From Manual Testing to Intelligent Test Automation
Stephan Schulz & Alexis Desperouxx
About the Speaker

• Academic work in model-based codesign
• Start of industrial career in testing at Nokia Research Center in 2001 with TTCN-3
• 3.5 years resident testing expert at ETSI
• 10 years at Conformiq mainly in role of CTO and PO for Enterprise IT products
• Since Oct 2019 test architect at G+D
• Total of 18 years of experience in working with model-based testing and test automation
From Manual Testing to Intelligent Test Automation

- Where Are We Today in Enterprise IT Testing?
- Evolution of Software Testing
  - From Manual Testing to Automated Test Design
- Evolution of Model Based Testing
  - 3 Generations of Test Automation with Model-Based Testing
- Artificial Intelligence in Testing
  - Intelligent Test Automation
  - The Future of Testing
Where Are We Today in Enterprise IT Testing?
How Much of Your Test Execution is Automated?

<table>
<thead>
<tr>
<th>Automation Level</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 80%</td>
<td>?</td>
</tr>
<tr>
<td>Between 41% and 80%</td>
<td>?</td>
</tr>
<tr>
<td>Between 20% and 40%</td>
<td>?</td>
</tr>
<tr>
<td>Less than 20%</td>
<td>?</td>
</tr>
<tr>
<td>We do not automate</td>
<td>?</td>
</tr>
</tbody>
</table>
Most of Today’s Testing is Still Done Manually

How much of your test execution is automated?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 80%</td>
<td>10%</td>
</tr>
<tr>
<td>Between 41% and 80%</td>
<td>12%</td>
</tr>
<tr>
<td>Between 20% and 40%</td>
<td>15%</td>
</tr>
<tr>
<td><strong>Less than 20%</strong></td>
<td><strong>56%</strong></td>
</tr>
<tr>
<td>We do not automate</td>
<td>7%</td>
</tr>
</tbody>
</table>

Poll results at a Enterprise IT Testing Workshop in Europe
Even More Dramatic: Enterprise IT Testing Market 2017

Automated Testing: $30BN
  ... of which only tools: $3BN
  ... of which only test execution: 60%

Manual Testing: $120BN (!)

Total Size: $150BN (14% CAGR)
Product Development is Moving Faster Every Day
Executive Management Objectives: Shift Left, Higher Quality

Source: 10th Edition of World Quality Report 2018-19
Example Quotes from the Field

“We have 3 testers per developer and defects still slip through.”
Test Lead Healthcare IT

“I have 30,000 test cases and my testers can not tell me what they cover.” Huawei Test Director

“I was surprised about the throughput of one of my testers – until I decided to review his test scripts ...” Enterprise IT Test Manager
The Net Is Testing Needs To Be Done ..

FASTER

BETTER

CHEAPER
The Evolution of Software Testing
Evolution of Software Testing

- Manual Testing
- Test Scripts Capture/Replay
- Test Models
- Keyword Driven Testing Frameworks
- Automated Test Design (ATD)
- Model Based Testing (MBT)
Manual Testing

• “It is cheap”! Anybody can do it!
• It is time consuming, ineffective, and error prone
• It is still not (yet) replaceable
  • When tests are hard (or too expensive) to automate
  • Exploratory testing (with data)
Test Scripts and Capture Replay

- Automatic execution!
- Interactions have to be manually performed or coded
- Scripting requires special skills
- But capture/replay is still driven by manual testing 😊
- One small change in SUT requires re-recording or re-coding
(Keyword Driven) Testing Frameworks

- Simpler to use, understand & maintain
- Allows reduction of required skills for test specification down to Excel!
- Keywords are reusable & tests become retargetable!
- Still requires (highly) skilled automation team for scripting keywords

Example:
```python
Login(username, password)
StartVehicle()
```
Test Models (MBT)

- Generally graphical representation of control flows instead of coding
- Easier to adopt & review than scripts
- Can be generated from manual test descriptions or test plans
- Allows generation of (partial) scripts and documentation
- Still user is responsible for test design
Automated Test Design (MBT)

- Specify functionality to be tested and generate optimal test coverage
- Generally graphical representation of control and data aspects
- Basically full script & documentation generation from generated tests
- Now manual testers can effectively contribute to test automation!

Don’t model tests; model the functionality to be tested!

When you model the functionality to be tested you can create a model capable of representing every test for that functionality. It is a much smaller task, since the number of all possible tests is very large and change behind your back, which means test models will become obsolete the moment they are made.

(adopted from Monica Anderson “Don’t model the World; Model the Mind”)
About Model-Based Testing (MBT)

• Umbrella term for using models in a testing context; only one approach is to use MBT for *automating test design*
  • MBT *complements* test execution
  • Recognized by worldwide industrial standards (ETSI, ISTQB)
The Evolution of Model-Based Testing
Evolution of Model Based Testing

- Standards
- Test Design
- Modeling
- Test Process

- People
  - ISTQB
  - Model Generation asset reuse

- Technology
  - SDLC Tool Integrations
    - modeling, ALM, test execution, CI
  - Automated Test Design
    - combined algorithms
    - symbolic state space exploration
    - constraint solving

- Algorithms
  - graph traversal, pairwise

- Programming
  - code

- General Purpose Languages
  - graphical + code

- Domain Specific Languages
  - fully graphical

1970 MBT 1.0

MBT 2.0

MBT 3.0

User Conference on Advanced Automated Testing
Automated Test Design: Planning & Searching

Constructs tests from the explored part of the state space by selecting paths that leads to testing goals.

Test generation algorithm searches the state space for testing goals, defined by the end user.

Model behavior implies typically infinite state space.
Automated Test Design: Reasoning & Problem Solving

- Test generation reasons about unspecified input data and derives a concrete data based on modeled logic.
- Generally, constraint solving theory is applied for computing unspecified input data.

Models interact with an unspecified environment via inputs and outputs.

Models describe the expected external behavior of the system under test.
Automated Test Design: Finding Important Tests

• Predict ways the system may fail
• Combine large set of heuristics
  • Equivalence classes
  • Boundary value analysis
  • Combinatorial testing
  • Mutation testing
  • And more...
• Know what do not cover!
MBT 1.0: Modeling Notation is Adoption Hurdle

From functional programming (10+ years ago) ...

... via some graphical & a lot of general purpose programming (~7 yrs ago) ...

```java
class GarageDoorController extends StateMachine {
    /** The default constructor. */
    public GarageDoorController() {
    }

    public void startMotor(String direction) {
        MotorStart start;
        start.direction = direction;
        out.send(start);
    }

    public void stopMotor() {
        MotorStop stop;
        out.send(stop);
    }

    public void reverseMotor() {
        MotorReverse reverse;
        out.send(reverse);
    }
}
```

```csharp
namespace Example1 {
    static class AccumulatorModelProgram {
        static int accumulator;

        [RuleAction = "Add(s)!
        static void AddRule(int a) {
            Condition.IsTrue(x => 3); accumulator += x;
        }

        [RuleAction = "ReadModIf
        static int ReadAndRelease(int value) {
            Condition.IsTrue(accuarator => return oldValue);
        }
    }
}
```

```lisp
(define quotient
  (lambda (a b)
    (let ((c (* a b)))
      (if (> c 0)
          ((floor c) (ceiling c))))))

(define remainder
  (lambda (a b)
    (- a (* (quotient a b) b))))
```

```ocaml
class GarageDoorController extends StateMachine;

    /** The default constructor. */
    public GarageDoorController() { }

    public void startMotor(String direction) { 
        MotorStart start;
        start.direction = direction;
        out.send(start);
    }

    public void stopMotor() {
        MotorStop stop;
        out.send(stop);
    }

    public void reverseMotor() {
        MotorReverse reverse;
        out.send(reverse);
    }
```

```java
public void startMotor(String direction) {
    MotorStart start;
    start.direction = direction;
    out.send(start);
}
```

```java
public void stopMotor() {
    MotorStop stop;
    out.send(stop);
}
```

```java
public void reverseMotor() {
    MotorReverse reverse;
    out.send(reverse);
}
```
Today We Use Domain Specific Modeling Languages

- Designed specifically to use tester’s language, concepts & terminology
- Easy to use: drag & drop building blocks to compose functionality
- Enabling direct automation
- Flexible and extensible
- See also UCAAT 2013 tutorial
MBT 2.0: Enabling Test Process Automation

- MBT is not just about test generation!
- MBT integrates & connects with requirement & test management, versioning control, and test automation/CI integration
But Still Every New Project Started Like This ...
MBT 3.0: “Eliminating” Modeling

Complete End to End Test Process Automation with MBT
[ALM] Test Plans: Excel as Input to MBT

Import & Refine

Generate Tests

Execute!

Test suite optimization during import:
- Reduction in Tests: 0%
- Reduction in Input Actions/Test Steps: 43%
- Reduction in Verification Actions/Expected Results: 29%
- Reduction in Total Generated Keywords: 38%

User Conference on Advanced Automated Testing
**Feature**: Shopping with esd.conformiq.com

**Scenario**: Successful shopping

Given a user has logged in successfully
When user adds 1 item(s) of "CQ0003"
Then shopping basket contains 1 item(s)

**Scenario**: Bad product id

Given a user has logged in successfully
When user adds 3 item(s) of "123"
Then application displays invalid entry

**Scenario**: Remove item

Given a user has logged in successfully
When user adds 1 item(s) of “CQ0003”
Then shopping basket contains 1 item(s)
When user clicks Remove all button for item "CQ0003"
Then shopping basket contains 0 item(s)
Test Reverse Engineering

- Readers normalize test input format
- Keywords are identified from test steps using Natural Language Processing (NLP)
- Redundant tests and subtests are eliminated
- Tests are converted into flows and merged
MBT Trends in 2018: Modeling in Prose

- Low end MBT tools are pushing very simple, prose based modeling notations into the test tool market
  - Arguably domain specific modeling still has the edge as it enables a direct path to automatic test execution (today not possible from prose)
- Users have started to perceive domain specific modeling (= creation of building blocks) as a burden when it comes to modeling
  - Model generation from test assets (reverse engineering) has addressed this issue but not removed it
- Frequent use of extension points ("custom actions") is diluting the value proposition of domain specific languages
  - Prose and domain specific action modeling start to mix
Artificial Intelligence in Testing
AI Started Disrupting Entire Software Market in 2016 (!)

- Multiple application domains
  - Home: Siri, Alexa, etc
  - Assistance systems: Tesla, etc
  - Robotic Process Automation (RPA)
- Emergence of (many) strong, open source AI platforms
  - Tensorflow, Watson, CNTK, etc
- Already one year later AI was in news headlines every week
AI Techniques Have Been Leveraged for (Many) Years!

- Fuzzy and Probabilistic Logic ✓
- Classification ✓
- Search & Optimization ✓
- Expert Systems ✓
- (Deep) Machine Learning (ML) ？

Remember this?

Today the term “AI” is generally reduced to (Deep) Machine Learning ...
More “Recent”: AI in Robustness Testing

- App store testing example: Fuzz vs search vs man+machine
- Initially, “break the system by clicking around” (= “Monkey Testing”)
- Most successful tools combining human guidance and learnings from previous failures
- No human required for test execution!

Source: Stuart Reid, “Smarter Testing with Artificial Intelligence”, JFTL 2017
More Recent: AI in Result Analysis of Large Test Suites

- Predictive analytics: what *could* fail?
  - “Test Less - Test Right” (M. Venkata)
  - Failure patterns, code changes, etc

- Diagnostic analytics: what is failing?
  - Data mining & visualization of defects
  - Application monitoring for anomalies

- Prescriptive analytics: what to fix?
  - Symptom vs defect
  - Root cause analysis
How Could We Use Learning in Functional Testing?

- Test automation tools have managed successfully to make creation of tests that are automatically executable more or less “dead easy”
- **BUT ...** automating from “arbitrary” text still has to be done manually

Enter user name and password and click the Sign In button

Manually refine into domain specific model inputs & outputs

Manual Test Step

Manually refine into concrete automation steps

Test Automation Keyword

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Object</th>
<th>LocatorType</th>
<th>Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>username</td>
<td>id</td>
<td>type</td>
<td>User nameS</td>
</tr>
<tr>
<td>password</td>
<td>password</td>
<td>id</td>
<td>type</td>
<td>PasswordS</td>
</tr>
<tr>
<td>signin</td>
<td>signin</td>
<td>id</td>
<td>type</td>
<td>clickAndExit</td>
</tr>
</tbody>
</table>
Introducing Intelligent Test Automation

- From manual to automatically executable tests in two steps
  1. Feed manual tests and train intelligent test automation (as required), e.g., resolve unknown text to automation rules
  2. Execute and fix unrecognized object identifiers (if any)
- The more intelligent test automation is trained - the less it needs to learn!
First Generation Intelligent Test Automation in a Nutshell

- Reader(s) normalize input format(s)
- Natural Language Processing is used to analyze test steps
- User “teaches” mappings to target test automation tool or framework
- Works with any input format, different target technologies, and any test automation framework!

Note: Conformiq patents pending
Intelligent Test Automation Naturally Extends to MBT!

Requirements

- Includes BAs & Testers

Test Design

- Includes optimization

Test Creation

- Includes scripts, data & validations

Test Execution

- Includes harnesses

Test Files

(fully generated)

Custom Libraries

Exec Tool(s)
The Start of a New Generation in Test Automation?
A New Breed of Test Automation Tools has emerged ...

- Target is Enterprise IT
- Machine Learning based
- Cloud based
- Mainly web app testing (Selenium)
It Could Just be the Beginning ...
Thank You for Your Attention!

Q&A