Model-Based Testing @ Spotify

Kristian Karl
How we use Model-Based Testing for test automation at Spotify

In the world of agile development cycle, it's ever more challenging for QA to cope with both new feature testing and regression testing. At Spotify, we try to automate as much of the regression tests as possible. But test automation in itself is a challenging and daunting task. To say the least.

• The separation of test design from test code implementation using the concept of Model-Based Testing simplifies the QA process in an agile team.

• Other important factors:
  - Use developers as coders when automating.
  - Random walks through the state charts.
  - Dynamic and random test data, or test data on-demand.
  - Continuously test execution.
  - Very simple dashboards.

• How manual exploratory testing approaches can benefit from Model-Based test automation.

• What tools to use? There are some really good Open Source/Free tools out there.
@ Spotify we use the term QA, meaning **Quality Assistance**

Dear Programmer, my job is to help you look good. My job is to support you as you create quality; to ease that burden instead of adding to it. In that spirit, I make the following commitments to you:

- I provide a service. You are an important client of that service. I am not satisfied unless you are satisfied.
- I am not the gatekeeper of quality. I don’t “own” quality. Shipping a good product is a goal shared by all of us.
- I will test your code as soon as I can after you deliver it to me. I know that you need my test results quickly (especially for fixes and new features).
- I will strive to test in a way that allows you to be fully productive. I will not be a bottleneck.
- I’ll make every reasonable effort to test, even if I have only partial information about the product.
- I will learn the product quickly, and make use of that knowledge to test more cleverly.
- I will test important things first, and try to find important problems. (I will also report things you might consider unimportant, just in case they turn out to be important after all, but I will spend less time on those.)
- I will strive to test in the interests of everyone whose opinions matter, including you, so that you can make better decisions about the product.
- I will write clear, concise, thoughtful, and respectful problem reports. (I may make suggestions about design, but I will never presume to be the designer.)
- I will let you know how I’m testing, and invite your comments. And I will confer with you about little things you can do to make the product much easier to test.
- I invite your special requests, such as if you need me to spot check something for you, help you document something, or run a special kind of test.
- I will not carelessly waste your time. Or if I do, I will learn from that mistake.

Sincerely,
Tester


Why test automation is so important in the agile dev. cycle

- If QA does too much feature testing, we take the risk of not regression test already released features.
- If QA does too much regression testing, we may release untested and buggy new features.
- And time is precious!

What if:
- The team could deliver automated regression tests, as a criteria of ‘definition of done’?
- Automated tests executed all of the time?

More time for feature testing!
What do we test automate?

• Desktop
  - Windows
    - XP
    - Vista, x84, amd64
    - 7, x84, amd64
  - Mac OSX
    - 10.5, 10.6, 10.7
What do we test automate?

• Mobile
  - iOS
    - iPad, iPhone, iPod
  - Android
    - a lot of devices...
Model-Based Testing

Separation of design and code

• MBT as DSL - Domain Specific Language
• QA uses MBT to design what to test
• Coders uses MBT as a spec and driver for the implementation
Model-Based Testing

• State Diagrams
  *Describes the expected behaviour of the system under test, from the QA perspective.*

• Classifications Trees
  *Describes interesting inputs, ‘classifies’ them, and generates test cases.*
State Diagrams

- Used when the QA wants to describe:
  - flows of events
  - user stories
- Generates online (and offline) test sequences (walks) using different strategies:
  - random, A* or a combination of both
  - different stop conditions
public interface SimpleLogin {
    public void e_Close();
    public void e_Exit();
    public void e_Init();
    public void e_InvalidCredentials();
    public void e_Logout();
    public void e_StartClient();
    public void e_ToggleRememberMe();
    public void e_ValidPremiumCredentials();
    public void v_ClientNotRunning();
    public void v_LoginPrompted();
    public void v WhatsNew();
}
The tools when using State Charts

**yEd**
- Graph editor
- License: Free

**GraphWalker**
- Generating offline and online test sequences
- License: Open source
- [http://graphwalker.org/](http://graphwalker.org/)

**Sikuli**
- Visual technology to automate and test GUI using images
- License: Open source
- [http://sikuli.org/](http://sikuli.org/)

**Java, TestNG and Python**
- Implementing the test automation
Classifications Trees

• *Used when the QA wants to describe:*
  - combinatorial type of tests
• *Reduces the number of test cases*
  - All-pairs, pairwise or twowise (even three-wise!)
• *Generates the test cases*
The tools when using Classification Trees

**CTL XL**
- Classification tree editor and test case generator
- License: Free

**Sikuli**
- Visual technology to automate and test GUI using images
- License: Open source
- [http://sikuli.org/](http://sikuli.org/)

**Java, TestNG and Python**
- Implementing the test automation
Random test data on-demand

Housekeeping test data is often a waste of time. Also, rolling back databases, and put a SUT to a predefined state is challenging. Instead, a Test Data Service supplies the tests with data when required. Test data is specified by properties or traits, and sometimes locked for other tests. For example:

- The Test:
  “I’m running on test environment XYZ. I need a premium user with an account from Finland, which is not connected to Facebook.”

- The Test data Service:
  “I found 34 matches, and returns by random one of those. The data is locked for other tests until the caller is done (or a time out has occurred)”
The tools for the Test data Service

Apache Tomcat
- Servlet container
- License: Open source
- http://tomcat.apache.org/

MySQL
- Relational database
- License: Open source
- http://www.mysql.com/

Java and Python
Continuous test execution

We run all tests continuously.
The GUI tests takes too long to execute. Nobody wants to wait hours for the results in a build server. Therefore our tests run separately and independently from our build server.

Spin-off effects using random walks and test data
Since we use random walks when generating sequences through the models, and using random on-demand test data, we:
• verify the features and functionalities every time, but
• we do it a bit differently each time.
• Thus creating a ‘wider’ test coverage of the SUT.
The tools when running continuous test execution

**Jenkins**
- Building/testing software projects continuously
- License: Open source

**VirtualBox**
- Virtualization software package
- License: Open source
- [https://www.virtualbox.org/](https://www.virtualbox.org/)

**GraphWalker**
- Generating online test sequences
- License: Open source
- [http://graphwalker.org/](http://graphwalker.org/)

**TestNG**
- Testing framework
- License: Open source
Running tests continuously creates a lot of test results.

- Most of those results are only interesting for the test automator.
- For all others, very simple dashboards are used.
- 1 dashboard per team.

<table>
<thead>
<tr>
<th></th>
<th>OSX 10.5.7</th>
<th>OSX 10.6.2</th>
<th>XP</th>
<th>Vista 32</th>
<th>Vista 64</th>
<th>Win 7, 32</th>
<th>Win 7, 64</th>
<th>Ubuntu 12.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Album</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>Red</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Player</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radio</td>
<td>Green</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Playlist</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The tools for our test results and dashboard

**Apache2**
- Web server
- License: Open source

**MySQL**
- Relational database
- License: Open source
- [http://www.mysql.com/](http://www.mysql.com/)

**Java** and **Python**
Interesting observations

- Developers, QA and test automators sitting together (as strongly encouraged by SCRUM), will help and aid each other.
- As a spin-off, the developers has made our clients more testable.
- A test interface is being actively developed that more effectively helps test automation to verify different states.
- Developers loves getting their stuff tested around the clock!
Interesting observations

- Using visual models that depict the actual design of the tests, facilitates the communication within the team.
- It’s an awesome discussion platform between QA and test automators.
- Also, the models can be used for manual testing, as charts or maps, when QA manually explores the features, as a part of the Testing Playbook*

Interesting observations

- The simple dashboards, exposes in an efficient way, not only to the team how the SUT is doing, but also to management and stakeholders.
- Dashboards alone does not tell the whole story. That needs to be communicated time after time.
- Test automation is not a replacement for manual testing, but becoming a necessity.
Interesting observations

Why licenses matters for tools
The choice of tools, and the licenses they have, are important.
Running GUI tests are slow. So executing in parallel on a lot of machines can be one solution.
• But, be sure that they scale economy wise.