TP326, 09/01

Rational. software

Automated Testing: A Silver Bullet?

Dawn Haynes
Technology Evangelist
IBM Rational

Table of Contents

Introduction

In 1986, Frederick P. Brooks, Jr. wrote a paper called "No Silver Bullet — Essence and Accidents of Software Engineering." This paper conveyed some of the expectations that folks had about advances in software engineering technologies and contrasted them with the realities. His argument can be summed up as follows:

There is no single development in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

Brooks encourages us to think of technologies and techniques as more evolutionary than revolutionary. When it comes to thinking about introducing automation to any kind of testing effort, I would like to encourage a similar approach.

In the five years I have worked with potential customers of automated testing products and solutions, I have encountered a significant amount of "silver bullet" thinking. This manifests itself through assumptions such as:

1. We will be able to automate all testing!
2. Test automation will increase productivity so much that we'll be able to do all the testing with fewer people (eliminate staff).
3. Test automation is so easy that we won't need to do any training.
4. Automation will reduce our whole testing workload.
5. We won't need to do any test planning.
6. Doesn´t automation make human testers "obsolete" or "redundant"?
7. That time-intensive test design effort will no longer be necessary.

Although I hate bursting people's bubbles, I have always felt compelled to help them understand the difference between implementing automated testing and attaining the Holy Grail. Most often, this means explaining what automated testing actually *is*, and what automated testing tools and solutions can actually *do*.

Are You Saying That Automated Testing Is NOT a Silver Bullet?

That's the idea. Automated testing — or the implementation of test automation strategies and tools — is just one big hammer in the tester's toolbox. Notice that I said it's a *tool* and that its place is *in a toolbox*. I'm purposefully avoiding equating automation with human testers: there's just no replacing those. Still, there is no question that test automation is powerful stuff and can provide benefits in terms of efficiency and thoroughness. The key is determining when and how to use its power. Let's begin doing that by posing another question.

Is There Ever Enough Time to Test Everything?

My guess is that the answer to this question is a universal and resounding "No!" There's always one more thing we could test or another platform or configuration we'd like to try. But as deadlines and ship dates draw closer, the time allocated for each testing cycle shrinks. So, how do software development project managers and testing groups deal with this? Typically they reduce the amount of testing they do for each cycle leading up to release. Have you ever experienced this? Ideally, it would be better to do some risk-based analysis to determine what to eliminate; more often, however, teams just narrow the focus of the entire testing cycle to verifying fixed defects. And often there isn't even enough time to complete this reduced testing plan.

How many products get shipped only when testing is complete? I don't hear about such scenarios very often. Usually teams look at other factors when they make a ship/don't ship decision:

- Have we run out of time?
- Run out of budget?
- Run out of resources?
- Run out of pizza and beer?

Unfortunately, when testing is cut off arbitrarily, the development team doesn't know enough about the product's overall quality, and they run the risk of shipping serious problems. Is this a dilemma we could resolve by applying the power of automated testing? Let's investigate.

How Can Automated Testing Help?

Before you build a plan to implement automation, you should understand how you define it. In other words, what does automation mean to you? Here are a few ways I've heard people describe automated testing:

1. Testing that requires no human intervention at all.
2. Test scripts.
3. Test tools.
4. I don't know.

Sometimes people interpret the notion of automated testing too narrowly, focusing only on test scripts generated by tools or by programming. In fact, *automation* can have a much more expansive meaning. Consider this definition from a Quality Engineering group that is building a set of test automation guidelines:

Automation, in our context, is the use of strategies, tools and artifacts that augment or reduce the need for manual or human involvement or intervention in unskilled, repetitive or redundant tasks.

In addition to this definition, the guidelines provide examples of automation methods the group employs, a few of which are listed in Chart 1.

Automation Method	Description	Example
Template	An outline of an artifact, usually containing formatting and guidance for adding content. Used as a starting point for creating an artifact.	Test case or test plan template (created internally, based on a sample in a book, or taken from a third party tool)
Test Scripts	Machine readable/executable instructions that (typically) automate the execution of a test. May be generated by a tool and hand coded.	Visual Test scripts, Rational Robot scripts, Perl scripts, or other coded executables or Dynamic Link Libraries.
Images	Compressed files or backups that are used to quickly return an environment to a pre-determined state. (In preparation for manual or automated testing.)	Create disk images using third- party tools or backup software.
Macros	Machine readable/executable instructions (usually in the context of a specific application) that automate the execution of a specific task or set of tasks.	Third-party tool macros (like those from Microsoft Excel), which capture, format, and merge data for management reporting.
Batch Files	Machine readable/executable instructions (usually in the context of an operating system or Integrated Development Environment (IDE) that automate the execution of a specific task or set of tasks.	Instructions used to install/configure specific options using the DOS (or other IDE) command console.

Chart 1: Automation Methods for Testing

Does this small set of examples get you thinking about automation in a different way? Now, it is important to define what automation means to you and your test team. Then you can use that definition to begin building a set of automation guidelines so that anyone on the team can quickly assess, using the same methods, whether a task is a suitable candidate for automation.

Creating Automated Testing Guidelines

Here are some strategies and issues to consider as you shape your definition and guidelines.

Define where automation fits

- Target specific areas of the total effort as candidates for automation.
- Start with highly redundant tasks or scenarios.
- Automate repetitive tasks that are boring or tend to cause human error.
- Focus on well-developed and well-understood use cases or scenarios first.
- Choose relatively stable areas of the application over volatile ones.
- Enhance automation by using data-driven testing techniques (increase the depth and breadth of testing coverage).
- Don't make everyone on the test team responsible for automation; designate a few specialists.
- Know that 100 percent automation is not a realistic goal, and that manual testing will still be essential.

Plan to do more testing

- Automating repeated tests leaves more time to test using other methods:
- Increase exploratory testing.
- Increase configuration testing.
- Build more automation.
- Do more manual testing, especially for high-risk features.
- Plan carefully: decide which tests will be done manually and which tests can be automated — don't just try to automate everything.
- Design all tests and document each design. If an automated test cannot be run, ensure that the test can be performed manually instead.

Think of automation as an investment

- Train users to fully leverage the automated tools.
- Build a reusable code base.
- Keep the tests modular and small for easier maintenance.
- Document the test scripts (code) for verification and reuse.
- Enforce back-up procedures.
- Utilize source control.
- Realize that automation *is* a software development effort: It often requires code generation.

Implement automated testing iteratively

- Don't attempt to automate all tests on day one. Gain experience and implement slowly.
- Start with a small portion of the total test plan and iteratively add to the automation test suite over time (i.e., Ramp up in a realistic and controlled way).

What Else Can Automation Do For Me?

Although automated testing requires a big up-front investment in terms of planning and training, it does pay off in a number of big ways, too. It can give you:

- Better quality software — because you can run more tests in less time with fewer resources.
- Potential for more thorough test coverage.

- More time to engage in other test activities, including
- Detailed planning
- Careful test design
- Building more complex tests (data driven, adding code for condition branching or special reporting, etc.)
- More manual testing, not less!!!!

Automated testing also provides intangible benefits. It can give testers:

- An opportunity to gain new skills (i.e., skill building and learning opportunities).
- Opportunities to learn more about the system under test because automation can expose internals, like object properties and data. (Better understanding of the system produces better testers.)

Now that you know what automated testing *is* and what it can *do*, I hope you'll use this knowledge to ensure more and better testing for your products. Although it's no silver bullet, automated testing is a **great tool**; if you match it with the right jobs, you'll get great results.

Now that you know what automated testing *is* and what it can *do*, I hope you'll use this knowledge to ensure more and better testing for your products. Although it's no silver bullet, automated testing is a **great tool**; if you match it with the right jobs, you'll get great results.

References

For more information on some of the topics mentioned here, please read the following articles from Cem Kaner's Web site: http://www.kaner.com/articles.html

1. "Architectures of Test Automation"
2. "Improving the Maintainability of Automated Test Suites"
3. "Avoiding Shelfware: A Manager's View of Automated GUI Testing"

Acknowledgements

Thanks to Cem Kaner for suggesting links to his articles.

I'd also like to acknowledge Ted Squire of IBM Rational and James Bach of Satisfice, Inc., for their careful review and assistance in the development of this article. For more information about Satisfice and its acclaimed testing seminars, please visit www.satisfice.com.

IBM software integrated solutions
IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

- *DB2® software helps you leverage information with solutions for data enablement, data management, and data distribution.*
- *Lotus® software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.*
- *Tivoli® software helps you manage the technology that runs your e-business infrastructure.*
- *WebSphere® software helps you extend your existing business-critical processes to the Web.*
- *Rational® software helps you improve your software development capability with tools, services, and best practices.*

Rational software from IBM
Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 rely on Rational tools to build better software, faster. Additional information is available at www.rational.com and www.therationaledge.com, the monthly e-zine for the Rational community.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America
01-03 All Rights Reserved.
Made in the U.S.A.

The IBM home page on the Internet can be found at **ibm.com**