WHITE PAPER | SEPTEMBER 2016

Testing, Analytics and Decision Making?

In this series, Paul Gerrard, Principal of Gerrard Consulting, discusses some of the key industry testing topics. Here, Paul explores analytics and test reporting to support decision making. How can we quantify testing? It's easy enough to count the cost, but how much testing is enough? What is the value of testing anyway? These are difficult questions that testers have to answer.

Test Analytics is a discipline that covers the entire lifecycle from idea through development, production and eventual retirement, and moves beyond using just the output from test management tools.

Paul Gerrard

Gerrard Consulting

Sponsored by

technologies

Introduction

At the most fundamental level, the purpose of testing is to gather information to learn about some aspect of a system and potentially make a decision based on the outcome of one or more tests.[1] Testing provides the most valuable information required by developers (to fix defects), project managers (to understand and manage progress) and stakeholders (to be updated and assured). In this one respect, testing is all-powerful—it is the single source of knowledge of achievement in systems projects.

In the digital or mobile space, businesses are releasing apps that capture information about their users and the usage patterns of the apps themselves. Today's apps collect data in real time and send it back to the provider in approximately real time too. This data is analyzed to detect trends, patterns of behaviour, user preferences and opportunities for improvement or new market initiatives. The apps are instrumented to collect information for decision making.

Introducing Test Analytics

In my paper, "Thinking Big: Introducing Test Analytics,"[2] I make the case for looking at pre-release testing and production monitoring and analytics after release as two activities in a continuum. I use the term "Test Analytics" as the label for a discipline that spans the entire lifecycle, from idea through development, production and eventual retirement. A comprehensive review of mobile testing and analytics can be found in "The Mobile Analytics Playbook."[3]

I define Test Analytics as:

> *The capture, integration and analysis of test and production monitoring data to inform business and software development decision making*

Usually, testers have treated the output from test management tools as the main source of information for reporting—but this is far too limited a viewpoint. Counts of test cases, requirements covered (whatever that means), incident or bug reports raised, triaged, fixed or ignored, summarized and in-time series are interesting, but limiting too.

Modern Practices—Opportunities for Testing

My previous articles in this series shed some light on how we can improve our reporting and analyses. The thinking behind shift-left, test models, DevOps and automation and the emerging testing disciplines they encourage are forcing us to rethink what we mean by test reporting. The fundamental goal of testing—to collect and analyze data to inform decision making—is unchanged, but there are many opportunities for improving the way we report from testing:

1. The shift-left discipline aims to reduce, if not eliminate, misunderstandings in requirements early on. Late, costly requirements, defects and crises will not confuse our reporting.
2. The move to pervasive automation in DevOps regimes generates much of the data we need automatically. Results capture and analyses are no longer manual; reporting is almost instant.

3. “A New Model for Testing”[4] puts modeling at the heart of test design. When testers use meaningful models, they enable more meaningful coverage measurement.
4. Test Analytics can also be captured during exploratory sessions. Usage and system models can be created before test sessions as charters, while test coverage during sessions can be captured in-flight.
5. Testing in production is increasingly common. Why? No one can test all mobile device types (there are more than 24,000 devices in the Android ecosystem).[5] The value of production monitoring analytics is clear.
6. Monitoring, logging, alerting and analytics products are more widely available. Applications are instrumented as well as infrastructure. Scenarios that testers couldn’t try in the lab can be monitored (tested) in the field.
7. Some companies are abandoning bureaucratic incident management procedures. When defects are found, they are fixed.
8. Incidents are gone. The source code control tool, the automated unit tests that drive and protect change, provide the evidence and new visualizations of change and impact. See “History of Python” source code visualization for an example of what is possible.[6]

Because of these new practices, techniques, tools and visualizations, the nature of testing is changing. We have a real opportunity to improve our core product—the information we provide to stakeholders. But how does testing support decision making?

Testing and Decision Making

Over the years, I’ve been involved in many testing projects, large and small. In each one, the inevitable moment of decision comes at the end of a testing phase—what do we do next? Do we release to the next phase? Do we release to production or customers? Do we delay and extend the project? Do we pause and rethink our goals? Do we abandon hope and kill the project?

These are all critical-sounding decisions, but decisions are made at many levels. For example, when a tester finds a problem and discusses it with a user and the developer. The decisions come quickly: Is it a bug? Can it be fixed? Can it be fixed quickly? What is the impact of change? Is there a viable workaround? Is it worth fixing? Does the problem matter to the user at all?

Testing supports decision making at all levels. Testing stakeholders (executives, customers, users, project managers, operations and developers) can have many roles. So the information they require from testing varies in line with their perspective and their need to make decisions. There isn’t a magic formula for deciding what information is required—we have to ask our stakeholders what they would find most useful, valuable and economic.

But there’s a problem. How can we quantify testing? It’s easy enough to count the cost, but how much testing is enough? What is the value of testing anyway? These are difficult questions that testers have to answer.

I am a big fan of physics and for a bit of fun, I have used some scientific-sounding labels to name one principle and two theories.

Functional Testing at Scale

When we test the functionality of components higher in the architecture, particularly the integrator and application levels, we might have to simulate thousands or millions of devices in the field. The numbers of combinations and permutations may be beyond computation or prediction. Our simulations will repeatedly generate scenarios to be tested, record the outcomes and might replay the simulations for later study.

The higher-level components must be testable. We'll need facilities like exception handlers, plus utilities that inject data, capture and reproduce or replay scenarios. Cem Kaner has written quite a lot about what he calls "High Volume Automated Testing."[7] This is a good starting point.

These techniques could also be called big data testing. We'll need:

- To find data that fits our purpose
- To generate, tag, edit and seed data so we can trace its usage
- Tools to monitor the use of tagged data and the ability to reconcile data from collection, storage, use and disposal
- New test visualization tools to support diagnostics and debugging

This is a volumes game. Individual tests may or may not be important, but we'll spend a lot of time dealing with large-scale outcomes, visualizations and decision making.

The Testing Uncertainty Principle

If you've ever planned to do "enough" testing and predict a completion date for the testing, you'll recall how hard it is to derive a reliable estimate. Where you expect to find problems—and the biggest variable in planned test execution—is in how much testing will be blocked and how much re-testing will be required. But this depends mostly on how many bugs you encounter and how hard they are to fix and test. You won't know this until the bulk of your testing, fixing and re-testing are complete. It is yet another testing paradox.

The challenge of test prediction and planning is conveniently expressed in the Testing Uncertainty Principle:

- We can predict test status, but not when it will be achieved.
- We can predict when a test will end, but not its status.

You can define an exit or completion criteria (e.g., all tests run and passed), but you can never be sure when you'll reach that point until you reach it. How often do we set exit criteria and fail to meet them? We have to treat exit criteria as planning assumptions. If we don't meet the criteria, our assumptions are wrong, our plan is wrong and we need to re-plan or re-scope the project.

We can timebox the testing and guarantee a finish date. But who knows how much testing we can fit into the period and whether that is enough?

The Testing Theory of Relativity

The value of a test is affected by many things. A single test case might cover five lines of code in a component unit test. Another might exercise five million lines of code in a large system. Clearly the second test has more value. But who can say what that value is?

We have to ask a less-ambitious question. The tester might have a view, but we must rely on our stakeholders to judge value. They cannot put a value on any test, but they can usually say which of two or more tests is the most valuable. In this respect, we cannot assign an absolute value, but we can discern a relative value—one test is more or less valuable than another. This sounds like a big disadvantage, but it isn't really.

Most of the challenge of test planning comes down to scope. We know we cannot test everything, so we have to prioritise in some way. Knowing the relative value of tests means we should be able to select the more valuable ones and set the other less-valuable tests aside—for now.

Only the stakeholder(s) we test on behalf of can judge the value of tests.

But what about the risks of failure? Isn't the value of a test related in some way to the risks of the failure mode that the test is designed to cover? That may be so—an assessment of risk (quantified or not) is a consideration in the value of tests. But we still have a problem.

Given a valuable test, is a second test that does something similar just as valuable as the first? Not usually. To discuss that, we need some quantum theory.

The Quantum Theory of Testing

When we perform a test, we obtain some outcome and compare that outcome to some expectation.

- If the outcome matches expectations, it meets, in an incremental way, a user's need or requirement. The test incrementally increases our knowledge and confidence.
- If the outcome does not match expectations, it does not meet, in an incremental way, a user's need or requirement. The test will incrementally increase our knowledge but decrease our confidence (until the system is fixed and re-tested successfully).

Each test generates a discrete quantum of evidence. A yes/no, a true/false, a 1 or 0. When all tests are run, we aggregate these quanta of evidence and take a more rounded view of the situation. (By the way, this happens in the "apply test, interpret outcome" loop in "A New Model for Testing"[8])

The important thing is that when we run a test, it only has value if we learn something we didn't know before. If we learn little or nothing, the test has little or no value. A test that has value is significant in that it increases our knowledge about the system under test. When we need to judge whether two tests are better than one, we must look at the potential value of the second test and its significance.

A second test may be valuable on its own but matched with a near duplicate; its significance—and true value—is low. The significance of a test maps directly to the incremental increase in coverage that it provides.

The tester—or at least the person who designed the test model—is the best judge of whether a test is significant. For any measure of coverage to have meaning, it must always be measured with respect to some test model.

Summary

With that rather academic-sounding discussion behind us, what are we to conclude?

The emerging approaches of DevOps, continuous delivery and shift-left provide a great opportunity to reduce the problem of poor requirements, to move testing earlier and to find and fix problems with less drama. But the increased use of automation also means it will be easier than ever before to collect data on test outcomes. How do we take advantage of this opportunity?

We need to create test models that are meaningful to our stakeholders. In this way, the stakeholder can recognise the value of the tests we propose. As testers and modelers, we can advise on the significance of these tests—the coverage, if you like—with respect to the models themselves. In this way, we can obtain valuable tests and minimise duplication.

The discussion of value and significance of tests also helps us better gauge the value of our test automation. A difficulty I have seen arise in many test automation projects is where a complete system test is scoped for automation. When first run as functional tests, they had value. But where these tests are run repeatedly as regression tests, their significance (and therefore value) is much reduced.

The goal of what might be better called an anti-regression test[9] is to detect unwanted behaviour after changes are made. It might be that only 10 percent of your system tests are necessary to raise the alarm, so to speak. The other 90 percent cover the same ground and are not significant in this respect.

This article should help you have more informed discussions about decision making, and determine how much testing is enough and the value of what you do as testers. Relativity and quantum theory might be useful to you after all.

About the Author

Paul Gerrard is a consultant, teacher, author, webmaster, developer, tester, conference speaker, rowing coach and a publisher. He has conducted consulting assignments in all aspects of software testing and quality assurance, specialising in test assurance. He has presented keynote talks and tutorials at testing conferences across Europe, the USA, Australia, South Africa and occasionally won awards for them.

Educated at the universities of Oxford and Imperial College London, in 2010, Paul won the Eurostar European Testing Excellence Award and in 2013, won The European Software Testing Awards (TESTA) Lifetime Achievement Award.

In2002, Paul wrote, with Neil Thompson, "Risk-Based E-Business Testing". Paul wrote "The Tester's Pocketbook" in 2009. Paul co-authored with Susan Windsor "The Business Story Pocketbook" in 2011 and wrote "Lean Python" in 2014.

In 2014, Paul was the Programme Chair for the EuroSTAR Conference in Dublin.

He is Principal of Gerrard Consulting Limited, Director of TestOpera Limited and is the host of the Test Management Forum.

Mail: paul@gerrardconsulting.com
Twitter: @paul_gerrard
Web: gerrardconsulting.com

For more information visit **Develop & Test** with CA Technologies.

References

1 Paul Gerrard, "The Tester's Pocketbook," http://testers-pocketbook.com

2 Pau Gerrard, "Thinking Big: Introducing Test Analytics," Oct 2013, http://blog.gerrardconsulting.com/?q=node/630

3 Julian Harty, Antoine Aymer, "The Mobile Analytics Playbook," http://www.themobileanalyticsplaybook.com/

4 Cem Kaner, "The Insapience of Anti-Automationism," Feb 2013, http://context-driven-testing.com/?p=69

5 Paul Gerrard, "A New Model for Testing", 2012, http://gerrardconsulting.com/?q=taxonomy/term/281

6 Cory Goldberg, "History of Python," Gource development visualization, June 2012, https://www.youtube.com/watch?v=cNBtDstOTmA

7 Paul Gerrard, "Regression Testing—What to Automate and How," Jan 2010, http://gerrardconsulting.com/?q=node/547

8 Paul Gerrard, "A New Model for Testing", 2012, http://gerrardconsulting.com/?q=taxonomy/term/281

9 Cory Goldberg, "History of Python," Gource development visualization, June 2012, https://www.youtube.com/watch?v=cNBtDstOTmA

CS200-218519_0916