Synergistic Virtualized Crowdsourced Agile Testing in the Cloud as a Service

Michael Bolton
DevelopSense
http://www.developsense.com
SAP
March 2011

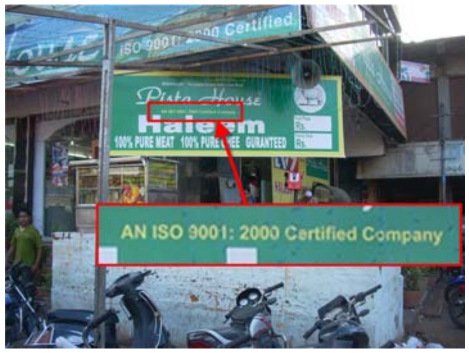
A Rapid Introduction to Rapid Software Testing

Michael Bolton
DevelopSense
http://www.developsense.com
SAP
March 2011

Why Do We Test?

- To make sure the product works?
- To increase confidence before shipping?
- To tell us when it's okay to ship?
- To evaluate the quality of the product by checking that it conforms to standards?

Is Quality "Conformance to Standards"?


ISO 9001: 2000 Certified
Pista House
Haleem
100% PURE MEAT 100% PURE GHEE GURANTEED
AN ISO 9001: 2000 Certified Company

Why Do We Test?

- To make sure the product works?
- To increase confidence before shipping?
- To tell us when it's okay to ship?
- To evaluate the quality of the product by checking that it conforms to standards?
- To find bugs?
- To assure quality?
- To *assist* those who produce quality?

We learn on behalf of others.

What is testing?
Getting answers…

"Try it and see if it works."

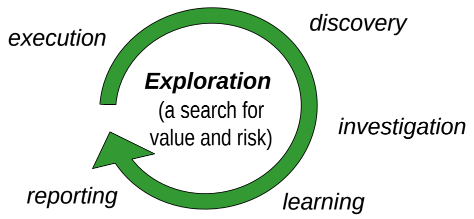
really means...

"**Try it to learn,**
sufficiently, everything that matters
about whether it can work and
how it might not work."

6

The Mission of Testing Is *Learning, Not Merely Confirming*

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



Testing is More Than Checking

- **TESTING:** A questioning activity that employs skills, senses, emotions and intelligence that we are unable to automate.
- **CHECKING:** An information gathering activity that, *in principle*, could be done by machine.

Testing is a *sapient* activity;
checking is not.
Testing encompasses checking,
not the other way round.

Testing is a *Sapient* Process

- "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 variable)

Why Sapience?

- Machines can be programmed to do non-sapient *checking*, to check for *repeatability* and *consistency*.
- But we *test* not only for repeatability, but also for *adaptability*, *value*, and *threats to value*

This kind of testing
CAN NOT
be scripted

What else *don't* we script? Management Cases!

```
Management Case #3412
=====================
Preconditions:

Ensure date is March 21; time 9:23am
Ensure staffing level = 4 members
Set coffee cup to full

Management Steps:
1) Receive annual departmental budget for $752,688.
2) Allocate $501,472 to burdened employee cost.
3) Allocate remaining $251,256 to equipment and tools.
3a) Leave training and book budgets at $0.
4) Receive email from development manager requesting 75 hours
   of testing work on Confabulator IV project. Offer 40.
5) Turn down 3:30pm meeting requested by lead programmer.
6) 3:15 leave office.

Postcondition: Observe whether par has been achieved
   on 4th hole.
```

"Test Cases" describe *only a fraction* of testing.

- Programming cases?
- Driving cases?
- Traveling cases?
- Parenting cases?
- Learning cases?
- Science cases?
- Living cases?

Activities are not
captured by "cases"

Excellent testing is a rich and open-ended intellectual activity.
It cannot be encapsulated into discrete procedural units.

Like a good manager,

- A good tester doesn't simply follow scripts asking

Pass or Fail?

- A good tester *investigates* and asks

Is there a problem here?

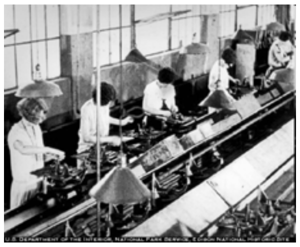
When we want to *learn* something…

- Do we assume that we know all of the right questions to ask in advance?
- Even if we *think* we know all of the right questions, do we know all the right *answers*?
- Do we write down all of our questions?
- Do we decide that once we've answered one set of questions, we're done?
- Do we assume that no new questions will come up as we learn?
- Do we periodically repeat every question we've asked before?

When we want to *learn* something…

- Do we periodically repeat every question we've asked before?
- Do we periodically repeat every question we've asked before?
- Do we periodically repeat every question we've asked before?
- Do we periodically repeat every question we've asked before?
- Do we periodically repeat every question we've asked before?

Software Development Is Not Much Like Manufacturing



- Repetitive checking makes sense for manufacturing, but in software, creating zillions of identical *copies* is not the big issue.

Software Development Is More Like Design

- New designs cannot be checked only; they must be *tested*.

Great Testing Means *Exploring*

- I follow (and contributed to) Kaner's definition, which was refined over several peer conferences through 2007:

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

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.

"Parallel test design, test execution, and learning."

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

Testing Is Like Working in a Crime Investigation Lab

- 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

We Are Sensory Instruments For Our Clients

Introducing Rapid Testing

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

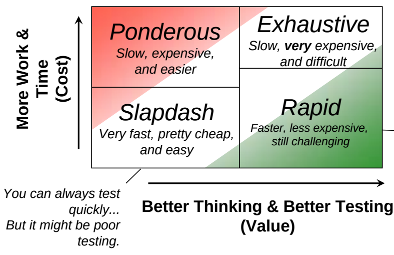
This is a general testing methodology. It adapts to any kind of project or product.

21

How does Rapid Testing compare with other kinds of testing?

When testing is turned into an elaborate set of rote tasks, it becomes ponderous without really being thorough.

Management likes to talk about exhaustive testing, but no one knows how to do it, and if anyone did, managers wouldn't want to fund it.


More Work & Time (Cost)
Ponderous
Slow, expensive, and easier
Exhaustive
Slow, very expensive, and difficult
Slapdash
Very fast, pretty cheap, and easy
Rapid
Faster, less expensive, still challenging
Better Thinking & Better Testing (Value)

You can always test quickly... But it might be poor testing.

Rapid testing may not be exhaustive, but it is thorough enough and quick enough. It's less work than ponderous testing. It might even be less work than slapdash testing.

It fulfills the mission of testing.

22

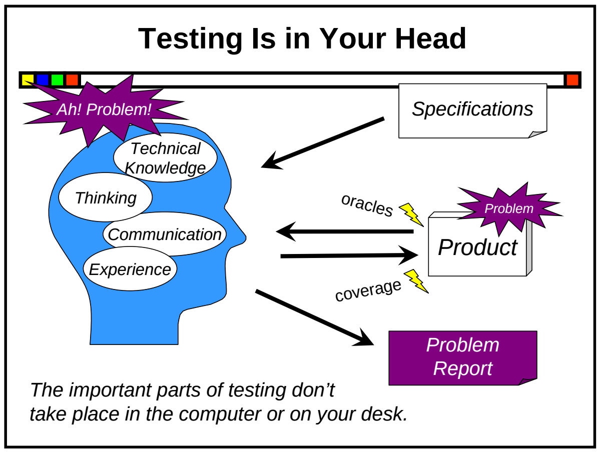
The Themes of Rapid Testing

- Put the **tester's mind** at the center of testing.
- Learn to **deal with complexity** and ambiguity.
- 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.
- Learn to **tell a compelling testing story.**
- Your **context should drive your choices**, both of which evolve over time.

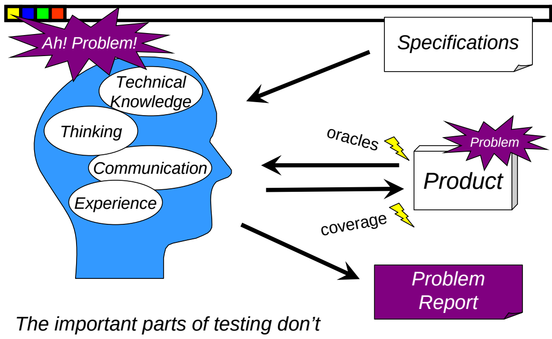
23

Testing is in your head

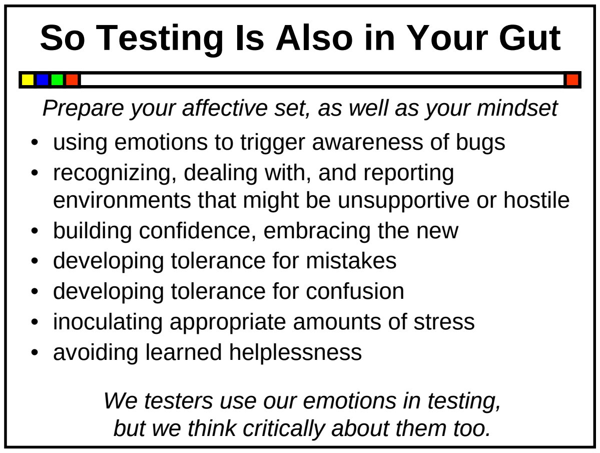
general systems thinking
learning folklore
design of experiments
writing
planning and preparation
programming
selecting tools
wrestling with biases
identifying oracles
recognizing non-linearity
rhetoric
platforms & frameworks
determining coverage
logic
telling the testing story
document design
critical thinking
combinatorics
economics
visualization



Testing Is in Your Head



The important parts of testing don't take place in the computer or on your desk.

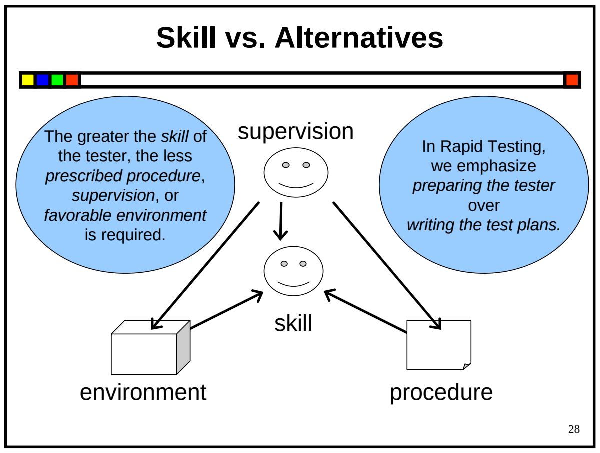


So Testing Is Also in Your Gut

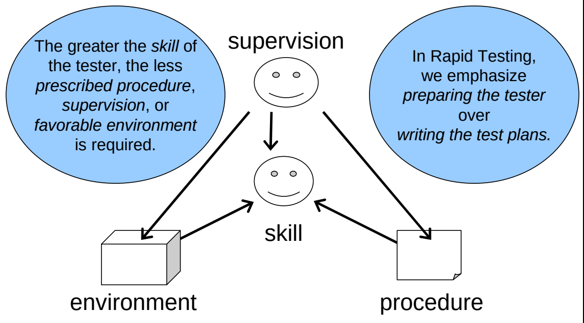
Prepare your affective set, as well as your mindset

- using emotions to trigger awareness of bugs
- recognizing, dealing with, and reporting environments that might be unsupportive or hostile
- building confidence, embracing the new
- developing tolerance for mistakes
- developing tolerance for confusion
- inoculating appropriate amounts of stress
- avoiding learned helplessness

We testers use our emotions in testing, but we think critically about them too.



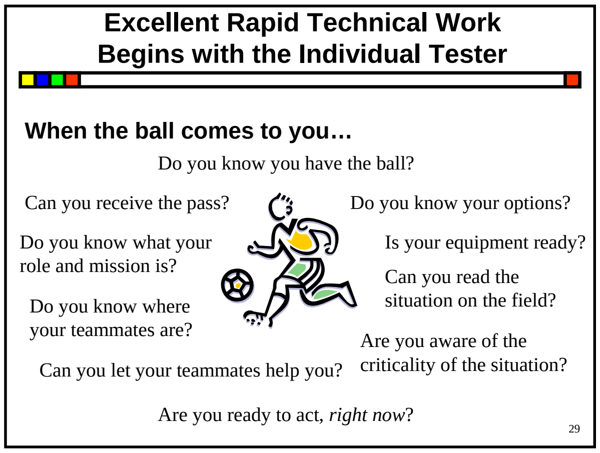
Skill vs. Alternatives



The greater the *skill* of the tester, the less *prescribed procedure*, *supervision*, or *favorable environment* is required.

In Rapid Testing, we emphasize *preparing the tester* over *writing the test plans.*

28



Excellent Rapid Technical Work Begins with the Individual Tester

When the ball comes to you…

Do you know you have the ball?

Can you receive the pass?

Do you know your options?

Do you know what your role and mission is?

Is your equipment ready?

Can you read the situation on the field?

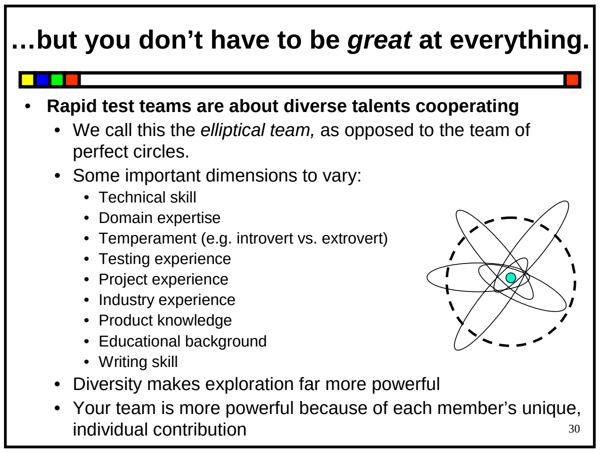
Do you know where your teammates are?

Are you aware of the criticality of the situation?

Can you let your teammates help you?

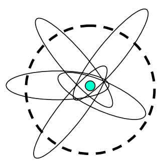
Are you ready to act, *right now*?

29



…but you don't have to be *great* at everything.

- **Rapid test teams are about diverse talents cooperating**
  - We call this the *elliptical team,* as opposed to the team of perfect circles.
  - Some important dimensions to vary:
    - Technical skill
    - Domain expertise
    - Temperament (e.g. introvert vs. extrovert)
    - Testing experience
    - Project experience
    - Industry experience
    - Product knowledge
    - Educational background
    - Writing skill
  - Diversity makes exploration far more powerful
  - Your team is more powerful because of each member's unique, individual contribution



30

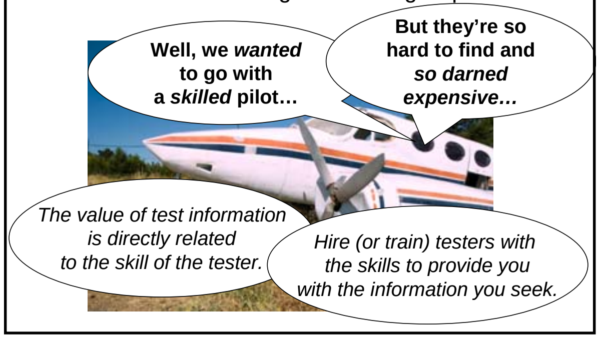
Testing is Multidisciplinary

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

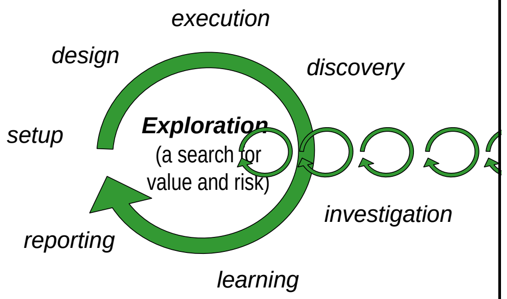
- Excellent testing is not a branch of computer science
  - focus only on programs, and you leave out questions of *value* and other relationships that include people
- To me, excellent testing is more like *anthropology*
  - highly multidisciplinary
  - doesn't look at a single part of the system
-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

Yes, Rapid Testing Requires Skill

- But doesn't ANY testing worth doing require skill?


Well, we *wanted* to go with a *skilled* pilot…
But they're so hard to find and *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.

Rapid Learning Happens In Tight Loops



Cost as a Simplifying Factor
Try quick tests *as well as* careful tests

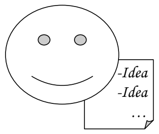
In my travels, I've seen extraordinary emphasis on long cycles of planning without feedback. **This makes testing ineffective and slow.**

A *quick test* is a cheap test that has some value, gives fast feedback, but requires little preparation, knowledge, or time to perform.

Bursts of quick tests represent a great way to *discover* risks upon which more investigation can be focused.

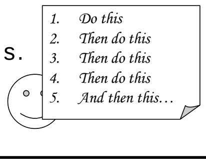
Heuristics are *applied*, not followed.

This…



The skilled tester remains in control of the process.

…not this.



Scripted procedures give the **illusion** of control over unskilled testers.

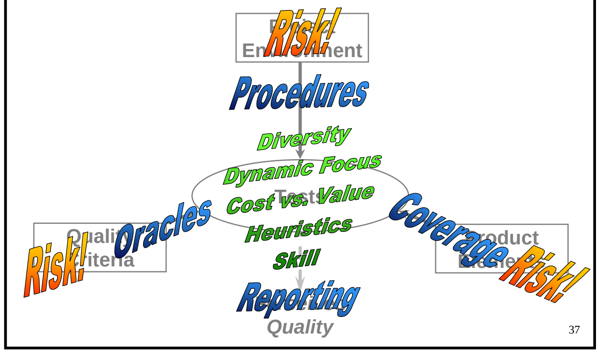
Exploratory Testing IS Structured

- We've studied the structure 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.

A Heuristic Test Strategy Model



37

Oracles

An *oracle* is a heuristic principle or mechanism by which someone might recognize a problem.

(usually works, might fail)

(but not decide conclusively)

📖 Bug (n): Something that bugs someone who matters

Consistency ("this agrees with that")

an important theme in oracles

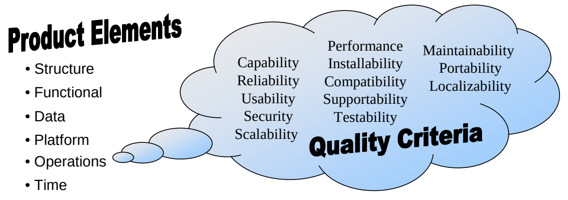
History
Image
Comparable Products
Claims
User Expectations
Purpose
Product
Standards

When a product shows inconsistency with one of these items, we suspect and report a *possible* problem.

Test Coverage Isn't Just *Code* Coverage

Test coverage is the amount of the *system space* that has been tested.

There are as many kinds of coverage as there are ways to model the product.

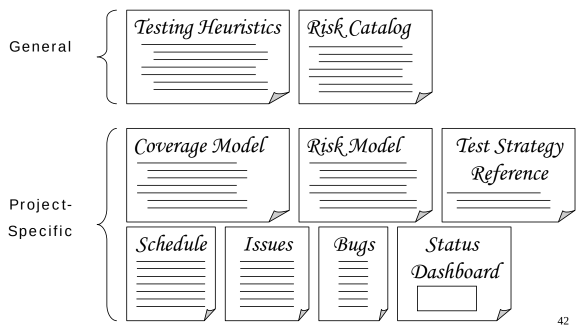


Rapid Automation: Keep It Lightweight

- Think of automation as
  any use of tools to support testing
- Automation can't test, but it *can* help you test more skillfully and more rapidly
  - when it's lightweight and flexible
  - when you decide to make a more testable product

How Do We Document Rapid Testing?

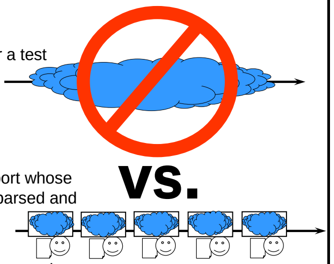
Concise Documentation Minimizes Waste



42

Accountability for Exploratory Testing: Managing Testing Based on Sessions (SBTM)

- 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



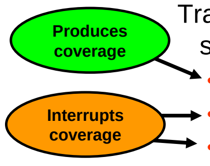
For more info, see http://www.satisfice.com/sbtm

How To Measure Test Coverage

(it's not merely *code* coverage)

- Identify quality criteria
- Consider product elements (structure, function, data, platform, operations, and time)
- Break them down into coverage areas
- Identify session time *focused* on each area
- Assess test coverage in terms of
  - Level 1: Smoke and sanity
  - Level 2: Common, core, critical aspects
  - Level 3: Complex, challenging, harsh, extreme, exceptional

How To Measure ET Efficiency



Track rough percentage of time spent on

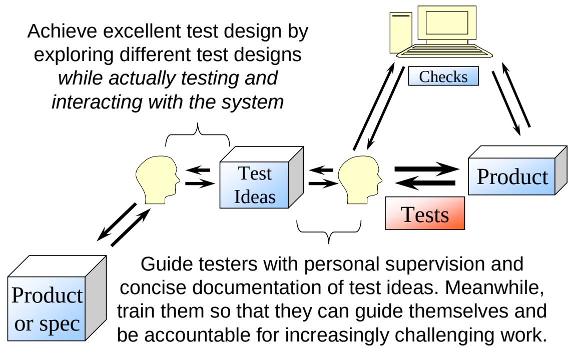
- Test design and execution
- Bug investigation and reporting
- Setup

Ask why time was spent on each:

- Lots on T *might* indicate great code, but *might* indicate poor bug-finding skill
- Lots on B *might* mean code quality problems, but might suggest inefficiency in reporting
- Lots on S *might* mean testability or configuration problems for customers, or it *might* mean early days of testing

How To Manage Exploratory Testing

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

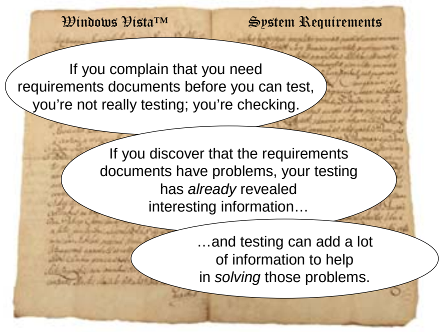


Guide testers with personal supervision and concise documentation of test ideas. Meanwhile, train them so that they can guide themselves and be accountable for increasingly challenging work.

Viewing Testing as an Investigative Service Solves Many Problems

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

Viewing Testing as an Investigative 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.

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 the value of the testing, and threats to it…

…what the risks and costs of testing are…

…how testable (or not) the product is…

…what you need and what you recommend.

49

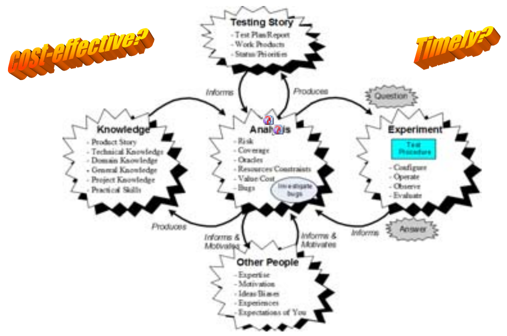
What is test framing?

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

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?

A Story of Rapid Testing
It's Not Linear!



52

STOP
PAY TOLL
We’re here to add value,
not collect taxes.

We’re here
to be a service
to the project,
not an obstacle.