Two Futures of Software Testing

Michael Bolton
DevelopSense
Ireland
September 2010

SoftTest Ireland
http://www.SoftTest.ie

Thank you to our sponsors!

Sogeti Ireland

Delivery Partners

SoftTest Ireland
http://www.SoftTest.ie

Thank you to our sponsors!

InterTradeIreland

Delivery Partners

SoftTest Ireland
http://www.SoftTest.ie

Thank you to our sponsors!

OUTCOMES THAT MATTER.

HP Ireland

Delivery Partners

SoftTest Ireland
http://www.SoftTest.ie

Special thank you to today's sponsors!

Software Quality Systems

Delivery Partners

SoftTest Ireland
http://www.SoftTest.ie

…and thank you to the organizers!

Anne-Marie Charrett
Brian Lambert
Stephen Sloan
Nicola McManus
David Jamison

Delivery Partners

Who I Am

Michael Bolton

(not the singer, not the guy in Office Space)

DevelopSense, Toronto, Canada

mb@developsense.com
+1 (416) 992-8378
http://www.developsense.com

Acknowledgements

- James Bach
  - some of the material comes from the Rapid Software Testing Course, of which James is the senior author and I am co-author
- Cem Kaner
- Bret Pettichord
- Jerry Weinberg
- Jonathan Kohl
- Anne-Marie Charrett

These are not *predictions*.
These are *proposals*.

These are not the only two futures.
They're offered for your consideration.
The choices are up to you.

The Dark Future: Testing ISN'T About Learning

- Testing is focused on confirmation, verification, and validation
- There are prescribed tests; testers check to make sure that prescribed tests pass
- Though we're in a "knowledge economy", some knowledge can be unpleasant and dangerous, thus…
- Exploration and investigation are luxuries at best, threats at worst

The Dark Future: Change is Rejected

- Nothing is more important than following our plans and our processes strictly
  - our clients will understand, of course
  - if they want to change the requirements, we say *they should have known that from the beginning*
  - and if they don't like that, we'll call them names like "immature" or "unprofessional"
- By insisting that requirements don't change, we can eradicate project risk

The Dark Future: Measurement

- We measure
  - requirements scope by *counting requirements*
  - test coverage by *counting test cases*
  - product quality by *counting bugs*
  - the value of testers by *counting bug reports*
  - developer output by *counting lines of code*
  - complexity by *counting code branches*

The Dark Future: Measurement

- We **don't** measure by
  - qualitative measures
  - direct observation
  - interaction between testers and programmers
  - conversation with actual users
- We don't trust stories; only statistics
- We don't worry about construct validity or other problems in measurement

The Dark Future: Automation is Paramount

- Machines are obviously better than people
- If testing is scripting and script is good, then automated scripting is better
- By eliminating the human element, we can eliminate variability and uncertainty
- Sure, high-level test automation takes time and effort to prepare, therefore…
- …we must slow down development to let "testing" catch up

The Dark Future: Putting The Testers In Charge

- Testers are the quality gatekeepers
- Testers refuse to test until they have been supplied with complete, unambiguous, up-to-date requirements documents
- Testers "sign off" on project readiness
- Testers can block releases
- Testers are the real project managers

Project managers don't know what's good for them!

The Dark Future: Not Putting The Testers In Charge

- Although testers are *called* the quality gatekeeper, they don't have control over
  - schedule
  - budget
  - staffing
  - product scope
  - market conditions or contractual obligations

Responsibility without authority!

The Dark Future: Promoting Orthodoxy

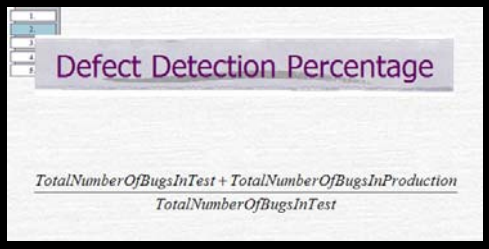
- All testers must be certified
  - by passing multiple choice exams
- All testers have the same skills
  - testing doesn't require skilled labour anyway
- Testers must be isolated from developers
- Investigation is banned; variation suppressed
- Testing is standardized across departments and throughout the "industry"

Context doesn't matter!

Standardization

- There shall be One True Way to Test
- There shall be one universal language for testing
  - and since American and British consultants promote it, it shall be English
- Agile approaches can still be made *very* orthodox
- If we find it hard to apply standard practices, we'll *say* that we apply them

The Dark Future: Some Of Our Proudest Accomplishments

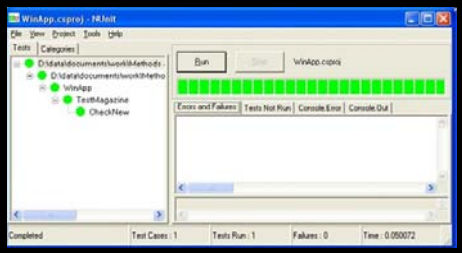


A bug is not a thing in the world. A bug is a relationship between some product and some person. Bugs are by their nature qualitative relationships, rather than quantitative units. Beware measurement dysfunction.

The Dark Future: Some Of Our Proudest Accomplishments

Test cases are like briefcases; they're containers. Counting containers without knowing what they contain is measurement without observation.
The containers are the least interesting part of the story.

The Dark Future: Some Of Our Proudest Accomplishments



Checking is very important and useful. But when test results are reduced to nothing more than a green bar, rather than the parallel, complex, and subtle product and testing stories, we run the risk of leaving out critically important information.

The Dark Future: Some Of Our Proudest Accomplishments

It's entirely possible, and even impressive, to deploy software continuously. But it begs the question of *why* you might want to do it, and the value that it adds. Are fifty deployments a day too few? Too many?

So you can deploy… but what are you deploying? And how do you know?

The Dark Future: Pathologies

- Places knowledge and learning up front, at the beginning of the project
  - when we know the *least* about it!
- Learning through the project is ignored
- Testing is confused with checking
- Testing is considered to be rote, unskilled work
- Machines are valued over human cognition
- Tasks and tools are confused with each other
- Measurement is riddled with *basic* problems
  - primarily reification error and rotten construct validity

The Dark Future: Pathologies

- Testers implicitly run the project *when it's convenient* for management to let them
- Even though testers are essentially powerless, testers are still held responsible for all quality lapses

The worst thing about the dark future is…

it's so much like today.

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

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 programs, 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
- Anthropology focuses on investigating
  - biology
  - archaeology
  - linguistics
  - cultures

The Bright Future: *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
BUT
We let project owners make the business decisions.

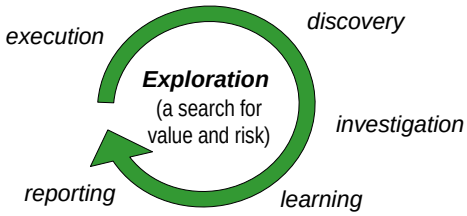
So What Are We Testers?

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)

The Bright Future: The Mission is *Learning*

Testers help to defend the value of the product by *learning* on behalf of our clients



We Are Sensory Instruments For Our Clients

The Bright Future: Testers Embrace Change

- Change WILL happen
  - in market conditions…
  - contracts…
  - requirements…
  - specifications…
  - designs…
  - documents…
  - products…
  - systems…
- We help our clients understand the *implications* of change

The Bright Future: Measurement for Inquiry, NOT Control

- Metrics like Defect Detection Percentage ignore almost every relevant factor
  - difficulty of the problems being solved
  - quality of the design
  - quality of the code
  - release timing
  - who made the release decision, and why
  - timing of customer adoption
  - the fact that requirements and bugs are *relationships*
- …but are routinely used to evaluate the quality of *testing*

The Bright Future: Observation Over Counting

Instead of this…	we consider this.
quantitative criteria	qualitative criteria
data	information
bug counts	problem and issue stories
test cases completed	multivariate coverage
pass/fail ratio	"Is there a problem here?"
release metrics	good enough quality
one test per requirement	risk focus
what numbers tell us	what numbers leave out
blame	understanding

The object of measurement is not to provide answers, but to suggest better questions.

The Bright Future: Testing Is More Than *Checking*

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

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

What *IS* Checking?

- A *check* has three attributes
  - It requires an *observation*
  - The observation is linked to a *decision rule*
  - The observation and the rule can be applied

without sapience

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

- "Sapient" means "requiring human wisdom"
- A ***non-sapient*** activity can be performed

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

by a human who has been
instructed NOT to think
(and who is slow and erratic)

Checking ISN'T New

- Despite what the Agilists might have you believe, checking is *not* new
  - D. McCracken (1957) refers to "program checkout"
  - Jerry Weinberg: checking was important in the early days because
    - computer time was expensive
    - programmers were cheap
    - the machinery was so unreliable
- Checking has been *rediscovered* by the Agilists
  - centrally important to test-driven development, refactoring, continuous integration & deployment
  - successful checking must be surrounded by skilled testing work

Checking IS Important

- Checks help to establish baseline functionality in test-driven development
- Checks serve as change detectors
- Excellent checking helps programmers to refactor (improve the quality of existing code without changing functionality) at top speed
- Checks provide a first-line defense against regression problems

…But Checking Has Limitations

- Checks tend to be designed early…
- *…when we know less than we'll ever know* about the product and the project
- Checks focus on "pass vs. fail?"
- Skilled testers focus on a different question:

Is there a problem here?

Risks With "Acceptance Tests"

- They tend to be set at the beginning of an iteration or development cycle
  - when we know less about the product than we'll ever know.
- Talk about acceptance tests tends to leave out questions of *who* is accepting *what*, and *for what purpose*.
- Acceptance tests are *examples*. They tend to cover non-implementation risks very poorly
- Acceptance tests are *checks*, not tests.
- Properly viewed, they should prompt rejection for failing, rather than acceptance for passing.
- Therefore: they should be called *rejection checks.*

Checks themselves are skill-free, but *checking* is dominated by testing skill.

Before the Check

• Recognize a risk	⇨ Testing skill
• Translate to a test idea	⇨ Testing skill
• Express a test idea as a bit	⇨ Testing skill
• Turn the question into code	⇨ Programming skill
• Determine the trigger	⇨ Testing skill
• Encode the trigger	⇨ Programming skill

After The Check

• Read the bit	⇨ Programming skill
• Aggregate bits	⇨ Programming skill
• Design a report	⇨ Testing, design skill
• Encode the report	⇨ Programming skill
• Observe the report	⇨ Testing skill
• Determine meaning	⇨ Testing skill
• Determine significance	⇨ Testing skill
• Respond	⇨ Testing, programming, and management skill

The Bright Future: Repeatability vs. Adaptability

- Repeatability, for computers, is relatively easy, but testing is not mere repetition. It's an open search.
- Skilled testing therefore focuses on *adaptability*, *value*, and *threats to value*

This kind of testing CAN NOT be scripted

The Bright Future: Testing IS Exploring

- Our community sees testing as exploration, discovery, investigation, and learning
  - Testing can be *greatly 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.**

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

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.

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

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

Why Explore?

- You cannot use a script to
  - investigate a problem that you've found
  - decide that there's a problem with a script
  - escape the script problem you've identified
  - recognize terrible risks in the product
  - determine the best way to phrase a report
  - unravel a puzzling situation

Even "scripted" testers explore all the time!

So why don't we hear more about E.T.?

FEAR

- Maybe managers fear that E.T. depends on skill
  - but who benefits from ANY unskilled testing?
- Maybe managers fear that E.T. is unstructured
  - but it is structured
- Maybe managers fear that E.T. is unaccountable
  - but it can be entirely accountable
- Maybe managers fear that E.T. is unmanageable
  - but you can manage anything if you put your mind to it

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.

Exploratory Testing IS Structured

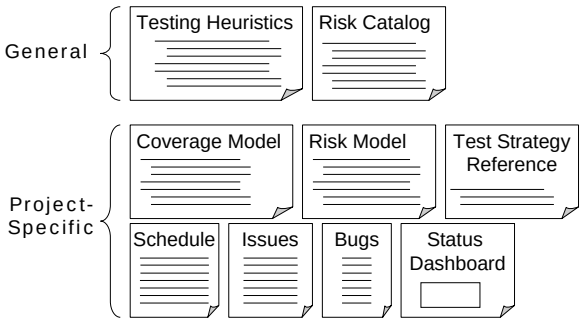
- We've studied the structures of ET, we've written about it, and we know how to teach it
- The structure of ET comes from *many* sources
  - Test design heuristics
  - Chartering
  - Time boxing
  - Perceived product risks
  - The nature of specific tests
  - The structure of the product being tested
  - The process of learning the product
  - Development activities
  - Constraints and resources afforded by the project
  - The skills, talents, and interests of the tester
  - The overall mission of testing

Not *procedurally* structured, but *cognitively* structured.

In other words, it's not "random", but systematic.

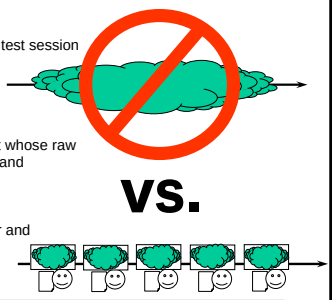
http://www.developsense.com/resources.html#exploratory

Exploratory Testing IS Accountable
Concise Documentation Minimizes Waste



Exploratory Testing IS Accountable
Session-Based Test Management

- Charter
  - A clear, concise mission for a test session
- Time Box
  - 90-minutes (+/- 45)
- Reviewable Results
  - a session sheet—a test report whose raw data can be scanned, parsed and compiled by a tool
- Debriefing
  - a conversation between tester and manager or test lead

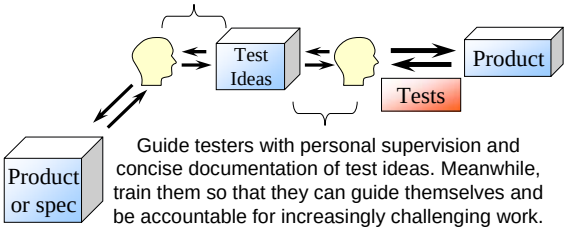


For more info, see http://www.satisfice.com/sbtm

57

Exploratory Testing IS Manageable

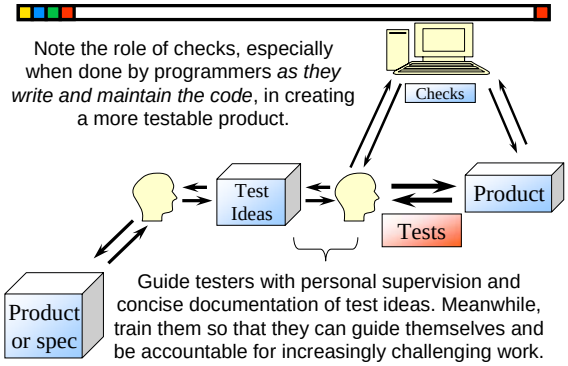
Achieve excellent test design by exploring different test designs *while actually testing and interacting with the system*



Guide testers with personal supervision and concise documentation of test ideas. Meanwhile, train them so that they can guide themselves and be accountable for increasingly challenging work.

Exploratory Testing IS Manageable

Note the role of checks, especially when done by programmers *as they write and maintain the code*, in creating a more testable product.



Guide testers with personal supervision and concise documentation of test ideas. Meanwhile, train them so that they can guide themselves and be accountable for increasingly challenging work.

My Alternative to Certification

- I read books and articles that are *not* about testing
  - science and physics
  - mathematics and statistics
  - cognitive psychology and critical thinking
  - computer programming and software design
  - food and cooking
  - general systems
  - medicine
  - economics
  - social sciences
  - history
  - comedy
- I relate these disciplines to testing, and describe the value of the relationships

My Alternative to Certification

- I practice and teach *testing*
  - whereby I gain experience by succeeding and failing
- I practice critical thinking
  - whereby I try to avoid fooling myself and others
- I practice systems thinking
  - whereby I learn to see the big and small pictures
- I practice programming
  - whereby I obtain humility
- I practice describing my practices
  - orally
  - in writing (magazine articles, blogs, etc.)
  - in presentations (like this one)
- I participate in a community that works this way.

The Movement to Standardize Testing

- Standardization of testing is like the standardization of tester certification
- We all know how well that has worked out
  - for the testing community at large
  - for individual testers
  - for organizations who have fallen for the marketing
  - AND for a small group of certification salespeople
- Ask yourself:
  - 130,000 testers times at least $100 per exam… where does that (at least) $13,000,000 go?
  - Who is most aggressively promoting ISO 29119?

The future of testing
is up to us.

These are not the only two futures.
They're offered for your consideration.
The choices are up to you.

Who I Am

Michael Bolton

(not the singer, not the guy in Office Space)

DevelopSense, Toronto, Canada

mb@developsense.com
+1 (416) 992-8378
http://www.developsense.com

Web Resources

- Michael Bolton http://www.developsense.com
- James Bach http://www.satisfice.com
- Cem Kaner http://www.kaner.com
- The Florida Institute of Technology
  - http://www.testingeducation.org
  - http://www.testingeducation.org/BBST/index.html
- StickyMinds http://www.StickyMinds.com
- Risks Digest http://catless.ncl.ac.uk/risks

Bibliography
How To Think About Testing

- *Perfect Software and Other Illusions About Testing*
  - Gerald M. Weinberg
- *Lessons Learned in Software Testing*
  - Cem Kaner, James Bach, and Bret Pettichord
- "Software Testing as a Social Science"
  - Cem Kaner; http://www.kaner.com/pdfs/KanerSocialScienceSTEP.pdf
- *Testing Computer Software*
  - Cem Kaner, Jack Falk, and Hung Quoc Nguyen
- *An Introduction to General Systems Thinking*
  - Gerald M. Weinberg
- *Exploring Requirements: Quality Before Design*
  - Gerald M. Weinberg

Bibliography
Recommended Test Technique Books

- *A Practitioner's Guide to Test Design*
  - Lee Copeland
- *How to Break Software*
  - James Whittaker
- *How to Break Software Security*
  - James Whittaker and Herbert Thompson
- *Lessons Learned in Software Testing*
  - Cem Kaner, James Bach, and Bret Pettichord
- *Testing Applications on the Web*
  - Hung Quoc Nguyuen
- *Hacking Web Applications Exposed*
  - Joel Scambray and Mike Shema

Bibliography
Jerry Weinberg

- *Quality Software Management Vol. 1: Systems Thinking*
- *Quality Software Management Vol. 2: First Order Measurement*
- *Secrets of Consulting: How to Give and Get Advice Successfully*
- **Anything** by Jerry Weinberg

Bibliography
Richard Feynman

- *The Pleasure of Finding Things Out*
  - see the Appendix to the Challenger Report.
- *Surely You're Joking, Dr. Feynman! Adventures of a Curious Character*
- *What Do You Care About What Other People Think?*

Bibliography
Other Areas

- *The Social Life of Information*
  - Paul Duguid and John Seely Brown
- *Please Understand Me*
  - David Kiersey
  - The Myers-Briggs Type Inventory, which provides insight into your own preferences and why *other people* seem to think so strangely
- *The Visual Display of Quantitative Information*
  - Edward Tufte
  - How to present information in persuasive, compelling, and beautiful ways
- *A Pattern Language*
  - Christopher Alexander et. al
  - A book about architecture
  - even more interesting as a book about thinking and creating similar but unique things—like computer programs and tests for them