Fully Leverage Agile Test Automation

Technical Success and Return on Investment

RBCS

TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Introduction

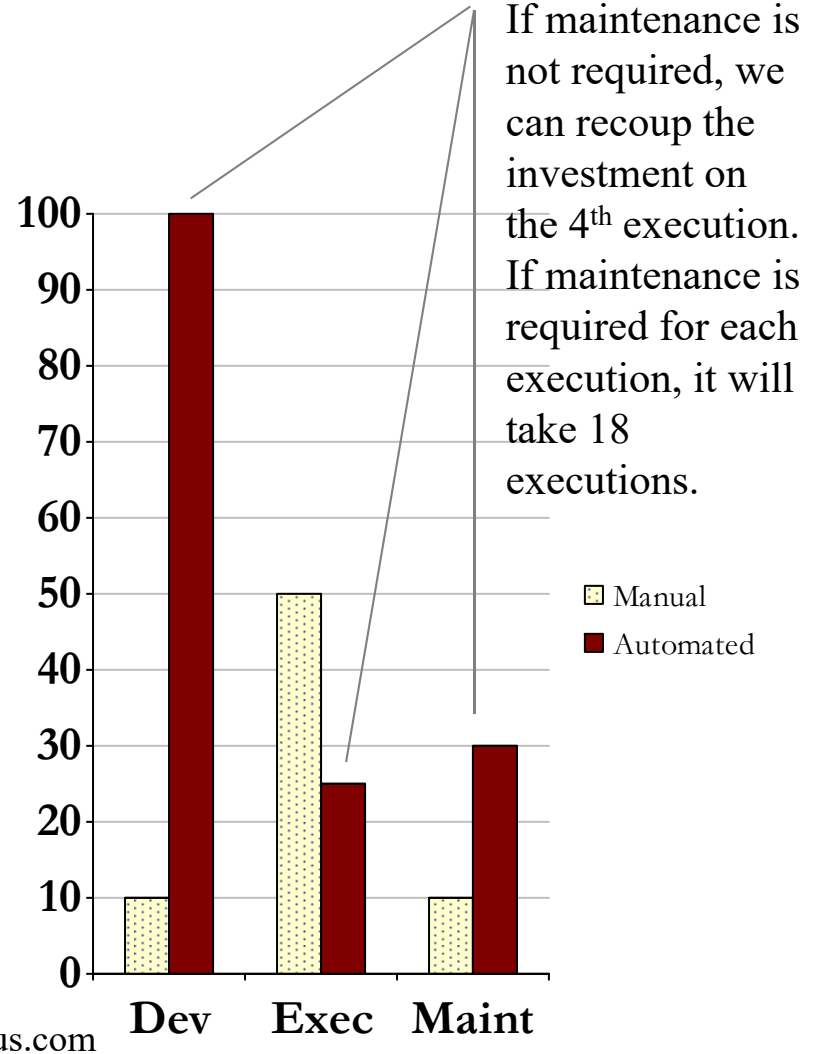
- "Give me a place to stand, and a lever long enough, and I will move the world."
- In Agile, the place to stand is testing, and the lever is test automation
- How well are you using that leverage?
- Many of our clients miss out on the full value of test automation
- Proper done, test automation can:
  - Reduce failure costs
  - Accelerate schedules
  - Expand test coverage
- Let's look at how to achieve technical success and high ROI in test automation…

Automated Testing Overview

- Automated tests can be static, white-box, black-box types…or multiple types
- Automated tests can be built from commercial, open-source, or custom testing tools
- In spite of the name, automated testing tools only automate a portion of the testing process
- Successful test automation can provide solid benefits
- Inappropriate automation is a frequent cause of trouble

Test Automation ROI

- A return on most test automation efforts requires:
  - Sufficiently high regression risks to justify repeating tests
  - A test system design that keeps the costs of running and maintaining the automated tests well below the cost of equivalent manual tests
  - A relatively stable system
- Must recoup high up-front test development costs over multiple projects



Continuous Integration

- Present in almost all of our clients with successful agile implementations
- Most use either Jenkins or Hudson
- Most include some additional tools for at least unit testing
- Supports testing by making the build process faster, more reliable
- Can create problems when builds are auto-deployed to test environments

Static Code Analysis

- Some of our clients extend continuous integration with static code analysis
- Sonar and splint are typical open-source options, though many clients opt for commercial static tools such as Klocwork or Fortify
- A few (too few) of our clients take advantage of complexity analysis tools (e.g., pmccabe, BattleMap)
- To understand the output of these tools, knowledge of the programming language is required
- These tools don't replace proper code reviews, but contribute to more secure, more maintainable code

Unit Testing

- Most of our clients using continuous integration extend it with unit testing
- The xUnit family of tools is most typically used (Cpp-Unit, J-Unit, etc.), and extends to many, many programming languages
- Some (just a few) of our clients use commercial unit test tools (e.g., Parasoft's)
- Some report developers using these tools for TDD, though what is meant by TDD varies widely among practitioners
- Code coverage analysis tools (e.g., gcov, Cobertura, Bullseye) are often used in conjunction with these tools
- Many developers don't have training in proper test design, so these tests are less useful than they could be
- Unit testing tops out at about 50% defect detection effectiveness, even when done well
- Testers should learn the basics of unit testing, white-box test design, etc., and help developers get more value

Feature Verification Testing

- Many tools for these activities
  - Fitnesse used commonly for ATDD
  - Cucumber family and Rspec used commonly for BDD
- None of our clients use commercial tools for either ATDD or BDD
- Developers and testers use these tools collaboratively
- Business stakeholders review the test results, and ideally review the tests as well
- Often (but not always) included in continuous integration frameworks
- Some of our clients who tried BDD gave it up due to maintainability issues, switched to ATDD

GUI Test Automation

- There are many, many open-source and commercial tools for GUI test automation
- Selenium is the most commonly mentioned open-source tool, but there are many others
- UFT, Rational Robot, Test Complete, and too many others to mention in the commercial world
- Data-driven or keyword-driven architecture is essential for maintainability
- Using an experienced lead (5+ years of test automation experience) is necessary for good architecture
- When I hear about test automation problems, it's usually in the context of GUI test automation

Performance Testing

- For open-source, JMeter (part of the Apache project) is the most commonly encountered, though OpenSTA has some users
- Commercial tools include MS VSTS, Load Runner, and many, many others
- Maintenance of performance tests is not as large an issue as for GUI functional tests
- Expertise with performance is critical, and trying to use these tools without such skills will result in misleading results
- Some previously open-source performance testing tools have gone away
- Correlating performance test results with performance modeling can increase confidence in both

Web Services/Service Virtualization Testing

- Parasoft and CA offer commercial tools, among many others
- SoapUI is the free tool most often mentioned
  - SoapUI, while workable, is not a very reliable tool
  - SoapUI is the open-source variant of a commercial offering (with similar problems)
- TestMaker and WebInject show up in web searches, but none of our clients have mentioned them
- These tools seem more limited in use, but that's often due to a matter of need rather than awareness

Dynamic Analysis

- Most operating systems have some built-in options (e.g., top, perfmon)
- Some compilers have the ability to build dynamic analysis code into programs
- Additional open-source tools are also available
  - Valgrind in Linux/Android/Mac world
  - Winleak for PCs
- Commercial tools such as Purify are also available
- Many of our clients under-utilize the options available here, especially given the serious reliability problems caused by runtime errors

Test Design Tools

- Certain types of tests require tools for design
- Pairwise testing is one good example
- PICT and ACTS are good free tools for pairwise testing, while various commercial pairwise tools also exist
- Model-based testing gets talked about a lot, but few of our clients use it
- We have built model-based test systems for some of our clients, using open-source tools
- Commercial test design tools often require particular formats for requirements, which are seldom available

Scripting Tools

- Many testers automate tests using scripting tools
- Ruby, Python, Tcl/Tk, and Unix shells are in common use
- If anything, there are too many different options, leading to Tower of Babel problem
- Scripting is best done by people with some programming skill and knowledge
- However, testers can learn or even teach themselves
- Care must be taken to avoid maintainability problems
- Very large, complex, sophisticated test systems can be built

Conclusions

- Test automation done right can provide solid benefits
- There are commercial tools and open-source tools available
- Free to download doesn't mean free to use
  - Direct costs of people's time
  - Opportunity costs (what could be done)
- Carefully scrutinize claims made by test tool vendors in terms of ease of use, ROI, etc.
- Building your own tools is an option, but the wealth of open-source tools makes this option less attractive
- Building integration and other "glue tools" does make sense

To Contact RBCS

For over twenty years, RBCS has delivered consulting, outsourcing and training services to clients, helping them with software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Phone:	+1 (830) 438-4830
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com
Twitter:	@RBCS, @LaikaTestDog
Facebook:	RBCS-Inc.