The Design and Implementation of a Flexible, Reusable, and Maintainable Automation Framework

HUNG Q. NGUYEN
LogiGear® Corporation
Objectives

• Prepare you to build a successful and reusable automation architecture
• Share the keyword approach to creating an automation framework
• Share lessons learned in implementing a flexible architecture
Background: The Evolution

• The early days
• Developing an automation framework
• The table-driven approach
• The keyword-driven approach
The Early Days

• Collect acceptance/regression test cases to be automated
• Record and script test cases
• Improve reusability
  – Parameterize hard-coded values
  – Separate data from code by moving variables to INCLUDE files
  – Create utility functions to be shared
• Train test specialists to run scripts
The Next Wave: Creating a Framework

• Work with test specialists to understand their testing needs
• Go beyond acceptance/regression tests--Analyzing user-scenario test cases
• Recognize the difference between task driven and object-driven test cases
Object-Driven vs. Task-Driven

- **Object-Driven**
  - Click User Name text box
  - Enter your_ID
  - Click Password text box
  - Enter your_password
  - Click Login button

- **Task-Driven**
  - Log in using
    - User Name = your_ID
    - Password = your_password
• Pre-separating data and code
  – Start by defining functions to be written
  – Variablize data and keep variables in INCLUDE files

• Pair up a test specialist and an automation engineer to improve communication and to ensure that the framework design and implementation meet the test objectives

• Train test specialists to run test scripts
The Table-Driven Approach

• Take advantage of tester’s familiarity with test case creation using tables and matrices
• Accommodate localization projects
• Recognize the importance of patterns in test cases
• Enable testers to catalog test cases with Excel spreadsheets
• Enable testers to specify expected results in spreadsheets
A Table-Driven Example
• for (i=1; i<= iLastDataSet; i++)
  – Open the dialog box.
  – Use the data in DataSet[i] (The first set is 1 and the last set is 12) to set the values of Match Case, Match Whole Word and Find What controls.
  – Click Find Next.
  – Verify the results.
The Need for Improvement

- Business issues
- People and process issues
- Technology issues
The Business Issues

• Need to expand our service offerings and share success through our test automation expertise
• Need to have a methodology for quick deployment of test automation
• Need to build a transferable architecture
• Need a better approach to test automation job costing
The Business Issues

• Need to deliver an automation program that is practical, explainable, and trainable
• Need to be more cost effective through reusability across projects
• Need to make technology a viable business solution
• Need a tangible approach to deciding between manual testing and automated testing
The People and Process Issues

• Need to standardize test methodology--Enabling testers and automation engineers to collaborate
• Enable testers to better specify their needs and automation engineers to better serve those needs
• Need to integrate test automation as part of the process of software testing
The People and Process Issues

• Need testers to focus on test case design, and automation engineers to focus on driver script writing
• Make data more visible and understandable from the human perspective
• Need to incorporate test case design techniques with Excel, which test specialists are already familiar
The Technology Issues

• Need to build an architecture that’s tool independent as well as application independent
• Need to improve the ability to share code across projects and tools
• Need to separate control of task variables, input variables, and code
• Need to integrate action keyword into the existing data-driven model
The Technology Issues

- Want to focus the development and maintenance of test scripts on the navigation of the application under test
- Need to take advantage of Excel features to automate test case and test data creation
- Need to incorporate test case design techniques using Excel, any database, XML, or other viable data service solutions
The Integrated Solution

Integrated Testing Solutions =
[Test Specialist’s Domain Expertise] +
[Manual Testing] +
[Automated Testing: Reusable Framework & Application Specific Scripts]
The Development Process

- Research possible solutions and evaluate options
- Develop requirements
- Develop the architecture
- Build the framework
- Test the framework
- Develop documentation
- Deploy the framework on a real project
- Measure performance and refine the design
Research Possible Solutions

• Learn from past experience
• Discuss possibilities with software developers
• Talk to friends
• Read books
  – Recommend “Software Test Automation” by Graham and Fewster, 1999, Addison-Wesley
• Use the Internet
  – Recommend www.QACity.com, the Automated Testing page
The Requirements

• Clearly state the business, people/process, and technology objectives

• Set expectations through well defined deliverables (e.g., requirement and design documents, code modules, whitepapers, training materials, etc.)

• Clearly define ways to measure success (e.g. quality of the design and code, budget, schedule, customer approval upon deployment, etc.)
The Architecture

Keyword Architecture

Test Designer
- Data Services

Test Interpreter
- Business Logic

Test Driver
- Dispatching Services

Driver Scripts

Test Execution Services

Application Under Test (AUT)
The Architecture

**Keyword Architecture**

- **Test Designer**
  - Database
  - XML

- **Test Interpreter**
  - FileName
  - GetTaskName
  - GetTaskRows
  - GetTaskParameters
  - etc.

- **Test Driver**
  - Main()
  - RunTestCase

**Driver Scripts**

testcase Login(parameters)
etc.
The Architecture

- testdriver.t
- interpreter.inc
- driverscripts.t
The Architecture: The Test Designer

MyWorksheet.xls = MyTestplan.xls

Sheet1 = TestSuite1

Sheet2 = TestSuite2

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Test Case 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>Test Case 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Architecture: The Test Designer

<table>
<thead>
<tr>
<th>Test Plan</th>
<th>Test Suite</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Test Suite: Administration Acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author: Joe Tester</td>
</tr>
</tbody>
</table>

### 1. Log in as Admin

<table>
<thead>
<tr>
<th>Task</th>
<th>User ID</th>
<th>Password</th>
<th>GroupMember</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogIn</td>
<td>myID</td>
<td>myPassword</td>
<td>myID in Admin</td>
</tr>
</tbody>
</table>

### 2. Adding users

<table>
<thead>
<tr>
<th>Task</th>
<th>Username</th>
<th>First Name</th>
<th>Last Name</th>
<th>Password</th>
<th>Email</th>
<th>Phone</th>
<th>Pager</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnterUser</td>
<td>joeQA</td>
<td>Joe</td>
<td>Tester</td>
<td>joeQA</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 100</td>
<td>NONE</td>
</tr>
<tr>
<td>EnterUser</td>
<td>joeDay</td>
<td>Joe</td>
<td>Tester</td>
<td>joeDay</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 101</td>
<td>NONE</td>
</tr>
<tr>
<td>EnterUser</td>
<td>joeProj</td>
<td>Joe</td>
<td>Tester</td>
<td>joeProj</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 102</td>
<td>NONE</td>
</tr>
<tr>
<td>EnterUser</td>
<td>joeAdmin</td>
<td>Joe</td>
<td>Tester</td>
<td>joeAdmin</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 103</td>
<td>NONE</td>
</tr>
</tbody>
</table>

### 3. Adding groups

<table>
<thead>
<tr>
<th>Task</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnterGroup</td>
<td>Marketing</td>
</tr>
<tr>
<td>EnterGroup</td>
<td>Engineering</td>
</tr>
<tr>
<td>EnterGroup</td>
<td>Sales</td>
</tr>
</tbody>
</table>

### 4. Checking Users and Groups

<table>
<thead>
<tr>
<th>Task</th>
<th>Username</th>
<th>First Name</th>
<th>Last Name</th>
<th>Password</th>
<th>Email</th>
<th>Phone</th>
<th>Pager</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckUser</td>
<td>joeQA</td>
<td>Joe</td>
<td>Tester</td>
<td>joeQA</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 100</td>
<td>NONE</td>
</tr>
<tr>
<td>CheckUser</td>
<td>joeDay</td>
<td>Joe</td>
<td>Tester</td>
<td>joeDay</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 101</td>
<td>NONE</td>
</tr>
<tr>
<td>CheckUser</td>
<td>joeProj</td>
<td>Joe</td>
<td>Tester</td>
<td>joeProj</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 102</td>
<td>NONE</td>
</tr>
<tr>
<td>CheckUser</td>
<td>joeAdmin</td>
<td>Joe</td>
<td>Tester</td>
<td>joeAdmin</td>
<td><a href="mailto:joe_tester@qacity.com">joe_tester@qacity.com</a></td>
<td>(650) 566 1212 ext 103</td>
<td>NONE</td>
</tr>
</tbody>
</table>
Building the Framework

• Prototype the components
• Implement the Test Designer
• Implement the Test Interpreter
• Implement AUT specific Test Drivers
• Add the reporting function to the Test Interpreter
• Test, fix bugs, and write documentation
Lessons Learned

• Clear requirements help focus the team on the important issues.

• Leaving “Fill in the blank” sections in requirements is manageable.

• Spending time on designing and prototyping helps flush out design issues; making it more scaleable, and helping write more maintainable code.

• If the project is overly complex and the schedule is aggressive, you may need to scale back. Don’t forget to communicate changes in your plan.
Lessons Learned

• The necessary information is available! We need a way to find and analyze relevant information more quickly and effectively.

• Thoroughly research your options. Choose your designs wisely by taking business issues, people and process issues, and technology issues into consideration.

• Keep in mind that your solution might be used by one group, and maintained by another group.
Lessons Learned

• Your effort is a serious development project. Treat it as such: The key to success is good planning, scheduling and budgeting.

• Get feedback! How else can you learn?

• It won’t be perfect! It’s acceptable to learn from mistakes and refine the design as you go. Iteration and hard work make perfection.

• The keyword approach works!
Acknowledgment

Special thanks to Hans Buwalda for sharing his experience and vision on the action-word approach to creating test automation framework.
Hung Q. Nguyen is Founder, President and CEO of LogiGear Corporation, Silicon Valley software testing company whose mission is to help software development organizations deliver the highest quality products possible while juggling limited resources and schedule constraints. LogiGear offers many value-added services including application testing, automated testing and web load/performance testing for e-business and consumer applications. Nguyen’s company produces and markets TRACKGEAR™, a web-based defect tracking system. LogiGear also specializes in Web application, handheld communication device and consumer electronic product testing, and offers the software development community a comprehensive “Practical Software Testing Training Series.” In the past two decades, Nguyen has held leadership roles in business development, engineering, quality assurance, testing, product development, and information technology. Nguyen is the author of Testing Applications on the Web (Wiley) and co-author of the best-selling book, Testing Computer Software (Wiley). He also develops and teaches software testing courses for UC Berkeley and UC Santa Cruz Extension, and for LogiGear. He holds a Bachelor of Science in Quality Assurance from Cogswell Polytechnical College, and is an ASQ-Certified Quality Engineer and active senior member of American Society for Quality.
LogiGear® Corporation is a full service software quality engineering firm that provides testing expertise and resources to software development organizations. Some of our value-added services include application testing, automated testing, and web load/performance testing for e-business and consumer applications. LogiGear specializes in Web application, hand-held communication device, and consumer electronic product testing. LogiGear also produces and markets TRACKGEAR™, a Web-based defect-tracking solution, and offers QA Training through the Practical Software Testing Training Series.

www.logigear.com