Domain-Specific Testing Languages

Mickey Phoenix
Senior Software Development Engineer, Certified ScrumMaster
SolutionsIQ

Rand Huso
Senior Software Development Engineer, Certified ScrumMaster
SolutionsIQ
Mickey Phoenix

- Specializes in narrative testing, test-driven development, automated testing, Scrum, Agile software development practices, and troubleshooting/debugging
- Recently spoke at SD West 2008 on Domain-Specific Testing Languages and Agile Usability
- Senior Software Development Engineer for SolutionsIQ
- Certified ScrumMaster
- Email: MPhoenix@SolutionsIQ.com
Rand Huso

- Over 20 years of software development experience in such roles as Software Architect, Team Lead, Senior Developer, and Lead Designer
- Began using Object Oriented methods in 1988
- Senior Software Development Engineer for SolutionsIQ
- Certified ScrumMaster
- Email: RHuso@SolutionsIQ.com
Domain-Specific Testing Languages

- Domain-Specific Testing Languages (DSTLs) express customer requirements as tests with the scope, granularity, and transparency you need.

- Dynamically extensible DSTLs help you keep the core testing tool simple while creating automated test scripts the customer can easily read, verify, and use as requirements documents.

- In this session, you'll get practical tips to foster test code re-use and reduce test maintenance costs, especially on large and long-running projects.

- Learn to use "refactor into abstraction" and "intentional testing" as two complementary paradigms for making stronger, more expressive, more maintainable tests.
What is Automated Testing?

• **Unit Tests**
  – Written by and for developers
  – Fast to execute
  – Verify correctness of a small piece of the code

• **Acceptance Tests**
  – Written by developers, testers, or customers, for customers
  – Can be slow to execute
  – Verify satisfaction of customer requirements
  – Provide a “definition of done”
How is Acceptance Testing Automated?

- Custom-coded tests (FIT fixtures)
- Click / keystroke recorders
- Command language scripts
Objections

• From Developers
  – Writing detailed tests is slow
  – Test maintenance cost grows as the product ages

• From Customers
  – No direct participation
  – Tests are written in a foreign language
  – Tests don’t prove anything except that a bar turns green
How Do We Get Customer and Developer Buy-In?

Domain-Specific Testing Languages -- dynamic, extensible, customer-specific tools for expressing test content -- address these objections.
What is a Domain-Specific Testing Language?

- Captures the Users’ Language
- Tailored and Situational
- Supports Natural Abstraction
Captures the User’s Language

- Concise and expressive
- Easily covers multiple levels of detail
- Readable by both the customer and the developers
- Implicitly transfers knowledge
Tailored and Situational

- Customized for each business domain, client, and project
- Portability and re-usability are not goals
- Over time, builds a body of knowledge that may be re-usable
Supports Natural Abstraction

- Assists logical decomposition
- Avoids getting lost in the details
- Matches the way people give instructions to each other
  - “How do you brush your teeth?”
Why Customers Love DSTLs: Overview

• Customers can participate fully
• Customers can write tests in their own language (no lossy translations)
• Customers can validate tests just by reading them – “executable documentation”
Customers Are Full Participants

- Customers pair with developers to write tests
- Customers understand developer-written tests just by reading them
Tests in Customer Language

- Translations are lossy:

| “The spirit is willing, but the flesh is weak.” | “The vodka is good, but the meat is spoiled.” |

We want as few translations as possible between the customer's "language of thought" and the test.
Tests As Executable Documentation

- The goal of Agile testing is “executable documentation”
- Tests written in a DSTL are like self-documenting code

They require minimal comments, because the test is written in a language the users can read
Why Developers Love DSTLs

- Easier and faster to write
- Abstraction reduces test maintenance cost
Appropriate Granularity Makes Tests:

- **Faster to Write**
  - fewer, bigger statements

- **Easier to Read**
  - comprehension by “chunking”

- **Easier to Verify**
  - don’t get lost in the details

- **Easier to Maintain**
  - all of the above
Abstraction Reduces Maintenance Cost

- Tests must change with changes to the spec
- Abstracting out common actions keeps changes to a single point
  - “once and only once”
Domain-Specific Testing Languages

- Let the Customer see the forest and the trees, and the Developers see each detailed leaf.
A DSTL command can encapsulate complexity, but at the cost of making it invisible to the customer (and future developers).

DSTL terms should be:

- Visible to the customer whenever possible (“sentences”)
- Written in developer language when necessary (“words”)

Transparency vs. Power
Compose DSTL “Sentences”

- Don’t hide information inside programmed DSTL commands only developers can read.
- Instead, compose higher-level DSTL commands from lower-level DSTL commands.
Why?

- Customers can “drill down” into a test.
- Increases test code re-use (lowers maintenance costs).
- Easier and faster than writing from scratch.
- Can be written by QA and customers as well as by developers.
• Write new low-level DSTL commands in code when necessary.
• Customer must “take it on faith” that the DSTL command does what it says.
Why?

- Exposes the full power of the test environment.
- Hides irrelevant detail from the customer.
- Some concepts are easy to name, but complicated to express.
- Can only be written by developers.
We can use the same tools and principles to write DSTLs that we use to write code:

- Top-down/intentional programming/ deductive reasoning
- Bottom-up/abstraction/inductive reasoning
Writing a DSTL

- Encapsulate user-irrelevant complexity.
- Factor out duplication (bottom-up abstraction).
- Intentional design (top-down decomposition).

- Collaborate with the user at all stages.
## Low-Level Test (no DSTL)

<table>
<thead>
<tr>
<th>comment</th>
<th>Assert that field_one is &quot;enabled&quot; (visible and editable, but not mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>assertJavascriptValue</td>
<td>document.getElementById(&quot;field_one&quot;).style.visibility</td>
</tr>
<tr>
<td>assertJavascriptValue</td>
<td>document.getElementById(&quot;field_one&quot;).style.editable</td>
</tr>
<tr>
<td>assertJavascriptValue</td>
<td>document.getElementById(&quot;field_one_mandatory_flag&quot;).style.visibility</td>
</tr>
</tbody>
</table>
What the User Sees

• Statically: an incomprehensible mess.
• On success: incomprehensible lines turning green.
• On failure: incomprehensible lines turning red.
### Low-Level Test (with DSTL)

<table>
<thead>
<tr>
<th>assertEnabled</th>
<th>field_one</th>
</tr>
</thead>
</table>

Copyright © 2008 SolutionsIQ. All rights reserved.
Selenium.doAssertEnabled = new function(fieldName)
{
  var field = document.getElementById(fieldName);
  var mandatoryFlag = document.getElementById(
      fieldName +
      " mandatory_flag");
  if (!field.style.visibility)
  {
    throw new Error("Field is not visible.");
  }
  if (!field.style.editable) {
    throw new Error("Field is not editable.");
  }
  if (mandatoryFlag.style.visibility){
    throw new Error("Field is mandatory.");
  }
}
What the User Sees

• **Statically:** a simple statement of intent, in their own language.
• **On success:** that simple statement turning green.
• **On failure:** an explanation, again in user terms, of the cause of the failure (and a red line).
## Mid-Level Test (no DSTL)

<table>
<thead>
<tr>
<th>type</th>
<th>bsmith</th>
<th>username</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>testPasswd</td>
<td>password</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>button=Login</td>
<td></td>
</tr>
<tr>
<td>clickAndWait</td>
<td>link=Create New Entry</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>This is a test entry subject</td>
<td></td>
</tr>
<tr>
<td>clickAndWait</td>
<td>button=Create Entry</td>
<td></td>
</tr>
<tr>
<td>clickAndWait</td>
<td>Link=List Entries</td>
<td></td>
</tr>
</tbody>
</table>

- `assertTextPresent` This is a test entry subject
- `assertElementPresent` link=This is a test entry subject
- `clickAndWait` link=This is a test entry subject
- `assertTextPresent` [no body text for this entry]
Mid-Level Test (with DSTL)

```plaintext
!include LoginStandardUser

| store | This is a test entry subject | entrySubject |
```

!include CreateEntry
!include VerifyEmptyEntry
Mid-Level DSTL Definition

- `<A TestComponent: “LoginStandardUser”>`

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>bsmith</td>
<td>type</td>
</tr>
<tr>
<td>password</td>
<td>testPasswd</td>
<td>clickAndWait</td>
</tr>
</tbody>
</table>
Mid-Level DSTL Definition

- `<A TestComponent: “CreateEntry”>`

<table>
<thead>
<tr>
<th>requireParameters</th>
<th>entrySubject</th>
</tr>
</thead>
<tbody>
<tr>
<td>clickAndWait</td>
<td>link=Create New Entry</td>
</tr>
<tr>
<td>type</td>
<td>${entrySubject}</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>Button=Create Entry</td>
</tr>
</tbody>
</table>
Example: Top-Down vs. Bottom-Up
Questions!
Further Resources

• **Narrative Testing tools:**
  - [http://storytestiq.solutionsiq.com/](http://storytestiq.solutionsiq.com/)
  - [http://selenium.openqa.org/](http://selenium.openqa.org/)

• **Our website:**
The authors would like to thank SolutionsIQ of Redmond, WA and STSII of Dublin, CA for their generous support of this presentation.
More from SolutionsIQ at Agile2008

**Architecture in an Agile Organization**
SolutionsIQ experts share their experiences and practical approaches to better align businesses with architecture goals while adhering to Agile principles.

— **Chris Sterling**, Principal Consultant, Certified Scrum Trainer and Agile Coach

**Narrative Testing: Tools for Story Test-Driven Development**
Increase your customers’ confidence in testing by leveraging script-based testing tools and DSTLs to express Story Tests in the user’s own language.

— **Mickey Phoenix**, Senior Software Development Engineer

**Panel Discussion: Troubleshooting Distributed Agile Team Projects**
Leading Agile experts *Esther Derby, Hubert Smits, Tamara Sulaiman, Samir Shah* join Monica Yap to share their experiences working with distributed Agile teams.

— **Monica Yap**, Engagement Manager, ScrumMaster, Agile Coach

**Punctuated Continuity: Using Ritual and Ceremony to Avoid Process Fatigue**
Learn techniques that can be employed to keep repetitive Agile routines invigorating, pulled from actual experiences with teams practicing XP and Scrum.

— **Michael Tardiff**, Agile Team Lead and Coach
Swarming: The Birds and the Bees and Agile
Discuss the fascinating set of swarming behaviors in the animal world that resonate strongly with some of the central tenets of Agile development.

– Dhaval Panchal, Agile Coach, Analyst, Certified Scrum Practitioner

Assembling a Real-Time Collaborative Development Platform in the Cloud
SolutionsIQ CEO demonstrates a whole platform for Agile development featuring mashups of SaaS and open-source tools for development and continuous integration

– Charlie Rudd, Chairman and CEO
Thank You!

• **Come to the SolutionsIQ booth at Agile 2008**
  – *Pick up a **free Agile t-shirt**, and*
  – **Schedule one-on-one sessions** with SolutionsIQ speakers

• **Visit** [solutionsiq.com/agile2008](http://solutionsiq.com/agile2008) **for additional Agile 2008 materials and related content from SolutionsIQ**

**About SolutionsIQ**
SolutionsIQ offers a full spectrum of services to develop **software** and fulfill technical **talent** needs, while improving your Agile **knowledge** and capabilities. **Clients include** AT&T (Cingular), Amazon, Corbis, Expedia, Federal Home Loan Bank, InfoSpace, Key Bank, Nike, Nordstrom, Regence Blue Shield, Safeco, US Bank, and Washington State University. A Microsoft Gold Certified Partner, SolutionsIQ is also a member of the Java Community Process, Scrum Alliance, Software Association of Oregon, and Washington Technology Industry Association. Learn more at [www.SolutionsIQ.com](http://www.SolutionsIQ.com).