Things Could Get Worse: Ideas About Regression Testing

Michael Bolton

http://www.developsense.com

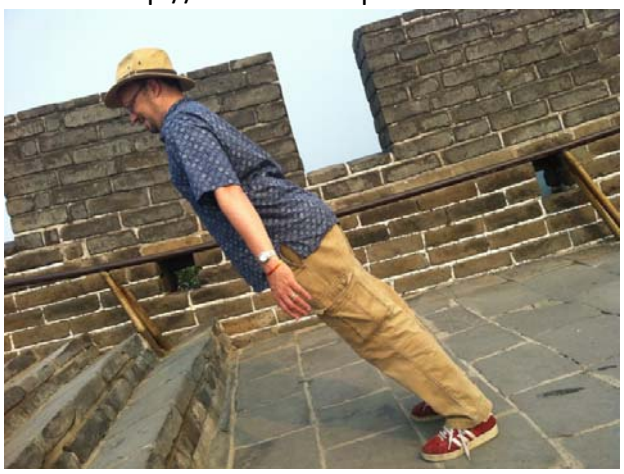
STAR East

May 2013

Michael Bolton

michael@developsense.com

http://www.developsense.com



Great Wall of China
August, 2012

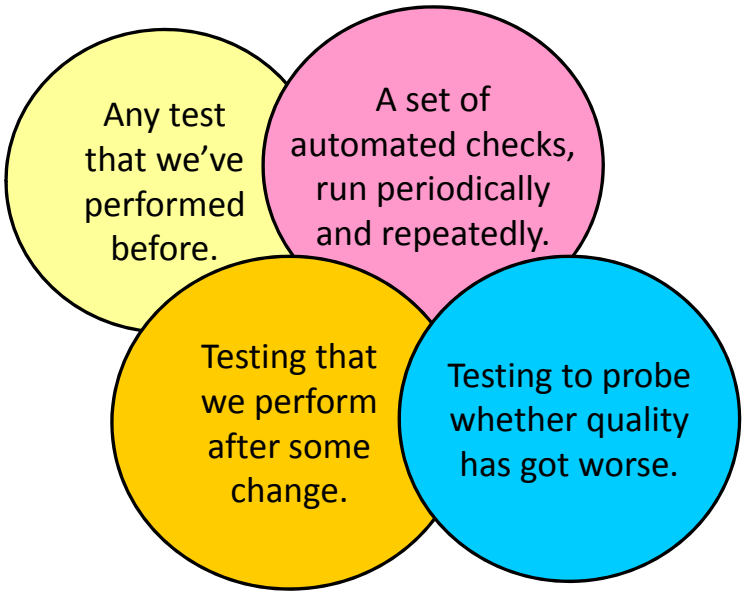
Do You See A Problem Here?

On starting Microsoft Office 2010 Business Contact Manager Setup Wizard

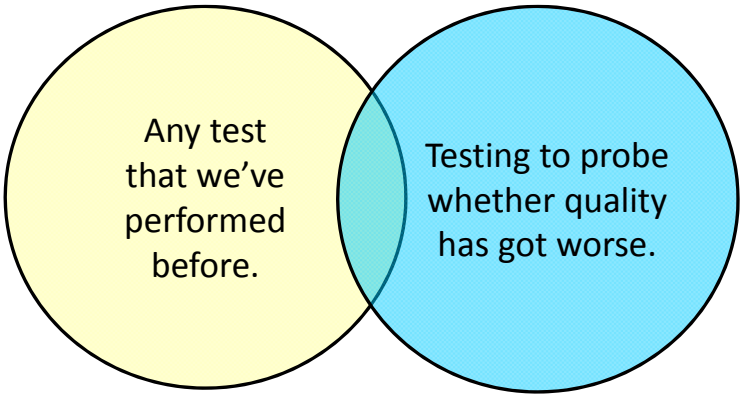
No regression test found this problem.
This is not a case of something getting worse,
but something that's bad to start with.

The Russian Optimist

Four Concepts of Regression Testing



Two Key Notions of Regression Testing



A *repeated* test
might not show that quality has got worse…

…and a test that shows that quality has got worse might be a *new* test.

A Regression Testing Hypothesis

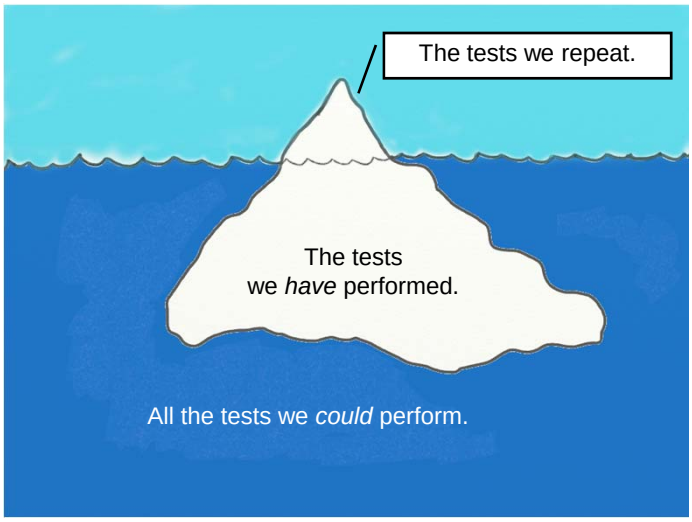
- Regression is an important product & project risk.
- Programmers are working so fast (especially in Agile), there's a danger of breaking stuff.
- Breaking stuff that worked before is embarrassing.
- Breaking stuff that was broken *and fixed* before is *really* embarrassing.
- THEREFORE we should focus the bulk of our attention on
  - tests that we've performed before
  - automated checks
  - testing after changes

What Wikipedia Says

- "The intent of regression testing is to ensure that a change, such as a bug fix, did not introduce new faults."
- "One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software."
- "Regression testing can be used to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change."

How Do You Cover an Ocean?

Testing is Always Sampling



"Any test that we've performed before"

- Is the primary purpose of testing to reveal new information about the product?
- If so, what is the information value of repeating a test that we've already performed?

When moving forward,
how much time should we spend looking behind us?

Checking vs. Testing

A CHECK is…

1. An *observation* linked to…
2. A *decision rule* such that…
3. both the observation and the decision rule can be applied *non-sapiently*

A ***non-sapient*** activity can be performed

by a machine
that *can't* think
(but that is quick and precise)

by a human who has been
instructed *not* to think
(and who is slow and variable)

Checks

Very good for

- Decidable propositions
- Anticipated risks
- Confirmation
- Verification
- Rapid execution
- Some bugs*
- Precision
- Functional correctness
- Low-level checking

Maybe not so good for

- Open-ended questions
- Unanticipated risks
- Exploration
- Qualitative evaluation
- Reflection
- Most issues*
- Nuance
- Parafunctional quality
- High-level testing

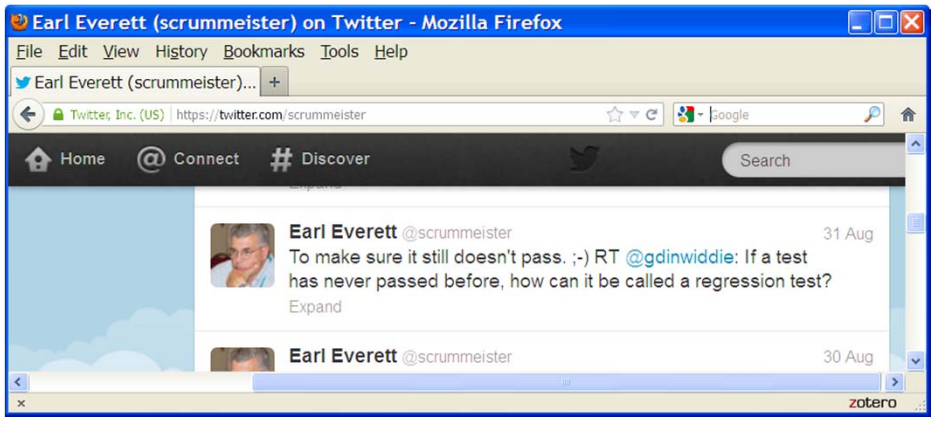
* A *bug* is a problem that threatens the value of the product.
An *issue* threatens the value of the testing, the project, or the business.

Pass or Fail?

- When a check "passes", it usually means "we (or the automation) didn't detect anything that might be a problem".
- When a check "fails", it usually means "we (or the automation) detected something that *might* be a problem".
- Instead of "pass or fail", skilled testers focus on a more fundamental question:

Is there a problem here?

Do Tests Really Pass or Fail?



Might it be better to ask,
"What can we learn from this test?
And *what else* can we learn?"

Insufficient Regression Check Focus

- may suppress or delay discovery of functional problems
- may suppress other kinds of variation
- may suppress learning about the API (when checks are performed without automation assistance)
- increases bug investigation and reporting time (when not done by programmers)
- increases the length of the feedback loop (when not done by programmers)

Excessive Regression Check Focus

- may suppress discovery of risks
  - especially when that focus is on the execution of checks.
- may suppress discovery of unchecked problems
- may suppress discovery of unnoticed value
- may suppress learning about the program generally
- may suppress learning about user's task
- may suppress certain kinds of variation

"Tests to probe whether quality has got worse."

Capability	✓
Reliability	✓
Usability	
Charisma	
Security	
Scalability	✓
Performance	✓
Installability	✓
Compatibility	
Supportability	
Testability	

✓ = Can be checked

✓ = Some checking helps

= Must be tested

What Else Could We Observe?

- Every test is multivariate
  - though not every *check* is multivariate
- A focus on a single oracle can distract us from other things, so…
- If you're going to check, try checking several things at once. Gather lots of data!
- Look at the data! Take advantage of human pattern recognition.

Is Regression The Biggest Risk?

- Before the Agile Manifesto was declared, a group of experienced test managers reported that regression problems ran from 6-15% of discovered problems
- In Agile shops, we now (supposedly) have
  - TDD
  - unit tests
  - pairing
  - configuration management
  - build and version control
  - continuous integration
- …many of which *already check for regression*
- How serious a risk is regression given these things?
- Are we paying attention to what regression tells us?

Maybe Regression Problems Are *Symptoms*

- If you see a consistent pattern of regression
  - maybe failing checks or tests aren't your biggest problem
  - more likely, the issue is that you've got favourable conditions for regression to happen
  - this is a potentially serious risk that testing can only *detect*, and that more testing *cannot fix. THAT'S A SEVERITY-ONE ISSUE.*

Maybe FEAR of regression is a symptom, too.

- Pay attention to feelings, worries, obsessions.
- Feelings are *signals*; what are they signaling?
- What are the problems *behind* the fear?

The Goalie Problem

Even on a great team, great goaltending will miss *some* shots.

The Goalie Problem

Not even great goaltending can make up for a weak defense.

Dimensions of Testing Costs

$ Design time
$ Development cost
$ Maintenance cost
$ Transfer cost
$ Equipment cost
$ Setup cost
$ Execution cost
$ Opportunity cost

How do these apply to checks? To other kinds of testing?

To test is to compose, edit, narrate, and justify THREE stories.

A story about the status of the PRODUCT…

…about how it failed, and how it *might* fail...
…in ways that matter to your various clients.

A story about HOW YOU TESTED it…

…how you configured, operated and observed it…
…what you haven't tested yet, but what you might test…
…and what won't test, at all, unless things change.

A story about HOW GOOD that testing was…

…what the risks and costs of testing are…
…what made testing harder or slower…
…how testable (or not) the product is…
…what you need and what you recommend.

26

Reasons to Repeat

- Heuristics from Rapid Software Testing
  - Recharge (renewed relevance when things change)
  - Intermittency (some tests take repetition to reveal bugs)
  - Retry (human observers see things differently each time)
  - Mutation (you can change something about the test)
  - Benchmark (tracking progress requires repetition)
  - Importance (the test is too critical to put aside)
  - Mandate (sometimes we do exactly what we're told)
  - Inexpensiveness (why not run a test if it's cheap?)
  - Enoughness (some suites might give adequate coverage)
  - Avoidance/indifference (you might repeat a test when testing isn't really the goal)

http://www.satisfice.com/repeatable.shtml

Heuristics for Regression Testing

- Heuristics from Karen Johnson: RCRCRC
  - Recent (what has changed lately?)
  - Core (what's really common and critical?)
  - Risk (where is there danger, complexity, brittleness?)
  - Configuration-sensitive (what's variable and vulnerable?)
  - Repaired (did it *really* get fixed? did it break something else?)
  - Chronic (what are the patterns in collections of problems?)
    http://searchsoftwarequality.techtarget.com/tip/A-software-experts-heuristic-for-regression-testing
- We could add even more Rs and Cs!
  - Randomized (high-volume, high-speed automated checks)
    - but beware the oracle problem
  - Confusing (poorly-understood or poorly-agreed-upon areas)
  - Rich (complex scenarios to shake out lots of problems)
  - Compelling (bugs that spark newspaper stories)

Regression Automation Heuristics

- Heuristics from Mike Kelly
  - To recreate the path to a specific bug
  - To capture current functionality to address potential breakage problems
  - To automate checks in areas that are so important they must be checked
  - To address code coverage gaps
  - To addressing requirements coverage gaps
  - To address a particular model of the problem space (e.g. all-pairs coverage)

http://www.michaeldkelly.com (look for it under "media")
OR
http://bit.ly/Ugdge8

What Wikipedia Says

REVISED

- "The intent of regression testing is to ensure that a change, such as a bug fix, did not introduce new faults"
- AND ALSO to investigate whether the fix addressed the general instance of the problem, or just the specific
- AND ALSO to investigate whether there were faults we missed in that area

What Wikipedia Says

REVISED

- "One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software."
- AND ALSO to learn about the relationships between parts of the software, to see where future changes might have other affects
- AND ALSO to see how changes in other things affect the system

What Wikipedia Says

REVISED

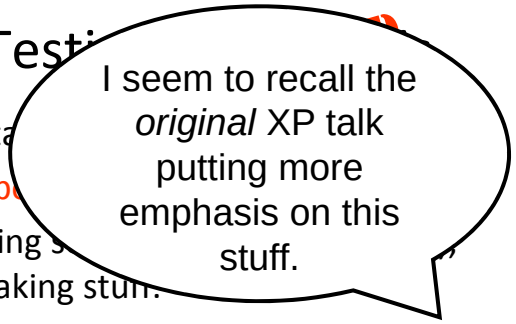
- "Regression testing can be used to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change."
- AND ALSO to question the notion of "the appropriate minimum number of tests"
- AND ALSO, especially in design, to question our notion of "adequate coverage".
- AND ALSO to increase dramatically the ways in which we could observe the product
- But *how* do you "systematically select"?

A Regression Testing Hypothesis

REVISED

- Regression is an important product & project risk
- *and* there are other important risks.
- Programmers are working so fast (especially in Agile), there's a danger of breaking stuff.
- Maybe they should maintain a sustainable pace
- Maybe they should work more carefully.
- Maybe they should work more collaboratively.
- Maybe the stuff they're working on should be less interdependent.
  - NOTE: testers can identify project risks like these but be careful: we don't manage the programmers or the project

A Regression Testi



- Regression is an importa
- and there are other imp
- Programmers are working s
  there's a danger of breaking stuff.
- Maybe they should maintain a sustainable pace
- Maybe they should work more carefully.
- Maybe they should work more collaboratively.
- Maybe the stuff they're working on should be less interdependent.
  - NOTE: testers can identify project risks like these but be careful: we don't manage the programmers or the project

A Regression Testing Hypothesis

REVISED

- Breaking stuff that worked before is embarrassing.
- Breaking stuff that was broken *and fixed* before is *really* embarrassing.
- and having bugs that we never noticed at all is *also* embarrassing
- THEREFORE we should focus attention on
  - tests that we've performed before
  - automated checks
  - testing after changes
- AND ALSO ON
  - performing *new* testing that extends coverage, giving
  - a more comprehensive understanding of the product and
  - a reasonable chance of identifying regression problems

Contrasting (and Complementary) Test Strategy Ideas

checking	⇨	testing
passing checks	⇨	checks that trigger questions
large number of checks	⇨	increasing test coverage
noting regression problems	⇨	noting problems generally
detecting regression	⇨	preventing regression
preventing regression	⇨	preventing problems
quicker development	⇨	more reliable development
making sure that our checks are passing	⇨	making sure that we understand the product
testing to search for bugs we anticipate	⇨	testing to learn about the product and new risk ideas

Ancient Wisdom

One of the lessons to be learned … is that the sheer number of tests performed is of little significance in itself. Too often, the series of tests simply proves how good the computer is at doing the same things with different numbers. As in many instances, we are probably misled here by our experiences with people, whose inherent reliability on repetitive work is at best variable. With a computer program, however, the greater problem is to prove adaptability, something which is not trivial in human functions either. Consequently we must be sure that each test does some work not done by previous tests. To do this, we must struggle to develop a suspicious nature as well as a lively imagination."

Herbert Leeds and Gerald M. Weinberg, *Computer Programming Fundamentals*, 1961

Final Ideas

- Sort out the difference between *regression* and *repetition*.
- Regression testing is an airbag that will *sometimes* protect you. It's not a substitute for good driving.
- Examine the motives for regression testing.
- Examine the causes of regression.
- Ask how we might detect and prevent regression more quickly, less expensively, more reliably.
  - checks can be good, and there are other ways
- Consider what your testing is telling you
  - and *tell that story*.
- Think of checking as *an element* of testing.
- Think cost vs. value.
- Focus on adaptability, not just repeatability.

Questions?

Michael Bolton
michael@developsense.com
http://www.developsense.com
Twitter: @michaelbolton
Don’t bother with Facebook.
Or LinkedIn.