Experience Report: Visual Test Design for Test Automation in Agile of a Large-Scale IT Systems

Presented by Elodie Bernard
Statement
Problem and workflows

Paris, 16-18 October 2018
Problem statement

• Complexity in use of the current MBT approaches
• Introduction of a Lightweight MBT: a visual test design approach
• Simplify the modeling notation
• Facilitate the maintenance of test cases during and through sprints
Workflows statement

Typical MBT modeling approach

Behavioral modeling

Visual test design
Modeling concepts
Example of visual test design with Yest® (from Smartesting)

- Limited number of modeling artifacts
- High modelling capability
- Ability to represent simple as well as complex business processes
Acceptance Test Driven Development
With a visual test design approach
**ATDD concept with a visual test design approach**

- **Start of a new sprint**
  - We have new:
    - User stories
    - Business rules
    - Acceptance criteria

- **We update** the graphical representation to be in line with business rules

- **Features are developed and made accessible on a test environment**

- **We generate** acceptance scenarios

- **Changes in product backlog occur in the sprint and will be developed**

- **We test** the new features

- **End of the sprint**
Lessons learned from using a visual test design approach

• Helps to easily update the test assets
• Allows to quickly generate tests that required an update
• Improves communication and work between project stakeholders
Test automation
Overview
Test automation overview

- Keyword-driven-testing
- Java Selenium add-on
- Data set management

Keywords table with Yest

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class</td>
<td>Keyword</td>
<td>param1</td>
<td>param2</td>
</tr>
<tr>
<td>2</td>
<td>com.test.Automation</td>
<td>Login</td>
<td>id</td>
<td>password</td>
</tr>
<tr>
<td>3</td>
<td>com.test.Automation</td>
<td>CheckData</td>
<td>correctOrNot</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>com.test.Automation</td>
<td>OpenThePage</td>
<td>page</td>
<td></td>
</tr>
</tbody>
</table>
Test automation process

The visual representation of the test

<table>
<thead>
<tr>
<th>Test steps</th>
<th>Actions</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connect to the application with a correct identifier and a correct password</td>
<td>The identifier and the password are provided</td>
</tr>
<tr>
<td>2</td>
<td>Check the data</td>
<td>The data are correct</td>
</tr>
<tr>
<td>3</td>
<td>Access to the home page</td>
<td>Access validated</td>
</tr>
</tbody>
</table>

The abstract scenario and the corresponding automated test script

```java
public class test_log_in.ok {
    private final String id;
    private final String password;

    public test_log_in.ok(final String id, final String password) {
        this.id = id;
        this.password = password;
    }

    @RunWith(Parameterized.class)
    @Public class test_log_in.ok {
        public class test_log_in.ok {
            public Test test_log_in.ok {
                public void execute() {
                    Login(id = Pueblo516r, password = pIpyds-4515)
                    CheckData (correctOrNot = true)
                    OpenThePage (page = homePage)
                }
            }
        }
    }
}
```

The test script in java/Selenium with the use of dataset
Test automation process

Data set collection

The scenario to automate

The scripting details

Create a data set collection

Link the keywords to the test actions

Link data sets to the future automated script
Lessons learned about the automation process

• Our experiences have shown that:
  • Having a visual link between the manual and automated test assets is beneficial
  • Documentation of automated test cases is directly accessible to all project stakeholders through a visual approach
  • The ability to transcribe test cases easily via the keyword-driven system provides visibility and improves maintenance management
Conclusion and futur works
What is new in our approach?

- The approach dramatically simplified the MBT approach
- Maintain a short learning curve and good usability by functional testers
- Be in line with iterative and incremental development approaches
- Supporting both scenario-based and automated test
Futur works

• To experiment the training of functional testers
• Continue to develop an add on in Yest
• To define good practice to facilitate and improve MBT approach, visual test design
• Apply new approaches and methodologies to a group of IT projects