EMBRACING NON-DETERMINISM IN TESTING

Andreas Ulrich, Stefan Dorsch
Non-determinism in testing – A Phantom Menace?

Martin Fowler (“loud-mouthed pundit on software development”) on Eradicating Non-Determinism in Tests
“Non-deterministic tests have two problems, firstly they are useless, secondly they are a virulent infection that can completely ruin your entire test suite.”
http://martinfowler.com/articles/nonDeterminism.html

Jason Polites: The Tao of Testing v1.1 – A Field Manual for SW Engineers
Rule #2 Unit tests should be deterministic
“If a test relies on non-deterministic components in order to succeed then failures in these components will cause a failure in the test.”
http://jasonpolites.github.io/tao-of-testing/ch4-1.1.html

Vojta Jina (Node.JS developer) on Make Your Tests Deterministic
“In order to guarantee correct behavior of our code, we need to be sure that it handles correctly all […] situations. The best way to do that is by simulating these situations in a fully controlled – a deterministic way.”
https://howtonode.org/make-your-tests-deterministic
Non-determinism in testing – The light side

- **Non-determinism in testing is there already** – mostly unknowingly!
  - Test selection strategies follow their own rules, hidden in tools
  - Examples
    - Different tools on pair-wise test generation produce different output
    - In fact, all MBT tools produce different test cases for the same SUT

- **Lifting non-determinism at test execution level**
  - Why? – Improving test coverage!
  - Example: exploratory testing
  - How to avoid its negative side effects?
    - Repeatability, fault analysis
The quest for higher test coverage – The typical practice

I have some running tests...

Test 1
Test 2
Test 3

Let’s combine them to have more complex tests...

Complex test

Test 2
Test 1
Test 3

Let’s run these tests for a long time...

for (int i = 0; i < LargeNum; i++)
{
    Test 2
    Test 1
    Test 3
}
Limits of this practice

I have some running tests…

Test 1

Test 2

Test 3

Let’s combine them to have more complex tests…

Let’s run these tests for a long time…

Simple repetition of (deterministic) tests does not improve coverage

Test 1

Test 2

Test 3

Complex test

I have some running tests...

Test 2

Test 1

Test 3

for (int i = 0; i < LargeNum; i++)
{
  Test 2
  Test 1
  Test 3
}
A proven strategy for non-deterministic testing

For testing software systems, black-box testing at the SUT façade (API testing)

- **Deterministic tests**
  - Test design and test execution are deterministic
  - Simple and workflow-driven tests focusing on SUT interactions
  - Combinatorial tests focusing on SUT state verification

- **Non-deterministic tests**
  - Test design contains arbitrary non-deterministic choices, but deterministic test execution
  - Descriptive test design with choices leads to non-deterministic test execution
  - Long-running tests focusing on SUT durability and performance
Step-wise test strategy visualized

Simple tests
- Test 1
- Test 2
- Test 3
- Test 4
- Test 5
- Test 6
- Test 7
- Test 8

Workflow & combinatorial tests
- Test 1
- Test 2
- Test 3
- Test 4
- Test 5
- Test 6
- Test 7
- Test 8

Long-running tests
- Test 1
- Test 2
- Test 3
- Test 4
- Test 5
- Test 6
- Test 7
- Test 8

Choice of behavior
- Deterministic tests
- Non-deterministic tests

Choice of data
Prerequisites for non-deterministic test execution

- **Modeling of SUT façade behavior**
  - Provide an abstract model as basis for test automation
- **Test automation layer for accessing SUT façade**
  - Build abstraction of concrete SUT behavior
  - Implement assertions for SUT responses here
- **Test runner tool for on-the-fly test execution supporting**
  - Non-deterministic choice of behavior (ND-Beh)
  - Non-deterministic choice of data (ND-Data)
### Available tools for non-deterministic testing

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Technology</th>
<th>Notation</th>
<th>ND-Beh</th>
<th>ND-Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>fMBT</td>
<td>OSS</td>
<td>Python</td>
<td>ASM</td>
<td>✔</td>
<td>—</td>
</tr>
<tr>
<td>GraphWalker</td>
<td>OSS</td>
<td>Java</td>
<td>EFSM</td>
<td>✔</td>
<td>—</td>
</tr>
<tr>
<td>JTorX</td>
<td>OSS</td>
<td>Java</td>
<td>IOTS</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Modbat</td>
<td>OSS</td>
<td>Scala</td>
<td>EFSM</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>ModelJUnit</td>
<td>OSS</td>
<td>Java</td>
<td>FSM</td>
<td>✔</td>
<td>—</td>
</tr>
<tr>
<td>OSMO</td>
<td>OSS</td>
<td>Java</td>
<td>ASM</td>
<td>✔</td>
<td>—</td>
</tr>
<tr>
<td>PyModel</td>
<td>OSS</td>
<td>Python</td>
<td>ASM/FSM</td>
<td>✔</td>
<td>—</td>
</tr>
<tr>
<td>TestOptimal</td>
<td>Commercial</td>
<td>Java</td>
<td>EFSM</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Legend:**
- ASM… Abstract State Machine
- (E)FSM… (Extended) Finite State Machine
- IOTS… Input/Output Transition System
- ND… Non-deterministic choice
- OSS… Open Source Software

**Disclaimer:** List created on best knowledge, not exhaustive.

Selection based on Zoltán Micskei’s collection of MBT tools, [http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html](http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html)
Putting non-deterministic testing into practice

Siemens TIA Portal


The engineering platform for the Totally Integrated Automation of complex production processes
Obtained results and achievements

Use of non-deterministic testing during integration test phase of TIA Portal development

- Non-deterministic tests introduced as last test step, i.e. nothing to start with!
  - Using our in-house test tool based on C#
  - Ensure that all known test scenarios are covered in deterministic test executions before testing based on non-deterministic choice of behavior and data proved very successful!
  - Creates test scenarios not tested before, because they are hard/impossible to anticipate
  - Realistic faults found that would slip through testing otherwise
  - Tests are suitable in regression testing as part of Continuous Integration

- Managing repeatability of test results
  - Ensure that non-deterministic choices are made from a random number generator using a seed value
  - Non-determinism inside the SUT cannot be addressed by any approach

- Fault analysis from long test execution runs
  - Can be alleviated with proper test logging and Design for Testability measures
Contact details

Andreas Ulrich
Siemens AG, Corporate Technology
München / Germany
E-mail: andreas.ulrich@siemens.com

Stefan Dorsch
Siemens AG, Digital Factory
Amberg / Germany
E-mail: stefan.dorsch@siemens.com