



Two Futures of Software Testing

Michael Bolton
DevelopSense
Scottish Testing Group
May 2010



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
- Justin Webster (HP)
- Annmarie Haynes (HP)
- Special thanks to Dennis Hong (HP)



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
- Testing is merely checking to make sure that prescribed tests pass
- Even though we're in a "knowledge economy", we believe that some knowledge can be unpleasant and dangerous, thus...
- Exploration and investigation are luxuries at best, threats at worst

The Dark Future: Automation is Paramount

- Very simply, machines are better than people; that should be obvious
- 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: 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; we only trust statistics
- We don't worry about construct validity or other problems in measurement

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, because project managers don't know what's good for them

Not The Dark Future: Putting The Testers In Charge

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

Not The Dark Future: Putting The Testers In Charge

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

**Responsibility
without authority!**

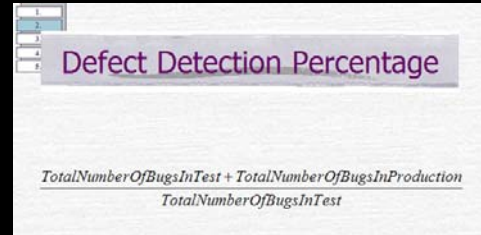
The Dark Future: Promoting Orthodoxy

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

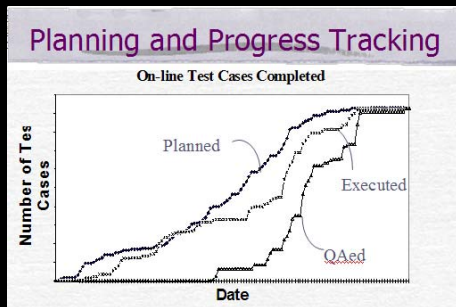
Standardization

- 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 follow the book
- If we find it hard to apply the practices, we'll say that we apply them, and that will be good enough

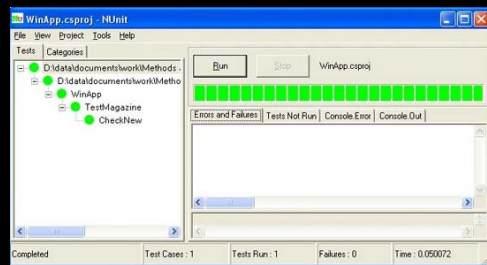
The Dark Future: Some Of Our Proudest Accomplishments



The Dark Future: Some Of Our Proudest Accomplishments



The Dark Future: Some Of Our Proudest Accomplishments



The Dark Future: Some Of Our Proudest Accomplishments



The Dark Future: Pathologies

- Places knowledge and learning up front, at the beginning of the project
 - when we know the *least* about it!
- Testing is confused with checking
- Learning through the project is ignored
- Testing is considered to be rote, unskilled work
- Machines are trusted; human cognition is devalued
- Measurement is riddled with basic critical thinking errors
 - 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 don't have control over schedule, budget, staffing, contractual obligations, product scope, or reward systems
 - testers neither create nor hide the bugs
 - ...testers are still held responsible for all quality lapses
- Even in the Agile world, we're working on the problems with *testers*, but we still haven't quite got our heads straight about...

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 don't have control over schedule, budget, staffing, contractual obligations, product scope, or reward systems
 - testers neither create nor hide the bugs
 - ...testers are still held responsible for all quality lapses
- Even in the Agile world, we're working on the problems with *testers*, but we still haven't quite got our heads straight about...

Testing!

The worst thing about
the dark future is...

The worst thing about
the dark future is...

it's so much like today.

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*

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*

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.*

The Bright Future: Value is Central

- Testing is a deeply human activity.
- It's all about value for people.
- It's strengthened by the unique contribution of the individual tester.
- The product is a solution. If the problem isn't solved, the product doesn't work, AND...
- If the product doesn't work, *the problem isn't solved.*

The Bright Future: Testing Isn't Just 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>

The Bright Future: Testing Isn't Just 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 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:

...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?

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

The Danger of Test Scripts



- Scripts **aren't necessary** for skilled (human) testers
- Script preparation **takes away from testing time**
- Bugs found and fixed during script prep tend to stay fixed
- Scripts **separate** design, execution, interpretation, and learning...and thus **DE-SKILL**
- Scripts drive **inattentional blindness**

See Kaner, "The Value of Checklists and The Danger of Scripts"
<http://www.kaner.com>

Positive Test Strategy

- "A tendency to test cases that are expected (or known) to have the property of interest rather than those expected (or know) to lack that property."
- "...can be a very good heuristic for determining the truth or falsehood of a hypothesis under realistic conditions."
- It can, however, lead to systematic errors or inefficiencies.

• Klayman and Ha, 1987

Escaping the Positive Test Strategy Trap

- When people seek matches, they use relatively few tests to counter their hypotheses
 - that is, they tend to run confirmatory tests
- When the categories are relabeled from *yes* and *no* to two neutral categories (“DAX” and “MED”), people use *even fewer* “negative” tests (Tweeny et. al., 1980)
- BUT... they run positive tests for *each* category
 - which gets around the problem

Our brains can be productively hacked.

Positive Test Strategy

- “When concrete, task-specific information is lacking, or cognitive demands are high, *people rely on the positive test strategy as a general default heuristic.*” BUT...
- “emphasis on the sufficiency of one’s actions is enhanced when one is rewarded for each individual success rather than only for the final rule discovery.”

• Klayman and Ha, 1987

which means...

The Great Traps of Test Cases

If you want to know how the system works and fails,

a focus on counting test cases is a really bad idea.

The Bright Future: Repeatability vs. Adaptability

- Repeatability, for computers, is relatively easy
- Skilled testing therefore focuses on *adaptability*, *value*, and *threats to value*

The Bright Future: Repeatability vs. Adaptability

- Repeatability, for computers, is relatively easy
- Skilled testing therefore focuses on *adaptability*, *value*, and *threats to value*

This kind of testing CAN NOT be scripted

Humans can...

recognize new risks investigate speculate
empathize anticipate judge predict suggest
recognize refocus contextualize elaborate
appreciate strategize evaluate
become resigned question charter assess
teach learn get frustrated
reframe work around a problem
invent model make conscious decisions
troubleshoot collaborate refine resource

Humans can...

recognize new risks investigate speculate
empathize predict suggest
recommen... judge project
become resigned contextualize elaborate
teach question change
reframe learn... frustrated
invent work around a problem
model take conscious decisions
troubleshoot collaborate refine resource

The Bright Future: 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

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

The Bright Future: 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

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

What IS Exploratory Testing?

- Simultaneous test design, test execution, and learning.
 - James Bach, 1995

But maybe it would be a good idea to underscore why that's important...

What IS Exploratory Testing?

- Simultaneous test design, test execution, and learning, **with an emphasis on learning.**
 - Cem Kaner, 2005

But maybe it would be a good idea to be really explicit about what goes on...

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
 - determine the best way to phrase a report
 - unravel a puzzling situation

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
 - 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

The Bright Future

- The Bright Future of Testing is all about exploration, discovery, and investigation
- To get management past the fear and into the value, we need to address issues of
 - Skill
 - Structure
 - Accountability
 - Management
- We need to learn them and practice them

The Bright Future Comes From The Past


“Because we are humans, we will tend to believe what we want to believe, not what the evidence justifies. When we have been working on a program for a long time, and if someone is pressing us for completion, we put aside our good intentions and let our judgment be swayed. So often, then, the results must provide the impartial judgment that we cannot bring ourselves to pronounce.”

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

The Bright Future Comes From The Past

“One of the lessons to be learned from such experiences 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



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



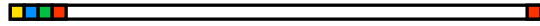
- *A Practitioner's Guide to Test Design*
 - Lee Copeland
- *How to Break Software*
 - James Whittaker
- *How to Break Software Security*
 - James Whittaker and Herbert Thompson
- *Lessons Learned in Software Testing*
 - Cem Kaner, James Bach, and Bret Pettichord
- *Testing Applications on the Web*
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

Bibliography
Jerry Weinberg



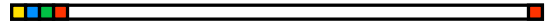
- *Quality Software Management Vol. 1: Systems Thinking*
- *Quality Software Management Vol. 2: First Order Measurement*
- *Secrets of Consulting: How to Give and Get Advice Successfully*
- *Anything* by Jerry Weinberg

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