Deploying MBT-based Test Automation in an Agile Development Project for Financial Industry
UCAAT 2014

Munich
17th September
Simone Krämer, Brightone GmbH
Jani Koivulainen, Conformiq
Project Context

- 18-month development project for a major financial institute in the US
- Team of 50: business/developers/testers
- Developed an ecosystem of backend and UI components
  - Java (Spring Framework)
  - RESTful web services between components
- MBT-activities focused on the ”core node”
  - UI
  - Backend serving UI but also other components in the UI
- Agile
  - 2-week sprints
  - 4-week ”development drops” with 2 sprints each
  - 15 development drops followed by final System Integration + Acceptance Test phases
- Test execution
  - QTP for UI
  - JUnit for backend
System Description

• The system consists of
  – UI
  – Backend

• UI runs on a web browser

• UI and Backend are connected over a RESTful interface (JSON payload)

• Backend serves not only its own UI but also other components (MBSC for example) over
  – REST (JSON/XML)
  – ESB (XML)

• The system persists its own data in an internal database
System Model

- Two *model components* for
  - UI
  - Backend

- Model components are connected over an interface just like the real components

- Just like the real backend, the *modeled* backend exposes external interfaces for communication towards other components

- There is also an *in-model database* to mirror the database of the real system

- Reusable components at the model level

1. **Modeled** UI i/f
2. **Modeled** RESTful i/f
3. **Modeled** RESTful i/f
4. **Modeled** ESB i/f

**UI Model**

**Backend Model**

**Modeled database**
Testing the Backend

- Tests generated from the Backend Model exercise the backend by:
  - RESTful calls
  - Interaction over ESB queues/topics
  - Verification of db contents
- Test are rendered in Java as JUnit tests
- Reusable components at test harness level
Testing the Full System

• Tests generated from full System Model exercise the system by
  – “simulated” user interaction through the UI
  – RESTful calls
  – Interaction over ESB
  – Verification of db contents

• Tests are rendered in VBScript for QTP
Reusability

• The tools used in implementing the automation are readily available technologies in active development.

• Components in the test execution layer are reusable across systems using similar techniques (RESTful web services, ESB)

• Modeling is based on reusability/compositionality
  – Backend model used to generate “backend only tests”
  – The same model used as a component of the “full system model” to generate tests for the “full system”
Findings

The main benefits and takeaways:

• With MBT we were able to maintain automated in-sprint progression testing through the project. Teams *not* doing this could *not* maintain automated progression testing.

• Testing coverage was better than that achieved by manual test design methods.

• We uncovered a lot of issues that would have gone completely unnoticed without the use of MBT:
  – Complex scenarios related to “timing” in particular
  – Interface level issues risking the reliability of the system as a service (towards *other* components)
Q&A

For further information and discussion visit us at the booths from brightONE or Conformiq

Visit our webpages: www.conformiq.com & www.brightone.de