, risk and business value, understandability, changeability and maintainability, fault localization
- What is TOSCA?
  - The software
  - Test generation strategies
    - Combination
    - Linear Expansion
- Conclusion
  - How do the generation strategies compare?
Company **TRICENTIS**
- Company offering tools and services in the SQA market
- ca. 200 employees in Austria, subsidiaries in Europe, Australia, and the US
- 15 years of experience in testing large SW applications
  - Financial domain (such as banking and insurance), healthcare, public sector and industry
- Customers all over the world
- Projects with # of test cases ranging from ca. 250 to 6000

Product TOSCA
- Includes wide range of features that support
  - Test automation, test case management,
  - Test data management as well as
  - Test case design

**SCCH**
- Software Competence Center Hagenberg GmbH
- Common project on concept analysis
REQ-1: High coverage of input parameters, data values and combinations
- Core requirement of combinatorial techniques
- Cost of full coverage nearly always prohibitive

REQ-2: Minimal number of combinations to test
- Linear increase of test case # would be ideal
  - Test effort easy to estimate -> basis for test planning and management

REQ-3: Dealing with dependencies
- Dependencies add complexity
- Need ways to cope with dependencies
REQ-4: Support for the value-based paradigm
- Requirements have different business values and different risks
- So have the associated test cases
- Standard test scenario accounts for 40-60% of overall value of requirement acc. to TRICENTIS

REQ-5: Estimation of contribution of a test
- Needed for test case prioritization and test case selection

REQ-6: Robustness of the test suite for a selective execution of tests
REQ: Understandability

- REQ-7: Ease of understandability of a technique for test design
  - Necessary for adequate and correct application of technique by tester
  - Influences quality of tests
- REQ-8: Understandability of produced output
  - Influences refinement of test activity
  - Transition to automated test scripts
  - etc.
REQ: Changeability and Maintainability

- REQ-9: Ease of extending a set of tests with new input parameters or new data values
  - Identify missing combinations
  - In practice changes to existing tests tend to be avoided
    - Costly post processing
    - Makes comparing different test runs more difficult

- REQ-10: Ease of reducing tests
  - Input parameter or data values become obsolete
**Support for fault localization**

- **REQ-11: Side effects of a failing test should be minimal**
  - Mass failures (not only in huge test suites) hinder analysis
    - e.g. as part of a daily test execution
    - Number of passed/failed test is often used as indicator for progress progress and health
      - Gives wrong picture

- **REQ-12: Identifying the root cause of a failure**
  - Failures not only due to combination of input values but as well due to
    - Script problems
    - Interface incompatibilities between test system and SUT
    - Data conversion problems
    - Infrastructure changes
    - Environmental problems (e.g. out of disk space for log)
    - etc.
A Very Short Introduction to TOSCA

- Comprehensive suite for test management and test automation
- Test process in TOSCA
  - Build risk-based structure of requirements
  - Design test cases
  - Specify/automate designed test cases
  - Execute test cases
  - Project execution results on defined requirements
- Handling of input parameters
  - Derived from business logic, equivalence partitions, boundary values
  - Character of values can be specified
  - Can be hierarchically structured (like classes)
Combination Strategies Supported by TOSCA

- Full combination
  - Every possible combination of instances of business-relevant attributes
  - Combinatorial explosion
  - Only for small sets of input parameter
- Pair-wise combination
  - Each pair of instances of business-relevant attributes at least once
- Orthogonal combination and
- Linear expansion
Short Example

- Test cases for calculating the annual premium of a vehicle insurance policy (liability insurance).
- Greatly simplified specification from the business perspective
  - Owners younger than 18 years of age are not insured.
  - Owners between 18 and 23 pay an additional 20% (beginners’ license).
  - Owners 60 and older receive a 10% discount (seniors’ discount).
  - Women receive a 5% women’s discount.
  - Owners living in the city pay an additional 15% (higher risk of accidents).
Orthogonal Combination

- Each instance of each business-relevant attribute at least once
- May lead to an acceptable functional coverage as long as attributes are independent

### Example

- **Named Insurant**: Age
  - TC 1: < 18
  - TC 2: 18 - 23
  - TC 3: 24 - 59
  - TC 4: > 59

- **Sex**: m, f
  - TC 1
  - TC 2
  - TC 3: m
  - TC 4: m

- **Residence**: city, country
  - TC 1: city
  - TC 2: country
  - TC 3: city
  - TC 4: city
Test case with most important equivalence classes for all attributes => “straight through”

Minimum requirement of testing
- Straight through runs all the way through without error
- All straight throughs can form a smoke test portfolio
  - Can be executed with the highest priority

Each test case differs from the straight through in exactly one test relevant feature.
## Conclusion: Strength/Weaknesses of CT wrt. Practical Requirements

<table>
<thead>
<tr>
<th></th>
<th>Full combination</th>
<th>Orthogonal combination</th>
<th>Pair-wise</th>
<th>Linear Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Combinations and dependencies</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ-1</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>~</td>
</tr>
<tr>
<td>REQ-2</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>REQ-3</td>
<td>+</td>
<td>–</td>
<td>~</td>
<td>–</td>
</tr>
<tr>
<td><strong>B. Value- and risk-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ-4</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>REQ-5</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>REQ-6</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td><strong>C. Understandability</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ-7</td>
<td>+</td>
<td>+</td>
<td>~</td>
<td>+</td>
</tr>
<tr>
<td>REQ-8</td>
<td>+</td>
<td>~</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td><strong>D. Changeability</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ-9</td>
<td>+</td>
<td>+</td>
<td>~</td>
<td>+</td>
</tr>
<tr>
<td>REQ-10</td>
<td>–</td>
<td>~</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td><strong>E. Fault localization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ-11</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>~</td>
</tr>
<tr>
<td>REQ-12</td>
<td>+</td>
<td>–</td>
<td>~</td>
<td>+</td>
</tr>
<tr>
<td><strong>Score (+/-)</strong></td>
<td>8/4</td>
<td>6/4</td>
<td>2/6</td>
<td>9/1</td>
</tr>
</tbody>
</table>
Thank you