Getting Started With Automated Testing

Michael Kelly
www.MichaelDKelly.com

Sept 22, 2004
Indianapolis Quality Assurance Association

Introduction

Focus will be on developing a strategy for automation and putting together an automation team.

Depending on your context, some of the may be more valid or necessary than others.

There is no particular order in which the steps need to be executed.

The steps discussed apply primarily to teams of three or more people.

Not geared to a specific test tool or tool vendor.

Assumptions

Automated testing is the use of tools to assist with a testing effort.

The following is a small sampling of different types of automation available:

- Load and performance testing
- Installation and configuration testing
- Testing for race conditions
- Endurance testing
- Helping the development effort with smoke tests and unit tests
- Analyzing code coverage and runtime analysis
- Automation of test input generation
- Checking for coding standards and compliance
- Regression testing
- many more…

Automated testing is a subset of the overall testing profession.

1. Set Goals for Automation

Do you want to find bugs?
Do you want to establish traceability for some sort of compliance?
Do you want to support the development team?
Do you want to ensure that there are no changes in the software?

- Use your goals to set the strategy and scope for your automation.
- When deciding what to automate for your first time, start with small milestones.
  - Start with testing simple functionality.
  - Start with just one virtual user and set your goal at a low number (no more than twenty).
  - You should be able to use record-and-playback features to perform basic testing or you can do some basic scripting.
- Once you have determined your goals, you will need to start thinking about what kinds of skills you will need to have on your team.

2. Have at Least One Programmer on Your Team

The most important thing you can do is select the right team. A tool will always only be a tool.

To ensure efficient well-planned automated testing, you will need to have at least one experienced programmer in your testing-automation group.

Record-and-playback features only offer quick solutions for the most common tasks and controls. For advanced automation of any kind or for any custom controls, you'll need to be able to write your own *maintainable* code.

Beware you don’t employ only programmers. All team members should be testers first. Skilled in the mental arts of testing as well as armed with the ability to code and design test systems.

3. Get Familiar with the Tools

The scope of your automation effort will depend heavily on the tools you use and have access to.

Most automated testing tools are designed for unit testing, some sort of regression testing, or performance testing.

There are now many good open source tools now available.

Regardless of what tool(s) you select, you will need to be sure that you spend time learning how to use them properly.

- Go through the tutorials that are provided
- Read through whatever documentation is available
- Attend a training class
- Hire a training consultant to come in and spend some time with you

3. Get Familiar with the Tools

- Disk imaging tools
- File scanners
- Macro tools
- Memory monitors
- Environmental debuggers
- Requirements verifiers
- Test procedure generators
- Syntax checkers/debuggers
- Runtime error catchers
- Source code testing tools
- Environment testing tools
- Static and dynamic analyzers
- Unit test tools
- Code coverage tools
- Test data generators
- File comparison utilities
- Simulation tools
- Load/Performance testing tools
- Network testing tools
- Test management tools
- GUI testing tools
- Any scripting language (Ruby, Pearl, VB, etc…)

3. Get Familiar with the Tools

Open Testware Reviews:
http://tejasconsulting.com/open-testware/

- A source of information about freely available test tools.

Testing Tool Information:
http://www.grove.co.uk/Tool_Information/Choosing_Tools.html

- Short Sharp Advice about how to choose a testing tool.

Test Tool Evaluation Center:
http://www.testtoolevaluation.com/index.asp

- An online resource with a lot of info and wizards for tool comparisons.

4. Set Some Standards

Naming standards

- Scripts
- Test logs
- Directory structures
- Data structures
- Verification points

Coding standards

- Steal the standards used by the development staff
- Go online and find some
- Customize them to fit your needs and the needs of your team

4. Set Some Standards

Environmental standards

- Ensure that the computers you use all have the same operating system, RAM, hard drive space, and installed software configurations.
- The only differences should be the specific differences you are looking for in your configuration testing.

Procedures for error and defect tracking

- Describe how to log errors in test scripts
- Describe how to submit defects via your defect-tracking tool
- Describes how to code workarounds into scripts, and remove them after a bug is resolved

5. Establish Some Baselines

Figure out what the bare minimum tests are for that type of testing to be successful and implement them in very small, very simple chunks.

In past projects I have established the following baselines:

- For **regression testing**: a baseline set of scripts that test basic functionality in the system using the most commonly used paths through the application.
- For **performance testing**: a baseline set of scripts that target each standalone service for the application that run successfully with one virtual user.
- For **code coverage analysis**: try to develop a series of tests that will allow you to get some level of coverage in each major module of your application.
- For **source code checking** tools: try to get a small subset of the application code to pass the verifications and modify/refine the rules as needed.
- For **establishing traceability**: take a small series of test cases and, using your test management tool, establish traceability from the manual/automated scripts, to the test cases, to the requirements. Generate a simple report that shows this traceability in a usable format.
- For **test data generation**: generate a subset of the overall data you will need for testing and verify the data's correctness and randomness.

6. Look for Ways to Modularize Your Scripts

Look through the code in the scripts for repetitive calls or other common actions. The goal is to make maintenance is as easy as possible.

For ideas on how to modularize your scripts:

- Look at user communities for the tools you use
- Read literature on the topic from any of the major testing websites
- Talk with your development team
- Hire in a consultant to train your staff on what to look for.

7. Use Data Structures & Data-Drive Techniques

Effective and cost-efficient automated testing is data-driven.

- Offers the greatest flexibility when it comes to developing workarounds for bugs
- Decreases time spent maintaining code
- Allows for the fastest development of large sets of test cases

Most likely, your tool will have some method for implementing this. If not you can use:

- Spreadsheets (any cvs file)
- Databases (any ODBC source)
- DAO/ADO objects
- Web services

8. Document Everything (…that it makes sense to document…)

Find out what minimum documentation is required for your context.

After you have met the minimum requirements:

- Document why you designed things the way you did.
- Document what each module does, and what each function in it does.

This documentation is often most useful as:

- Training material
- Future technical reference (much like an API)
- Helps you keep track of lessons learned

Review

1. Set goals for automation.
2. Have at least one experienced programmer in your testing-automation group.
3. Get familiar with your tools.
4. Develop standards for your team.
5. Establish some baselines.
6. Modularize and build reusability and maintainability into your scripts.
7. Use data-driven testing techniques whenever possible.
8. Document what makes sense to document.

Review

- Focus on the clarity and simplicity of your goals
- Focus on your understanding of the tools
- Focus on the makeup of your team
  - It has been my experience that the makeup of the team is the most influential factor in an automation efforts success.
  - Test automation requires a balance between programming knowledge, testing skill, and business knowledge.
  - Incorporating all of those assets into your team will be the most important step of all.

Thank You

Questions?