Testers: Get Out of the Quality Assurance Business!

Michael Bolton
DevelopSense
http://www.developsense.com
SQA Days
St. Petersburg, November 2010

I’m Michael Bolton

Not the singer. Not the guy in Office Space. No relation.

Updates

• This presentation is ALWAYS under construction
• Updated slides at http://www.developsense.com/past.html
• All material comes with lifetime free technical support

Let’s Start With a Simple Question:

What is “quality”?

The Quality Answer

• Quality is “value to some person(s)”
  – Jerry Weinberg
• (with respect to testing) “…who matter.”
  – James Bach and Michael Bolton
• Decisions about quality are always political and emotional
  – made by people with the power to make them
  – made with the desire to appear rational
  – yet ultimately based on how those people feel

If you’re a tester, do you…

design the product?
write the code?
hire the programmers?
decide which bugs to fix?
allocate staff?
set the schedule?
fix problems in the code?
decide on raises?
allocate training budgets?
produce manuals?
choose the development model?
fire some programmers?
control the budget?
set the company’s strategic direction?
How Can You, Tester, Assure Quality?

**YOU CAN’T.**

But not to worry. That’s not the tester’s job.

We Can’t Assure Quality

but we can TEST.

A Computer Program

A set of instructions for a computer.

See the Association for Software Testing’s Black Box Software Testing Foundations course, Kaner & Bach

A House

A set of building materials, arranged in the “House” design pattern.

A House

Something for people to live in.
Kaner’s Definition of a Computer Program

- A computer program is a **communication** among several people and computers separated over distance and time that contains instructions that can be run on a computer.

The purpose of a computer program is to provide **value** to **people**.

Implications of Kaner’s Definition

- A computer program is **far more** than its code
- A software product is **far more** than the instructions for the device
- Quality is **far more** than the absence of errors in the code.
- Testing is **far more** than writing code to assert that other code returns some “correct” result
- Testing is **not about** “writing test cases”.

Quality is value to some person(s).

Testing is an investigation of code, systems, people, and the relationships between them.

What Is Testing?

Software testing is the investigation of systems composed of people, computer programs, and related products and services.

- Excellent testing is not a branch of computer science—focus only on program code and functions, and you leave out questions of value and other relationships that include people
- To me, excellent testing is more like **anthropology**—highly multidisciplinary—doesn’t look at a single part of the system
- Anthropologists investigate many things—biology (human mechanisms; human “code” and “hardware”)—archaeology (human history)—linguistics (human communication)—cultures (what it means to be human)

So What Is Testing?

- “Questioning a product in order to evaluate it” – James Bach
- “Gathering information with the intention of informing a decision.” – Jerry Weinberg
- “A technical, empirical investigation of a product, done on behalf of stakeholders, with the intention of revealing quality-related information of the kind that they seek.” – Cem Kaner

No assurances!

Testing Is More Than **Checking**

- Checking is a process of confirming and verifying existing beliefs
  - Checking can (and I argue, largely should) be done by automation
  - It is a **non-sapient** process

Oh no! What Does “Non-Sapient” Mean?

- A **non-sapient** activity can be performed
  - by a machine that can’t think (but it’s quick and precise)
  - by a human who has been instructed NOT to think (and that’s slow and erratic)

What Is Sapience?

- A **sapient** activity is one that requires a thinking human to perform
- We test not only for repeatability, but also for adaptability, value, and threats to value

This kind of testing **CANNOT** be scripted

But...

- A good tester doesn’t just ask **Pass or Fail?**
- A good tester asks **Is there a problem here?**

Besides...

- Automation cannot
  - program a script
  - investigate a problem you’ve found
  - determine the meaning or significance of a problem
  - decide that there’s a problem with a script
  - escape a script problem you’ve identified
  - determine the best way to phrase a report
  - unravel a puzzling situation

But automation **CAN** help YOU do those things.

Is Regression Your 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
- Is regression a serious risk?
- If so, can testing (whether automated or not) fix it?
- Is regression really a symptom of problems elsewhere?

Regression Problems Are Symptoms

- If you see a consistent pattern of regression
  - the failing tests are not the organization’s biggest problem
  - you might want to raise awareness that there’s a favourable environment for regression
Testing Is More Than Checking

• Testing is an ongoing, continuously re-optimizing process of

exploration, discovery, investigation, and learning

Testing is Exploring

• Our community* sees testing as exploration, discovery, investigation, and learning
  – Testing can be assisted by machines, but can’t be done by machines alone
  – Testing is a sapient process

I can’t test, but I can help you act on test ideas.

What IS Exploratory Testing?

• I follow (and to some degree contributed to) Kaner’s definition, which was refined over several peer conferences through 2007:

Exploratory software testing is…
• a style of software testing
• that emphasizes the personal freedom and responsibility
• of the individual tester
• to continually optimize the value of his or her work
• by treating test design, test execution, test result interpretation, and test-related learning
• as mutually supportive activities
• that run in parallel
• throughout the project.

Whew. Maybe it would be a good idea to keep it brief most of the time…

“Parallel test design, test execution, and learning.”

See Kaner, “Exploratory Testing After 23 Years”,
www.kaner.com/pdfs/ETat23.pdf

Irony Alert!

• We talk about checking with test cases
• We often manage testing with checklists

Oh well!
Smart people can deal with stuff like this.

So What Are We Testers?

Skilled investigators

We Are Sensory Instruments

The tester doesn’t have to reach conclusions or make recommendations about how the product should work. Her task is to expose credible concerns to the stakeholders.
  - Cem Kaner, Approaches to Test Automation, 2009 (my emphases)
Software Development Is Not Much Like Manufacturing

- In manufacturing, the goal is to make zillions of widgets all the same.
- Repetitive checking makes sense for manufacturing, but...
- In software, creating zillions of identical copies is not the big issue.
- If there is a large-scale production parallel, it's with design.

Software Development Is More Like Design

- If existing products sufficed, we wouldn't create a new one, thus...
- Each new software product is novel to some degree, and that means a new set of relationships and designs every time.
- New designs cannot be checked only; they must be tested.

Testing of Design Is Like CSI

- There are many tools, procedures, sources of evidence.
- Tools and procedures don't define an investigation or its goals.
- There is too much evidence to test anything like all of it
- Tools are often expensive
- Investigators are working under conditions of uncertainty and extreme time pressure
- Our clients (not we) make the decisions about how to proceed based on the available evidence

These ideas come largely from Cem Kaner, Software Testing as a Social Science: http://www.kaner.com/ftp/Kaner/SocialScienceSTEP.pdf

Testing of Design Is Like CSI

Viewing Testing as a Service Solves Many Problems

- When testing is a investigative service, we have exactly as much time as the client is willing to give.

Viewing Testing as a Service Solves Many Problems

- The problem is not that testing is the bottleneck. The problem is that you don’t know what’s in the bottle.
- That’s a problem that testing addresses.
More Relevant Comparisons
• Investigative reporters and journalists
  – What’s actually going on? What’s the story?
• Anthropologists
  – What do people in the real world actually do?
• Historians
  – What can we learn from the past?
• Field botanists
  – Why does this thrive over here, but not over there?
• Philosophers
  – What do we know? How do we know we know it?
• Film critics
  – Will this movie appeal to its intended audience?

Can’t We Help With Quality Tasks?
• Sure; (to me, at least) development teams should be autonomous and self-organizing
  – when you’re providing other services to your team, that might be good and very useful.
  – but that could be a problem if you’re not also testing.
• To the extent that your work is
  – requested by your colleagues
  – appreciated by your colleagues
  – not busy work
  – not busybody work
  …rock on! Help out! But also test.

Where Do We Go From Here?

We must build knowledge and skills

What Skills and Knowledge?
• Critical thinking
• General systems thinking
• Design of experiments
• Visualization and data presentation
• Observation
• Reporting
• Rapid learning
• Programming

Not a comprehensive list!

What is test framing?

Test framing is the set of logical connections that structure and inform a test.

Not a comprehensive list!
Framing ~= Traceability

- Framing is, in essence, traceability…
- …but typically we hear people talk of traceability in an impoverished way: between tests and requirements documents
- Can you demonstrate traceability between tests and implicit requirements?
- Can you demonstrate traceability between the test result and the mission?

Yes, Exploratory Testing Requires Skill

- Doesn’t ANY testing (worth doing) require skill?
  
  Well, we wanted to go with a skilled pilot...
  
  But they’re just so darned expensive...
  
  The value of test information is directly related to the skill of the tester.
  
  Hire (or train) testers with the skills to provide you with the information you seek.

A Heuristic Test Strategy Model

We’re not here to enforce The Law.

We are neither judge nor jury.

We’re here to add value, not collect taxes.
We’re here to be a service to the project, not an obstacle.

Book References: Cem Kaner
- The Ongoing Revolution in Software Testing
- Software Testing as a Social Science
  - http://www.kaner.com/pdfs/KanerSocialScienceSTEP.pdf
- Software Engineering Metrics: What Do They Measure and How Do We Know? (with Walter P. Bond)
- Approaches to Test Automation
- Lessons Learned in Software Testing
  - Kaner, Bach, & Pettichord

Book References: Jerry Weinberg
- Perfect Software and Other Illusions About Testing
- Quality Software Management
  - Volume 1: Systems Thinking
  - Volume 2: First Order Measurement
- Quality Software Management: Requirements Before Design
- An Introduction to General Systems Thinking
- The Psychology of Computer Programming
  - Jerry Weinberg

Book References
- The Black Swan
- Fooled by Randomness
  - Nassim Nicholas Taleb
- Secrets of a Buccaneer Scholar
  - James Bach
- Everyday Scripting in Ruby
  - Brian Marick
- How To Program
  - Chris Pine
- Sciences of the Artificial
  - Herbert Simon
- How Doctors Think
  - Jerome Groopman

What About ATDD? (Acceptance Test Driven Development)
- Acceptance tests are Examples
- Examples are NOT tests.
- Experiment is NOT experience.
- When an acceptance test passes, it means that the product appears to meet
  - some requirement
  - to some degree
  - in some circumstance
  - at least once
  - on my machine

Book References
- Blink
  - Malcolm Gladwell
- Tools of Critical Thinking
  - David Levy
- Mistakes Were Made (But Not By Me)
  - Carol Tavris and Elliot Aronson
- How to Lie With Statistics
  - Darrell Huff
- The Visual Display of Quantitative Information
- Envisioning Information
- Visual Explanations
- Beautiful Evidence
  - Edward Tufte
Some Problems With Acceptance Tests

- They're set at the beginning of an iteration or development cycle, when we know less about the product than we'll eventually.
- Acceptance tests are examples. They do not (and cannot) cover everything that might be important.
- Acceptance tests are checks, not tests.
- Talk about acceptance tests tends to leave out questions of who is accepting what, and for what purpose.
- Properly viewed, they should prompt rejection for failing, rather than acceptance for passing.
- They should be called rejection checks.

What About Regression Tests?

There appears to be a presumption in many Agile shops that regression tests
1. are intrinsically repeated tests
2. must be repeated in full on each build
3. must be automated
4. are essential to handle regression problems

The “Regression = Repeated” Problem

Two definitions:
1. Any repeated test.
2. Any test intended to show that quality hasn’t worsened.

Yet…

- a repeated test might not show that quality hasn’t worsened, even if it has
- a test that shows quality has worsened might be a new test

The Repeat-Them-In-Full Problem

- Automating regression checks makes execution fast and cheap, BUT…
- Development, testing, and maintenance of high-level automated checks is hard and expensive
- A test declines in value as its capacity to reveal new information diminishes
- High-level checks may not be very risk-focused
- High-level checks may be unnecessary when there are plenty of (cheap!) low-level checks

The Must-Be-Automated Problem

- Automation gives us tremendous gains in execution speed at the cost of loss of some opportunities to observe
- As automation gets higher-level, it tends to be
  - more complex
  - more expensive
  - less representative of most real-world behaviour
    - which may be a good or a bad thing
  - less aligned with things that make automation most useful

Unhelpful Ideas, Past Their Sell-By Date

See James Bach, “Manual Tests Cannot Be Automated”
http://www.satisfice.com/blog/archives/58
“Automated” vs. “Manual” Tests

- “Manual” refers to the wrong body part
  – it’s the brain, not the hands that do the work
- A good manual test cannot be automated
  – if you think it can, it wasn’t a good manual test
- Automated tests cannot be done manually
  – see http://www.satisfice.com/blog/archives/58
  – see http://www.satisfice.com/blog/archives/99
- What about automation-assisted tests?

More Unhelpful Ideas

- “Developers” vs. “Testers”
  – we’re all developers; if anything, it’s “programmers”
- “Automated testers” vs. “manual testers”
  – consider the “toolsmith” specialty instead
- “Quality assurance”
  – testers don’t assure quality

More Unhelpful Ideas

- Counting test cases
  – a test case is a container for an idea
  – do you measure your productivity in briefcases?
  – the number of test cases is of little interest in itself
  – see “The Case Against Test Cases”
- Defect escape ratios
  – since testers don’t decide to ship the product, “defect escape ratios” are measures of product management, rather than of testing on its own

More Unhelpful Ideas

- Passing test cases
  – when a test passes, there may still be terrible problems for which you are not applying an oracle
  – when a test case fails, there’s a story; what is it?
- Pass/fail ratios
  – a passing test case is a hope fulfilled
  – a failing test case is a rumour of a problem
  – the pass/fail unit is therefore hopes/rumours
    – is this a valid basis for measurement?