How to Get What You Really Want from Testing

Michael Bolton
DevelopSense
http://www.developsense.com
@michaelbolton
michael@developsense.com

I'm Michael Bolton

Not the singer.

Not the guy in Office Space.

No relation.

Updates

- This presentation is ALWAYS under construction
- Slides for this presentation available on request.
- All material comes with lifetime free technical support

A Clarification

- "How You Can Get The Best Results From Testing" is a potentially misleading title.
- As a member of the Context-Driven School of Software Testing, I maintain there are no *universally* best results available from testing, just as there are no best practices.
- In this presentation, I'll try to help introduce ideas that will allow you to choose or assign testing missions that line up with *what managers and other testing clients need to know*—how to decide what's best *for you*.

I'm Biased Towards Rapid Testing

Rapid testing is a *mind-set*
and a *skill-set* **of testing**
focused on how to do testing
more quickly,
less expensively,
with *excellent results*.

This is a general testing methodology. It adapts to any kind of project or product.

Read more about Rapid Testing at http://www.developsense.com

A Computer Program

```
A set of instructions
for a computer.
```

See the Association for Software Testing's
Black Box Software Testing Foundations course, Cem Kaner & James 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.

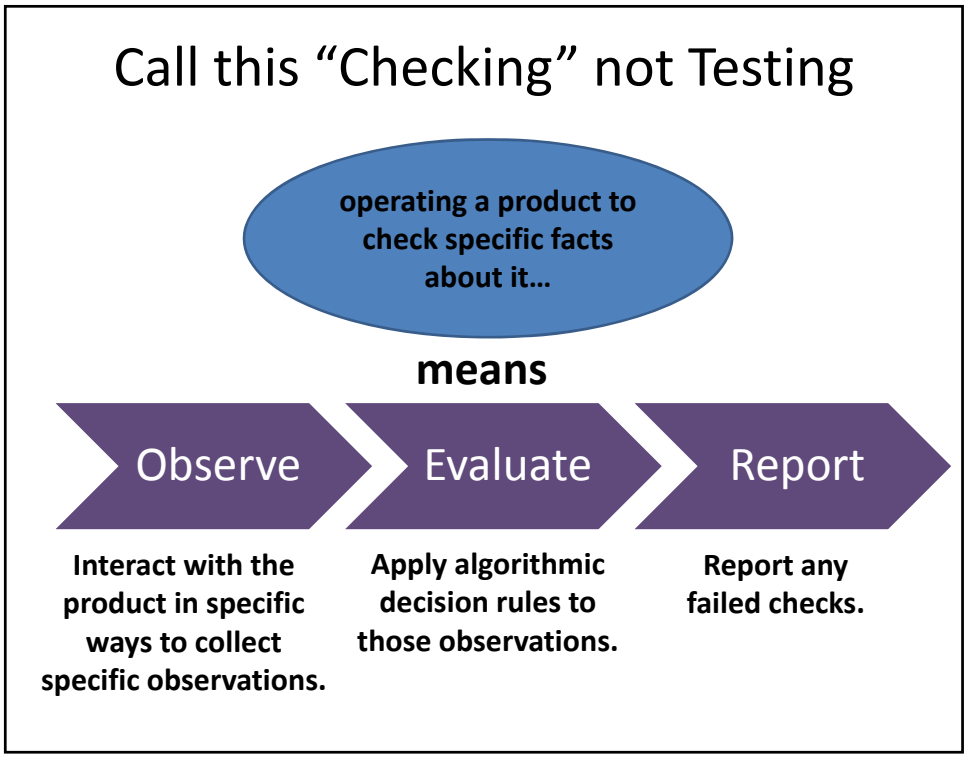
The purpose of a computer program is
to provide **value** to **people**

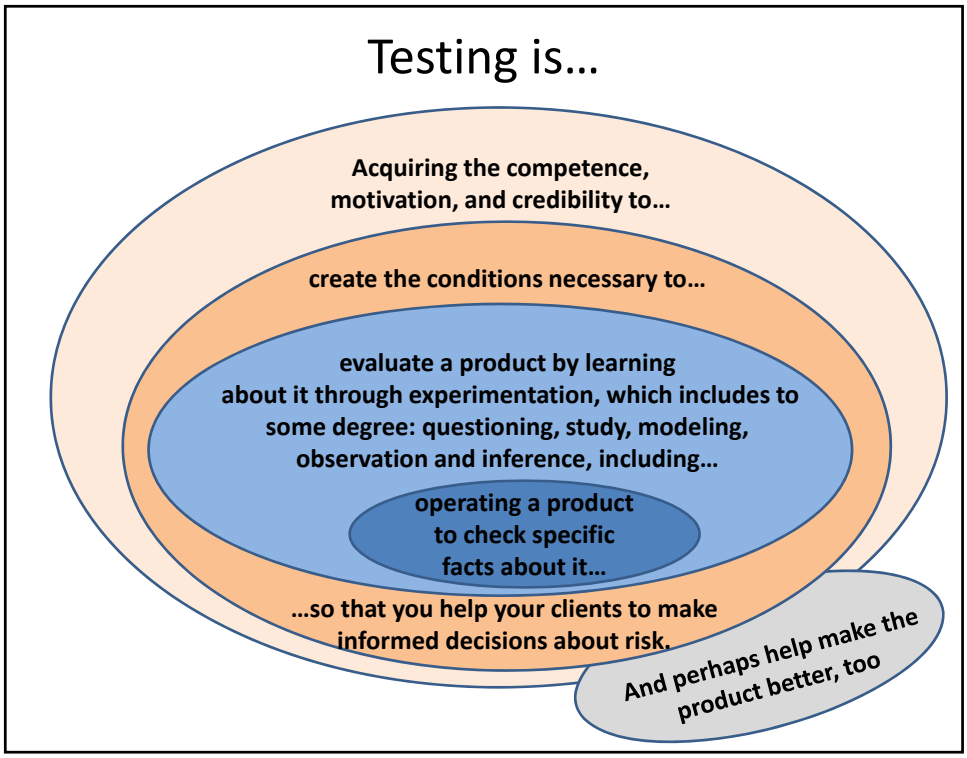
Implications of Kaner's Definition

- 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 some code to confirm that other code returns a "correct" result.

Quality is value to some person(s).
—Jerry Weinberg

Software testing is the investigation of *systems* consisting of people and their work, computers, programs, and the relationships between them.





A Check Has Three Elements

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

A ***check*** can be performed

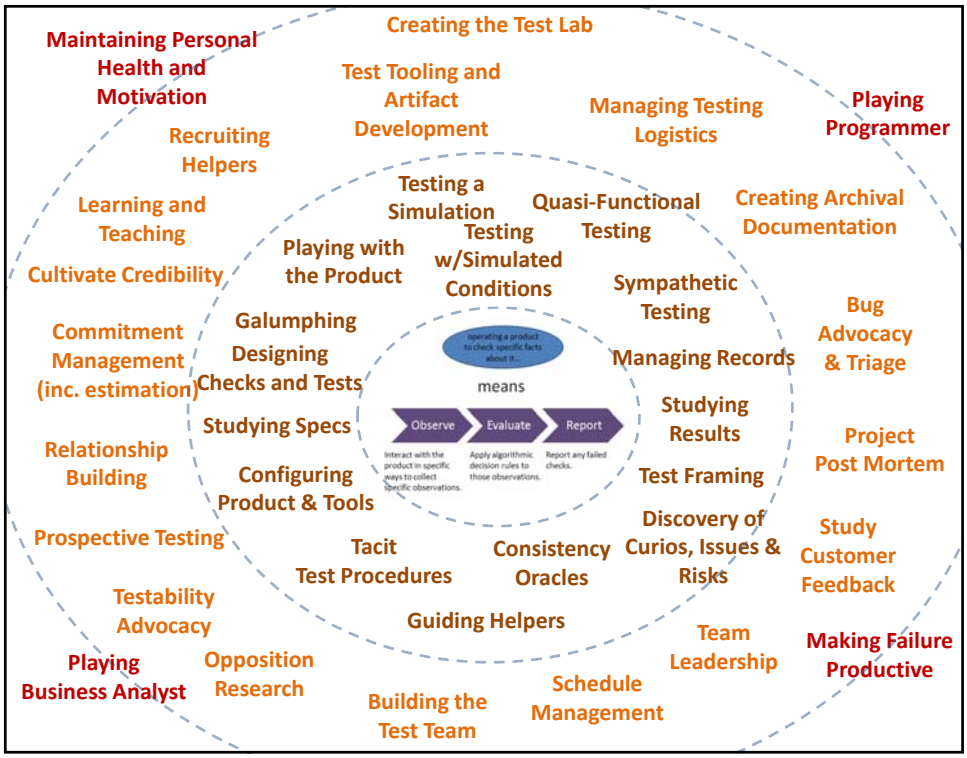
by a machine
that *can't* think
(but that is quick and precise)

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

Testing Is More Than *Checking*

- *Checking* is a process of confirming and verifying existing beliefs
- Checking can (and we argue, largely should) be done mechanically

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



What Is Testing?

- Excellent testing is not merely a branch of computer science
  - testing *includes* computer science, mathematics, technical domains
  - BUT… focus only on programs and functions, and you leave out questions of *value* and other relationships that include people
- To me, excellent testing is more like *anthropology*—interdisciplinary, systems-focused, investigative, storytelling

Biology

Archaeology

Language

Culture

Testing Is Like Crime Scene Investigation

- There are many tools, procedures, and 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.
- The clients (not investigators) make the decisions about how to proceed, informed by evidence that investigators reveal.

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

Find the roaches:

Boss's *fantasy (good searching is easy!)*

Find the roaches:

Tester's actual job (good searching is hard)

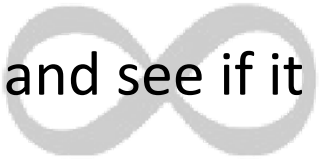
What is testing?

Getting Answers

"Try it and see if it works."

Procedures	Coverage	Oracles
• Get different versions • Set them up • Try simple things • Try complex things • Try sequences • Try combinations • Try weird things • Try them again	• Where to look? • How to look? • What's there? • What's not there? • What's *invisible*? • Did it change? • *Will* it change? • How 'bout now?	• Read the spec • There's no spec? Oh. • There IS a spec! Cool! But… it's old and wrong. • Find inconsistencies • Find "obvious" problems • Find obscure problems • Find **bad** problems

What is testing?



"Try it and see if it works."

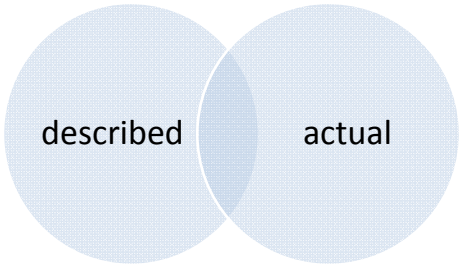
really means...

"Try it to **discover enough**,
about whether it **can work**,
and how it **might not work**,
to learn whether it will work."

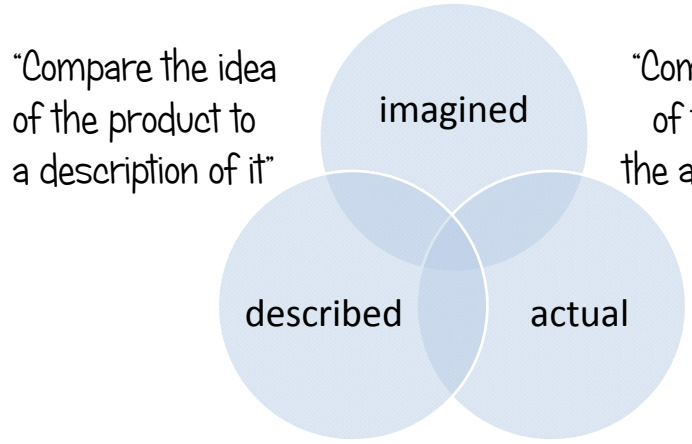
21

This is what people think testers do

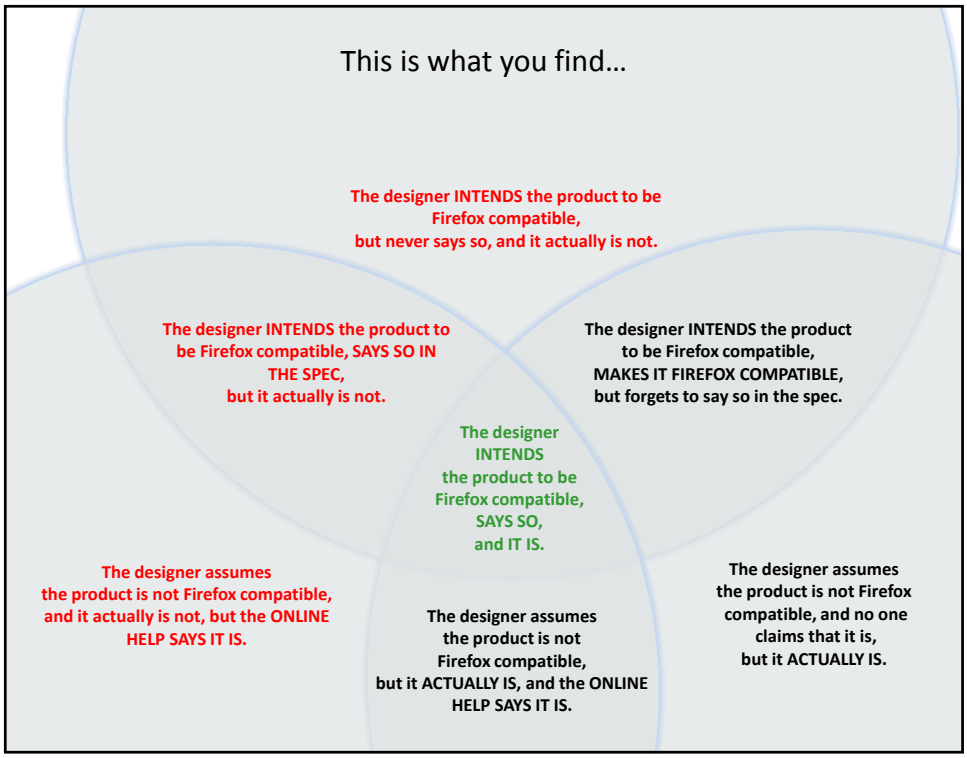
"Compare the product to its specification"



This is more like what testers *really* do



This is what you find…



Testers Are Sensory Instruments For Their Clients

Like scientific instruments, we testers extend our clients' senses to help raise awareness about product, projects and risks.

We also use our own tools to extend our clients' senses even farther.

How Do People React to Software?

Impatience

Frustration

Amusement

Surprise

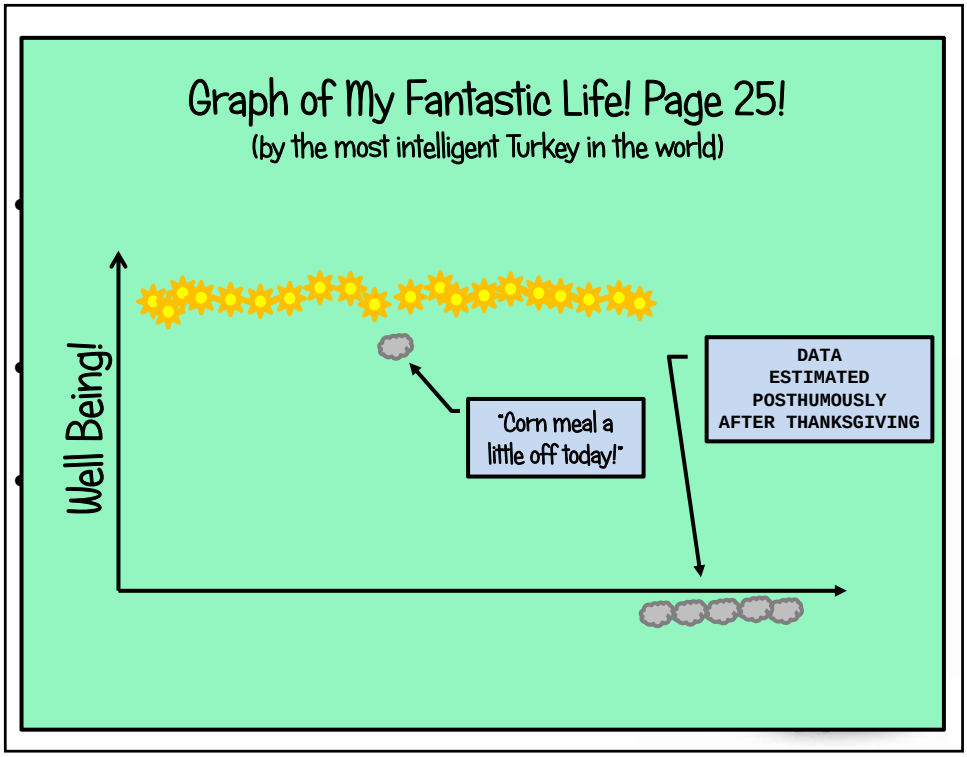
Confusion

Annoyance

Emotions as Trigger Heuristics

What might feelings be telling us?

- Impatience ⇨ an intolerable delay?
- Frustration ⇨ a poorly-conceived workflow?
- Amusement ⇨ a threat to someone's image?
- Surprise ⇨ inconsistency with expectations?
- Confusion ⇨ unclear interface? poor testability?
- Annoyance ⇨ a missing feature?
- Boredom ⇨ an uninteresting test?
- Tiredness ⇨ time for a break?



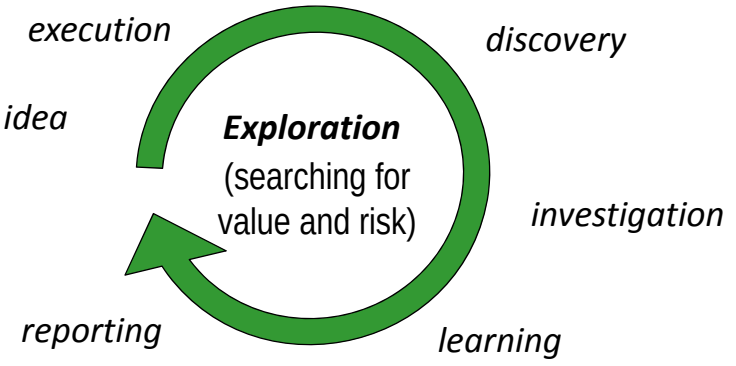
Don't Be A Turkey

- No experience of the past can LOGICALLY be projected into the future, because we have no experience OF the future.
- This is no big deal in a world of stable, simple patterns.
- **BUT NEITHER SOFTWARE NOR PROJECTS ARE STABLE OR SIMPLE.**
- **"PASSING" TESTS CANNOT PROVE SOFTWARE GOOD.**

Based on a story told by Nassim Taleb, who stole it from Bertrand Russell, who stole it from David Hume.

Focus on *Learning* and *Adaptation*

Skilled testers help to defend the value of the product by *learning* about it on behalf of our clients.



…and there are little loops between all of these things.

Confirmatory testing *feels* positive,
but prevents insight and conceals risk.

Four Kinds of Risk Drivers

- problems in the *product*
- *unawareness* of problems in the product
- problems in the *project*
- *unawareness* of problems in the project

📖 *Risk (n.) Some **person** will suffer **annoyance, loss, or harm**, because of a **vulnerability** in a product or project that is triggered by some **threat**.*

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?
- What about all the tests you *haven't* performed yet?

Regression Problems Are *Symptoms*

- If you see a consistent pattern of regression
  - the bugs you're seeing are probably not your biggest problem
  - your biggest problem might be *a favourable environment for regression*
  - is the project simply going too fast in a complex, volatile world?

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…
…about what you haven't tested, yet…
…and won't test, at all…

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.

34

Not-So-Good Questions for Testers

- Is the product done?
- Are we ready to ship?
- Is it good enough?
- How much time do you need to test?
- How many test cases are passing and failing?
- How many tests cases have you run?
- How many bugs are in the product?

Seek more than data.
Seek information.

Better Questions for Testers

- What is the product story? What can you tell me about important problems in the product?
- What risks I should be aware of?
- What have you done to obtain the product story?
- What important testing remains to be done?
- What problems are slowing testing down or making it harder to find out what we might need to know?
- What do you need to help speed things up?
- What *specific* aspects of testing are taking time?
- How do your tests link to the mission?

What might prevent the on-time, successful completion of the project?

What Interrupts Test Coverage?

- Setup, bug investigation, and reporting *take time away* from test design and execution
- Suppose that a testing a feature takes two minutes
  - this is a highly arbitrary and artificial assumption—that is, it's *wrong*, but we use it to model an issue and make a point
- Suppose also that it takes an extra eight minutes to investigate and report a bug
  - another sweeping generalization in service of the point
- In a 90-minute session, we can run 45 feature tests—*as long as we don't find any bugs*

How Do We Spend Time?

(assume we're fabulous testers and would find everything our clients deem a bug)

Module	Bug reporting/investigation (time spent on tests that find bugs)	Test design and execution (time spent on tests that find no bugs)	Number of tests
A (good)	0 minutes (no bugs found)	90 minutes (45 tests)	45
B (okay)	10 minutes (1 bug, 1 test)	80 minutes (40 tests)	41
C (bad)	80 minutes (8 bugs, 8 tests)	10 minutes (5 tests)	13

Investigating and reporting bugs means….

SLOWER TESTING **or…**

REDUCED COVERAGE **…or both.**

- For Module A, our *coverage* is great—but if our clients assess us on the number of bugs we're finding, we look bad.
- For Module B, coverage looks good, and we found a bug, too.
- For Module C, we look good because we're finding and reporting lots of *bugs*—but our *coverage* is suffering severely.
- A system that rewards us or increases confidence based on the number of bugs we find might mislead us into believing that our product is well tested.

What Happens The Next Day?

(assume 6 minutes per bug fix verification)

	Fix verifications	Bug reporting and investigation today	Test design and execution today	New tests today	Total over two days
A	0 min	0 min (no new bugs)	90 min (45 tests)	45	90
B	6 min	10 min (1 new bug)	74 min (37 tests)	38	79
C	48 min	40 min (4 new bugs)	2 min (1 test)	5	18

Finding bugs today means….

VERIFYING FIXES LATER

…which means….

EVEN SLOWER TESTING **or…**

EVEN LESS COVERAGE **…or both.**

- …and note the optimistic assumption that all of our fixed verifications worked, and that we found no new bugs while running them. Has this ever happened for you?

39

13 Commitments for Testers to Make to Clients

1. I provide a service. You are an important client of that service. I am not satisfied unless you are satisfied.
2. I am not the gatekeeper of quality. I don't "own" quality. Shipping a good product is a goal shared by all of us.
3. I will test anything as soon as someone delivers it to me. I know that you need my test results quickly (especially for fixes and new features).
4. I will strive to test in a way that allows you to be fully productive. I will not be a bottleneck.
5. I'll make every reasonable effort to test, even if I have only partial information about the product.
6. I will learn the product quickly, and make use of that knowledge to test more cleverly.
7. I will test important things first, and try to find important problems. *(I will also report things you might consider unimportant, just in case they turn out to be important after all, but I will spend less time on those.)*

13 Commitments for Testers to Make to Clients

8. I will strive to test in the interests of everyone whose opinions matter, including you, so that you can make better decisions about the product.
9. I will write clear, concise, thoughtful, and respectful problem reports. *(I may make suggestions about design, but I will never presume to be the designer.)*
10. I will let you know how I'm testing, and invite your comments. And I will confer with you about little things you can do to make the product much easier to test.
11. I invite your special requests, such as if you need me to spot check something for you, help you document something, or run a special kind of test.
12. I will not carelessly waste your time. Or if I do, I will learn from that mistake.
13. I will not FAKE a test project.

The Themes of Rapid Testing

- Put the **tester's mind** at the center of testing.
- Learn to **deal with complexity** and ambiguity.
- Learn to **tell a compelling testing story.**
- Develop **testing skills** through practice, not just talk.
- **Use heuristics** to guide and structure your process.
- **Be a service** to the project community, not an obstacle.
- **Consider cost vs. value** in all your testing activity.
- **Diversify** your team and your tactics.
- Dynamically **manage the focus** of your work.
- Your **context should drive your choices**, both of which evolve over time.

Testers light the way.

This is our role.

We see things for what they are.
We make informed decisions about quality possible, because we think critically about software.

How to Get What You Really Want from Testing

Michael Bolton
DevelopSense
http://www.developsense.com
@michaelbolton
michael@developsense.com