Anatomy of an Automated Testing Framework

Last month we discussed why companies embark on automated testing, why some companies fail in this effort, and how using a keyword driven automated testing framework can increase your chance of success. To view that newsletter, visit http://www.pragmaticsw.com/newsletters/Newsletter_2009_06_SP.htm. This month we discuss the key ingredients of an automated testing framework that can lead to automation success.

Why Automate Your Test Cases?
Many companies run their regression test cases manually, so when does it make sense to begin automating your regression test cases? It makes sense to automate your test cases when you can no longer run the regression test cases on each build created. For example, if you are doing daily or weekly builds of your code to quality assurance and you cannot quickly run your regression test cases with each build, it is time to consider automating them. Automating your test cases provide these benefits:

- **Quicker Releases** – By having your regression test cases run automatically, your software quality team can concentrate on testing new features of your software and less time regressing existing features.
- **Higher quality releases** – Your software releases will have fewer bugs and require less customer support because they will be of higher quality.
- **Happier Customers** – Your customers will be happier and more willing to serve as testimonials for future prospects.

What is an Automated Testing Framework?
Most automated tools require the test engineer to understand a scripting language (VB Script, Java Script, etc.) to write their automated test cases. These tools usually have the ability to create the scripts using record and playback, but this does not always write the most efficient scripting code and is not as re-usable and maintainable. An **Automated Testing Framework** is a set of assumptions, concepts, and practices that provide support for automated software testing, allowing you to re-use automated test cases, reduce maintenance, and enhance maintainability.

Anatomy of a Successful Automated Testing Framework

When designing your Automated Testing Framework, consider these features, as they can dramatically improve success and reduce the chance of abandonment:

1. **Keyword Driven Testing** - If your test engineers are not experts at scripting languages, consider replacing your automated scripts with keyword driven testing. This approach allows a tester (or even a subject matter expert) to create automated tests by describing each step of the automation. For example, if you are automating the login process of your application, your user will access your application, type in their user-id and password and press a button to login. Traditionally, testers would do this by writing VB Script that will navigate to your application, identify each object on the screen (user-id, password and login button), then write script to enter in the user-id, password and to press the login button. With keyword driven testing, the tester does not need to understand the scripting language to make this happen, they can simply describe the event (navigate to your application, enter in "abc" for the user-id, enter in "xxx" for the password, press the Login button when done). As you can imagine, this is a much simpler approach to automated testing than scripting.
2. **Allow Data Iterations -** It is good practice to allow your testers to re-use automated test cases,

but allow them to run them with different sets of data. For example, you might use the same automated test case for logging into your software, but allow the test case to be run multiple times with different user-id and password combinations to test different scenarios. If your automated testing framework is flexible enough to allow your testers to define different data iterations, it will reduce time spent creating automated test cases.

3. **Allow Running on Multiple Hosts** - When designing your automated testing framework, consider multiple hosts. For example, as your quality assurance demands grow, you may need to setup a QA Lab with multiple servers. Once that is done, you will need a way to launch your automated test cases on multiple servers, so consider building that into your automation framework.
4. **Allow Scheduling of Automation Runs** - Once you have developed your automated test cases, you will want to schedule them so that they can run unattended. When designing your automation framework, build in the capabilities to schedule automation runs at different intervals (days of the week), allow recurrence (run every day at 6 p.m., etc.), and allow them to be scheduled on different hosts.
5. **Reporting Run Results** - When developing your automation framework, consider how you will report on the run activity of your automation effort. If you are using an existing test management solution for your manual test effort, consider integrating your automated test result data in with the reporting of your test management solution, that way you can analyze your manual and automated test effort in a consolidated fashion.

Example of an Automated Testing Framework
If you wish to learn more about automated testing frameworks and view how an existing automated testing framework was developed, view this User's Guide: http://www.pragmaticsw.com/UsersGuide_KDT.pdf.

Helpful Resources

Below are some helpful resources and templates to aid you in developing software solutions:

- **Software Planner** - http://www.SoftwarePlanner.com
- **AutomatedQA TestComplete (Automated Testing Tool)** - http://www.TestComplete.com
- **STAR QA (Automated Testing Resources)** - http://www.star-qa.com
- **Software Development /QA Templates** -http://www.pragmaticsw.com/Templates.asp
- **Test Case Training** - http://www.PragmaticSW.com/Services.asp
- **Pragmatic Agile Development** - http://www.pragmaticsw.com/PADOverview.pdf

About the Author

Steve Miller is the President of Pragmatic Software (http://www.PragmaticSW.com). With over 24 years of experience, Steve has extensive knowledge in project management, software architecture and test design. Steve publishes a monthly newsletter for companies that design and develop software. You can read other newsletters at http://www.PragmaticSW.com/Newsletters.asp.

Pragmatic Software Company, Inc.
7935 E. Prentice Ave, Suite 105
Greenwood Village, CO 80111 USA
Tel:+1 303.768.7480