Testers: Get Out of the Quality Assurance Business!

Michael Bolton
DevelopSense
http://www.developsense.com

STP Conference

Las Vegas, October 2010

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
- “…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*

Do you…

design the product?

negotiate customer contracts?

write the code?

hire the programmers?

decide which bugs to fix?

allocate staff?

set the schedule?

set the product scope?

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?

No?

Then how, exactly, do you ASSURE quality?

How Can You, Tester, Assure Quality?

YOU CAN'T.

But not to worry.
That's not our job.

We Can't Assure Quality

but we can

TEST.

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 Isn't Just *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

See http://www.developsense.com/2009/08/testing-vs-checking.html

Oh no! What Does "*Non-Sapient*" Mean?

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

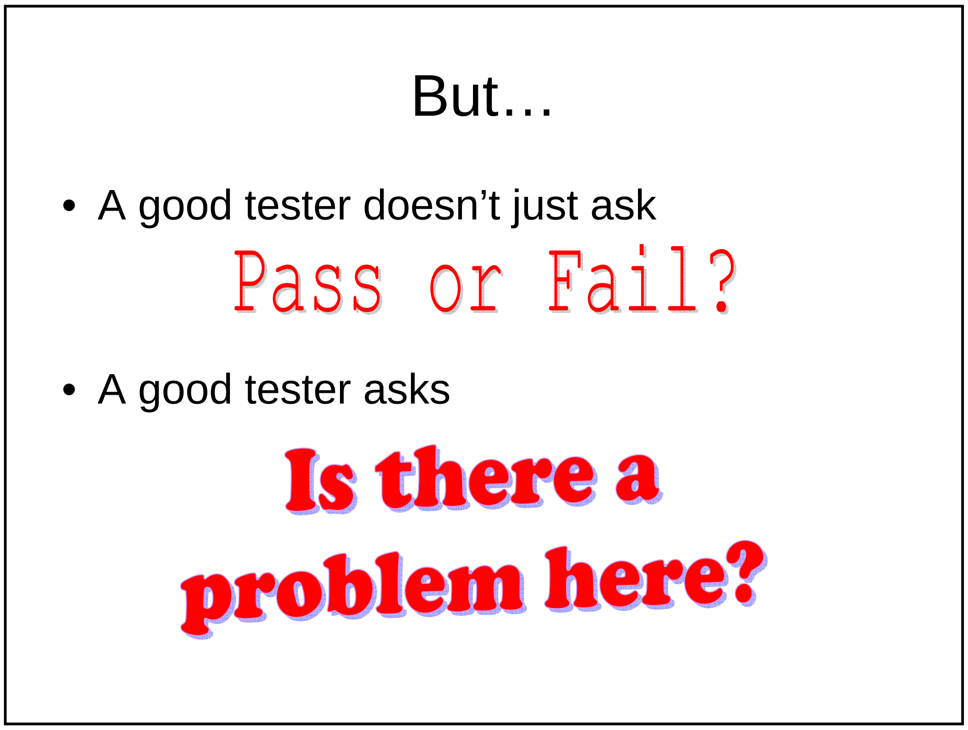
by a machine
that *can't* think
(but it's quick and precise)

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
CAN NOT
be scripted



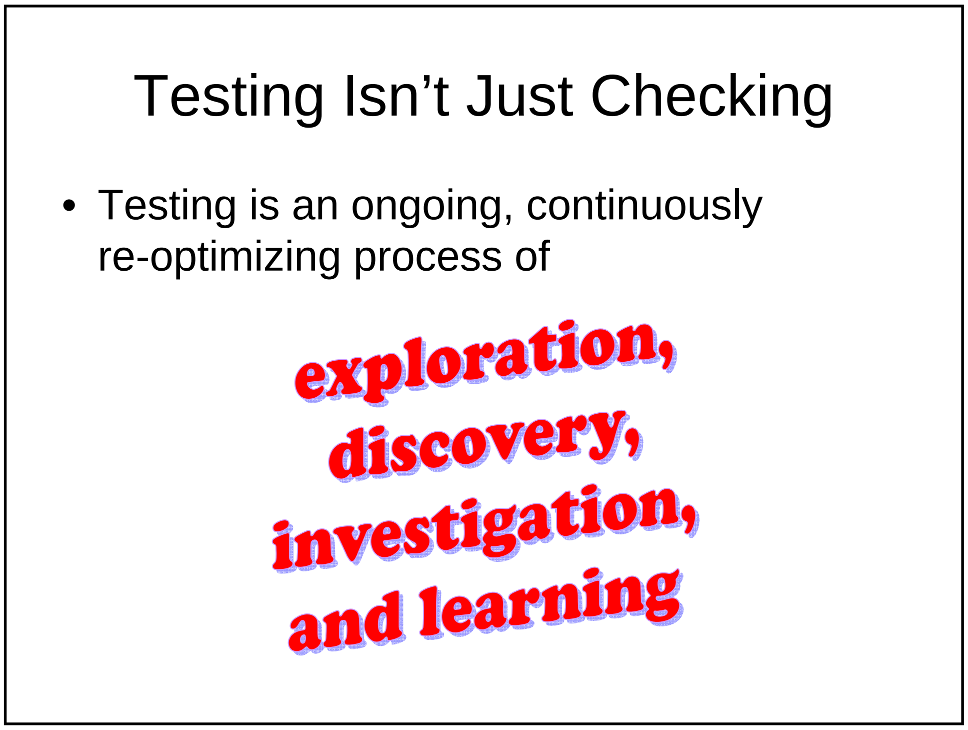
But…

- A good tester doesn’t just ask

Pass or Fail?

- A good tester asks

Is there a problem here?



Testing Isn’t Just Checking

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

exploration, discovery, investigation, and learning

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.

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 programs, and you leave out questions of *value* and other relationships that include people
- To me, excellent testing is like *anthropology*
  - highly multidisciplinary
  - doesn't look at a single part of the system
- Anthropology focuses on investigating
  - biology
  - archaeology
  - linguistics
  - cultures

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.

So What Are We Testers?

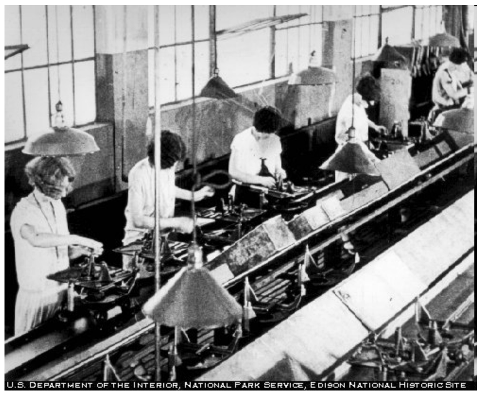
Skilled investigators

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)

We Are Sensory Instruments

Software Development Is Not Much Like Manufacturing


U.S. Department of the Interior, National Park Service, Edison National Historic Site

- 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

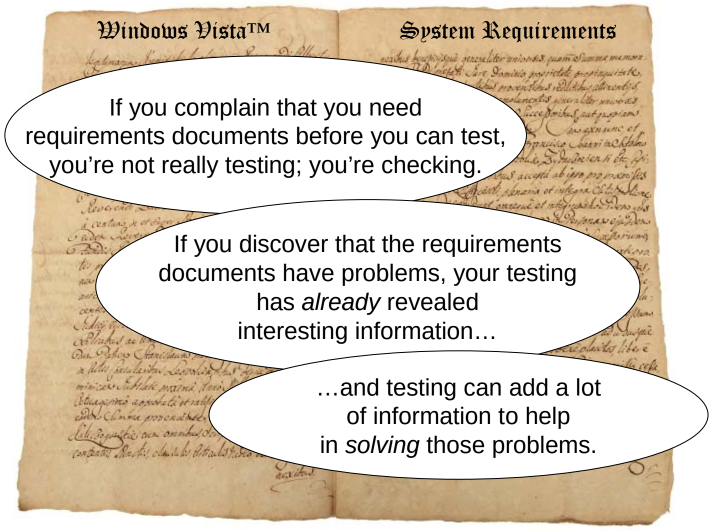
- There are many tools, procedures, sources of evidence.
- Tools and procedures don't *define* an investigation or its goals.
- There is too much evidence to test anything like all of it
- Tools are often expensive
- Investigators are working under conditions of uncertainty and extreme time pressure
- 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/pdfs/KanerSocialScienceSTEP.pdf

Viewing Testing as a Service Solves Many Problems

When are we going to be done eating?
What the…?
When testing is an investigative *service*, we have exactly as much time as the *client* is willing to give.

Viewing Testing as a Service Solves Many Problems


Windows Vista™
System Requirements
If you complain that you need requirements documents before you can test, you're not really testing; you're checking.
If you discover that the requirements documents have problems, your testing has *already* revealed interesting information…
…and testing can add a lot of information to help in *solving* those problems.

Other 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?

How Did We Get Here?

- “Managers asked me a simple question: ‘is it good enough to go live?’ When I answered that question “yes” or “no”, I gave my personal opinion about quality.
- “To my managers I had become an oracle*–like all oracles a fallible one. I didn’t have all the information. I didn’t know the whole context. And I surely didn’t test every possible situation in the product (which even is impossible).
- “However, my managers didn’t acknowledge my opinion as an oracle. As they knew me and my professionalism for a long time they accepted my comments as *factual*.”

– *Michel Kraaij, Software Tester*

An oracle is a fallible means or method of solving a problem or making a decision.

Testers provide *technical* information, but shipping decisions are *business* decisions.

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…
  - but you’re not *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

What Skills and Knowledge?

- Measurement
- Anthropology
- Teaching
- Risk analysis
- Cognitive psychology
- Economics
- Epistemology

References: Cem Kaner

- The Ongoing Revolution in Software Testing
  - http://www.kaner.com/pdfs/TheOngoingRevolution.pdf
- 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)
  - www.kaner.com/pdfs/metrics2004.pdf
- Approaches to Test Automation
  - http://www.kaner.com/pdfs/kanerRIM2009.pdf
- Lessons Learned in Software Testing
  - Kaner, Bach, & Pettichord

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

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

References

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