A Rapid Introduction to Rapid Software Testing

James Bach, Satisfice, Inc.
james@satisfice.com
www.satisfice.com
+1 (360) 440-1435

Michael Bolton, DevelopSense
mb@developsense.com
www.developsense.com
+1 (416) 656-5160

Updates

- This presentation is ALWAYS under construction
- Updated slides at http://www.developsense.com/past.html

Acknowledgements

- Some of this material was developed in collaboration with **Dr. Cem Kaner**, of the Florida Institute of Technology. See www.kaner.com and www.testingeducation.org.
- **Doug Hoffman** (www.softwarequalitymethods.com) has also contributed to and occasionally teaches from this material.
- Many of the ideas in this presentation were also inspired by or augmented by other colleagues including Jonathan Bach, Bret Pettichord, Brian Marick, Dave Gelperin, Elisabeth Hendrickson, Jerry Weinberg, Noel Nyman, and Mary Alton.
- Some of our exercises were introduced to us by Payson Hall, Jerry Weinberg, Ross Collard, James Lyndsay, Dave Smith, Earl Everett, Brian Marick, Cem Kaner and Joe McMahon.
- Many ideas were improved by students who took earlier versions of the class going back to 1996.

3

Assumptions About You

- You test software, or *any other complex human creation.*
- You have at least *some* control over the design of your tests and *some* time to create new tests.
- You are worried that your test process is spending too much time and resources on things that aren't important.
- **You test under uncertainty and time pressure.**
- **Your major goal is to find important problems quickly.**
- **You want to get *very good* at (software) testing.**

4

Exercise

How would you test…?

Premises of Rapid Testing

1. Software projects and products are relationships between people.
2. Each project occurs under conditions of uncertainty and time pressure.
3. Despite our best hopes and intentions, some degree of inexperience, carelessness, and incompetence is normal.
4. A test is an activity; it is performance, not artifacts.

Premises of Rapid Testing

5. Testing's purpose is to discover the status of the product and any threats to its value, so that our clients can make informed decisions about it.
6. We commit to performing credible, cost-effective testing, and we will inform our clients of anything that threatens that commitment.
7. We will not knowingly or negligently mislead our clients and colleagues or ourselves.
8. Testers accept responsibility for the quality of their work, although they cannot control the quality of the product.

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.

8

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 they don't want to fund it and they don't know how to do it.

More Work & Time (Cost)

Ponderous — *Slow, expensive, and easier*	*Exhaustive* — *Slow, **very** expensive, and difficult*
Slapdash — *Much faster, cheaper, and easier*	*Rapid* — *Faster, less expensive, still challenging*

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

It fulfills the mission of testing.

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

Better Thinking & Better Testing (Value)

9

Excellent Rapid Technical Work Begins with You

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?

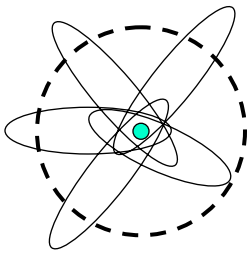
Can you let your teammates help you?

Are you ready to act, *right now*?

10

…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 your unique individual contribution



11

Exercise

The Triangle Program

What Is A Problem?

An problem is…

How Do We Recognize Problems?

An oracle is…

a heuristic principle or mechanism
by which
we recognize a problem

Learn About Heuristics

Heuristics are fallible, "fast and frugal" methods of solving problems, making decisions, or accomplishing tasks.

"The engineering method is the use of *heuristics* to cause the best change in a poorly understood situation within the available resources."

Billy Vaughan Koen
Discussion of the Method

Heuristics: Generating Solutions Quickly

- **adjective:**

  "serving to discover or learn."

- **noun:** "A **fallible** method for **solving a problem** or **making a decision**."

"Heuristic reasoning is not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution to the present problem."
- George Polya, *How to Solve It*

Oracles

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

"It works!"

really means...

"...it appeared at least once to meet some requirement to some degree."
"...uh, when I ran it."
"...on my machine."

Exercise

How Did You Recognize That Problem?

What makes a report useful?

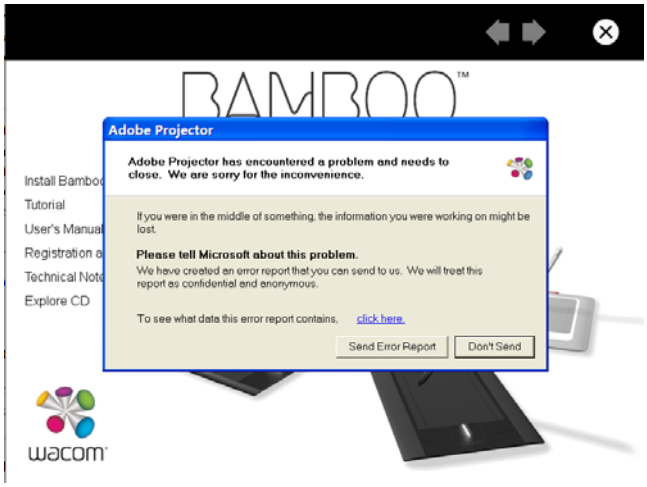
PEW

- Problem
  - What is the observed behaviour?
- Example
  - What would someone have to do to see it?
- Why
  - Why do we think it's a problem?
  - What's the *oracle*?

Bugs vs. Issues

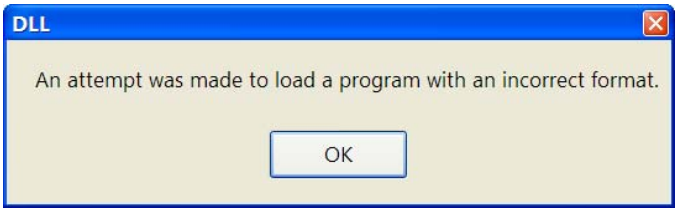
- Bug (n.)
  1. (formal) Any attribute or behaviour or attribute of the product that threatens its value to some person.
  2. (informal) Something that bugs some person who matters.
- Issue (n.)
  1. Anything that threatens the value of the testing (or the project, or the business).
  2. Anything that makes testing harder or slower.

Familiar Problems



If a product is* consistent *with problems we've seen before, we suspect that there might be a problem.

Explainability



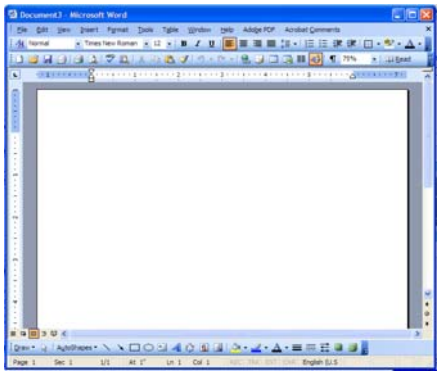
If a product is inconsistent with our ability to explain it, we suspect that there might be a problem.

World

If a product is inconsistent with the way the world works, we suspect that there might be a problem.

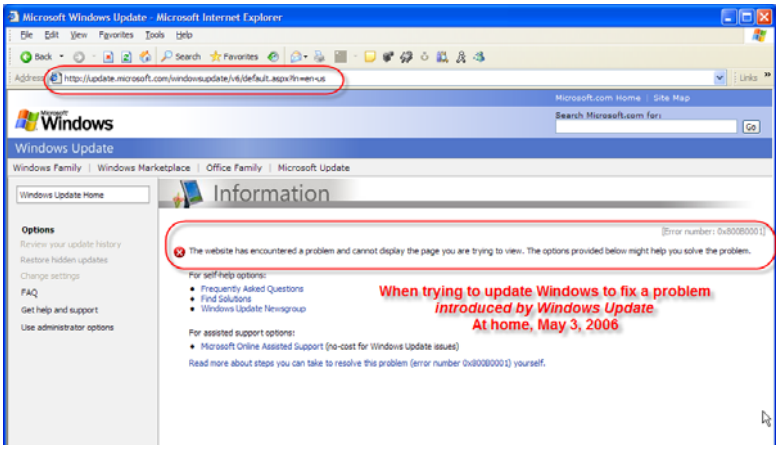
History

Okay, so how the #&@ do I print now?



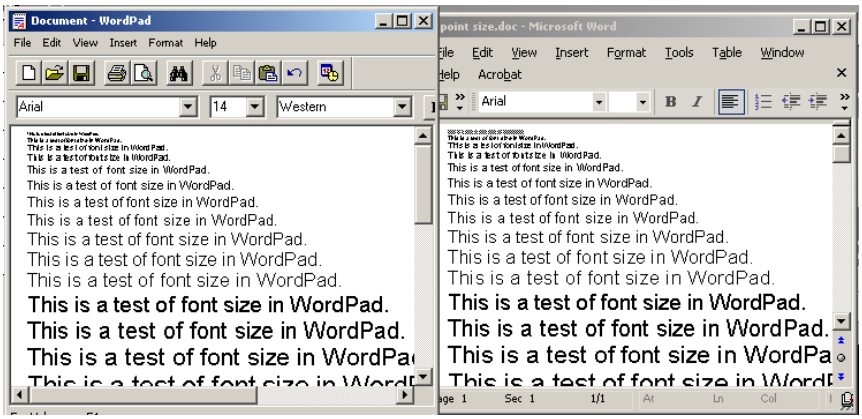
If a product is inconsistent with previous versions of itself, we suspect that there might be a problem.

Image



If a product is inconsistent with an image that the company wants to project, we suspect a problem.

Comparable Products



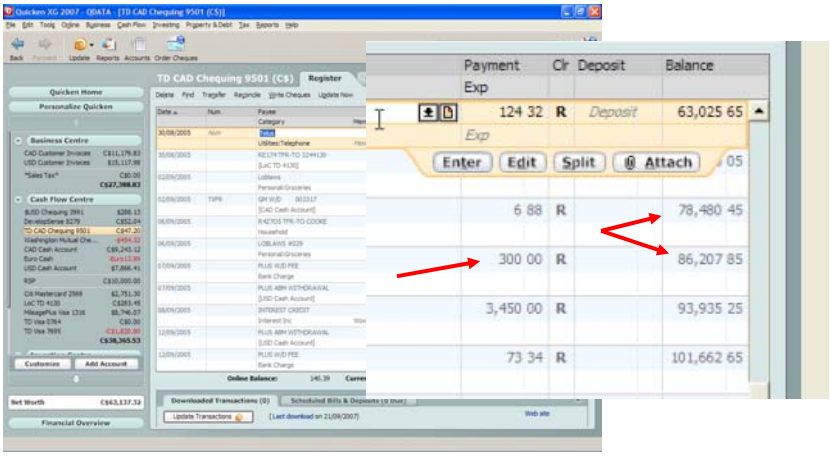
WordPad Word

When a product seems inconsistent with a comparable product, we suspect that there might be a problem.

Claims

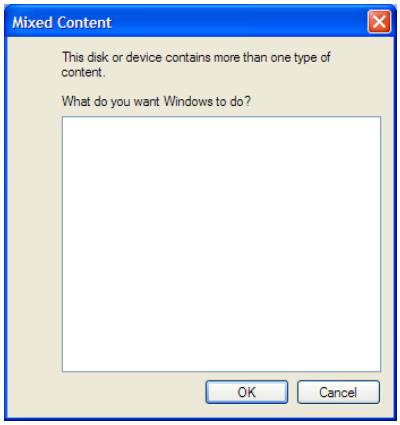
When a product is inconsistent with claims that important people make about it, we suspect a problem.

User Expectations



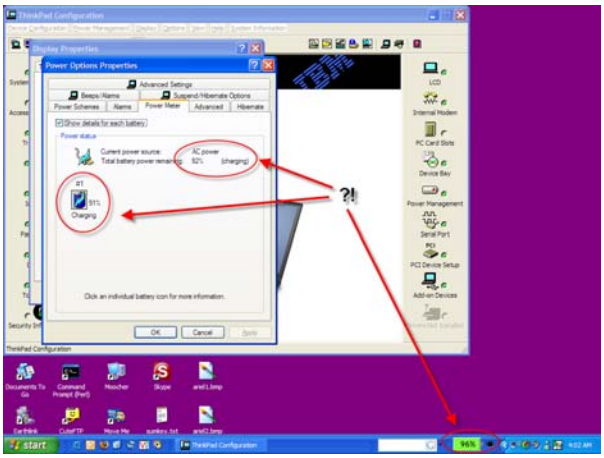
When a product is inconsistent with expectations that a reasonable user might have, we suspect a problem.

Purpose



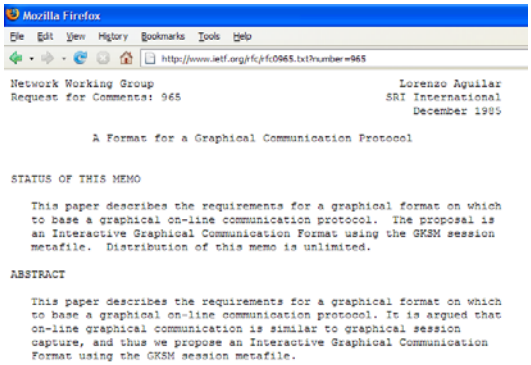
When a product is inconsistent with its designers' explicit or implicit purposes, we suspect a problem.

Product



When a product is inconsistent internally—as when it contradicts itself—we suspect a problem.

Statutes and Standards



Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.ietf.org/rfc/rfc0965.txt?number=965

Network Working Group
Request for Comments: 965

Lorenzo Aguilar
SRI International
December 1985

A Format for a Graphical Communication Protocol

STATUS OF THIS MEMO

This paper describes the requirements for a graphical format on which to base a graphical on-line communication protocol. The proposal is an Interactive Graphical Communication Format using the GKSM session metafile. Distribution of this memo is unlimited.

ABSTRACT

This paper describes the requirements for a graphical format on which to base a graphical on-line communication protocol. It is argued that on-line graphical communication is similar to graphical session capture, and thus we propose an Interactive Graphical Communication Format using the GKSM session metafile.

When a product is inconsistent with laws or widely accepted standards, we suspect a problem.

We like consistency when…

- the present version of the system *is consistent* with **past versions** of itself.
- the system *is consistent* with **an image** that the organization wants to project.
- the system *is consistent* with **comparable systems**.
- the system *is consistent* with **what important people say** it's supposed to be.
- the system *is consistent* with **what users seem to want**.
- each element of the system *is consistent* with comparable **elements in the same system**.
- the system *is consistent* with implicit and explicit purposes.
- the system *is consistent* with relevant laws or standards.

…unless *consistency* is a problem.

- We like it when the system ***is not consistent*** with patterns of familiar problems.
- We like it when the system ***is not consistent*** with things we can't explain.

All Oracles Are Heuristic

An oracle doesn't tell you that there IS a problem.
An oracle tells you that you *might be seeing a problem.*

Consistency heuristics rely on the quality of your models of the product and its context.

Rely solely on documented, anticipated sources of oracles, and your testing will likely be slower and weaker.

Train your mind in *patterns* of oracles and your testing will likely be faster and your coverage better.

How Do I Keep Track?
FEW HICCUPPS!

- Familiar Problems
- Explanaiblity
- World
- History
- Image
- Comparable Products
- Claims
- User Expectations
- Purpose
- Product
- Statutes

Remember…

For skilled testers,
good testing isn't just about
pass vs. fail.

For skilled testers,
testing is about
problem vs. no problem.

General Examples of Oracles
things that say “problem” or “no problem”

- A person whose opinion matters.
- An opinion held by a person who matters.
- A disagreement among people who matter.
- A reference document with useful information.
- A known good example output.
- A known bad example output.
- A process or tool by which the output is checked.
- A feeling, like confusion or annoyance.
- *A desirable consistency between related things.*

People

Mechanisms

Feelings

Principles

37

Oracle Cost and Value

- Some oracles are more *authoritative*
  - but less predictable
- Some oracles are more *consistent*
  - but maybe not up to date
- Some oracles are more *immediate*
  - but less reliable
- Some oracles are more *precise*
  - but less accurate
- Some oracles are more *accurate*
  - but less precise
- Some oracles are more *available*
  - but less authoritative
- Some oracles are *easier to interpret*
  - but more narrowly focused

Feelings As Heuristic Triggers For Oracles

- An emotional reaction is a trigger to attention and learning
- Without emotion, we don't reason well
  - See Damasio, *The Feeling of What Happens*
- When you find yourself mildly concerned about something, someone else could be *very* concerned about it
- Observe emotions to help overcome your biases and to evaluate significance

An emotion is a signal; consider looking into it

Oracles are Not Perfect And Testers are Not Judges

- You don't need to know FOR SURE if something is a bug; it's not your job to DECIDE if something is a bug.
- You do need to form a justified belief that it MIGHT be a threat to product value in the opinion of someone who matters.
- And you must be able to say why you think so; you must be able to cite good oracles… or else you will lose credibility.

MIP'ing ***VS.*** *Black Flagging*

40

What IS Coverage?

Coverage is “how much of the product we have tested.”

It’s the extent to which we have traveled over *some map* of the product.

...but what does it mean to "map" a product?
Talking about coverage means talking about
models

Models

- **A model is a heuristic idea, activity, or object…**

  such as an *idea in your mind*, a *diagram*, a *list of words*, a *spreadsheet*, a *person*, a *toy*, an *equation*, a *demonstration*, or a *program*

- **…that represents (literally, *re*-presents) another idea, activity, or object…**

  such as something complex that you need to work with or study

- **…whereby understanding something about the model may help you to understand or manipulate the thing that it represents.**

- A *map* is a model that helps to navigate across a terrain.
- *2+2=4* is a model for adding two apples to a basket that already has two apples.
- *Atmospheric models* help predict where hurricanes will go.
- A *fashion model* helps understand how clothing would look on actual humans.
- *Your beliefs about what you test are a model of what you test.*

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

Structure
Business Risk
Time
Technical Risk
Functions
Platform
Data
Operations

And each could be...

Intentional or... Accidental

One Way to Model Coverage: Product Elements (with Quality Criteria)

SFDIPOT—"San Francisco Depot"

Product Elements

- Structure
- Function
- Data
- Interfaces
- Platform
- Operations
- Time

Capability
Reliability
Usability

Charisma
Security
Scalability
Compatibility

Performance
Installability
Development

Quality Criteria

To test a *very simple* product meticulously,
***part* of a complex product meticulously,**
or to maximize test *integrity*…

1. Start the test from a *known* (clean) state.
2. Prefer *simple, deterministic* actions.
3. Trace test steps to a *specified model*.
4. Follow *established and consistent* lab procedures.
5. Make *specific* predictions, observations and records.
6. Make it *easy to reproduce* (automation may help).

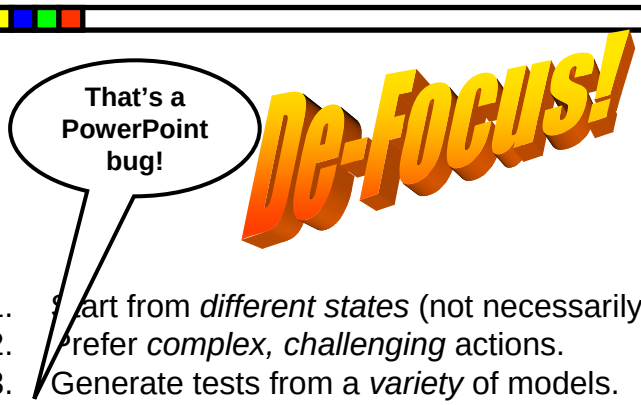
45

General Focusing Heuristics

- use test-first approach or unit testing for better *code* coverage
- work from prepared test coverage outlines and risk lists
- use diagrams, state models, and the like, and cover them
- apply specific test techniques to address particular coverage areas
- make careful observations and match to expectations

To do this *more rapidly*, make *preparation* and *artifacts* fast and frugal: leverage existing materials and avoid repeating yourself. Emphasize doing; relax planning. You'll make discoveries along the way!

To find *unexpected problems*,
***elusive problems* that occur in sustained field use,**
or more problems *quickly* in a complex product…



1. Start from *different states* (not necessarily clean).
2. Prefer *complex, challenging* actions.
3. Generate tests from a *variety* of models.
4. *Question* your lab procedures and tools.
5. Try to *see everything* with open expectations.
6. Make the test *hard to pass*, instead of easy to reproduce.

47

General Defocusing Heuristics

- diversify your models; intentional coverage in one area can lead to unintentional coverage in other areas—this is a Good Thing
- diversify your test techniques
- be alert to problems other than the ones that you're actively looking for
- welcome and embrace distraction
- do some testing that is *not* oriented towards a specific risk
- use high-volume, randomized automated tests

What About Quantifying Coverage Overall?

- A nice idea, but we don't know how to do it in a way that is consistent with *basic* measurement theory
  - If we describe coverage by counting test cases, we're committing reification error.
  - If we use percentages to quantify coverage, we need to establish what 100% looks like.
    - But we might do that with respect to *some specific* models.
  - Complex systems may display emergent behaviour.

How do we hang meaningful numbers on that stuff?

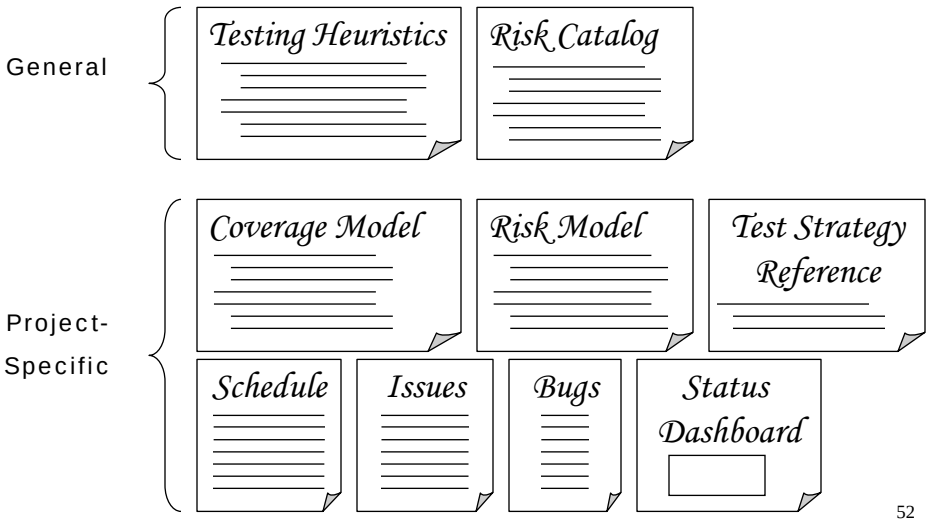
Extent of Coverage

- Smoke and sanity
  - Can this thing even be tested at all?
- Common, core, and critical
  - Can this thing do the things it *must* do?
  - Does it handle happy paths and regular input?
  - *Can* it work?
- Complex, harsh, extreme and exceptional
  - Will this thing handle challenging tests, complex data flows, and malformed input, etc.?
  - *Will* it work?

How Might We Organize, Record, and Report Coverage?

- automated tools (e.g. profilers, coverage tools)
- annotated diagrams and mind maps
- coverage matrices
- bug taxonomies
- Michael Hunter's You Are Not Done Yet list
- James Bach's Heuristic Test Strategy Model
  - described at www.satisfice.com
  - articles about it at www.developsense.com
- Mike Kelly's MCOASTER model
- coverage outlines and risk lists
- session-based test management
  - http://www.satisfice.com/sbtm

What Does Rapid Testing Look Like? *Concise Documentation Minimizes Waste*

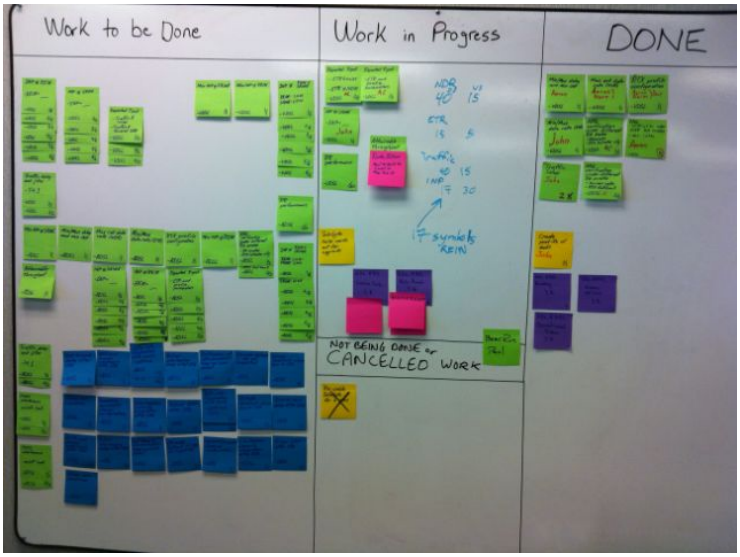


52

Rapid Testing Documentation

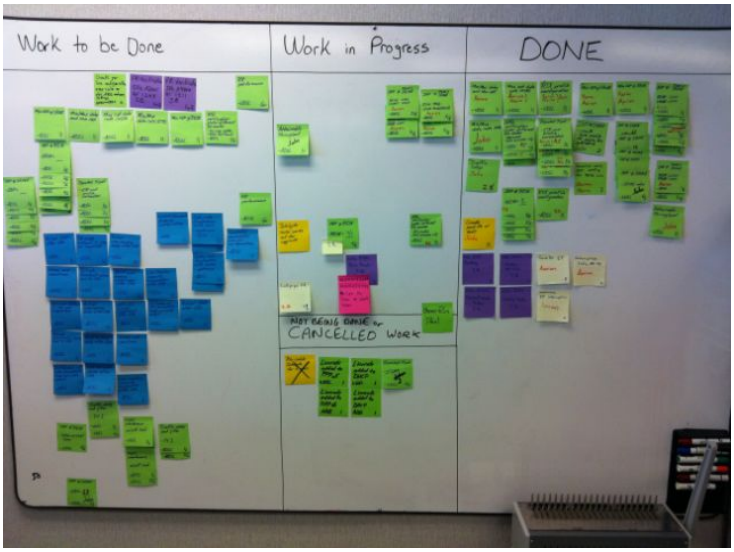
- Recognize
  - a requirements *document* is not *the requirements*
  - a test plan *document* is not *a test plan*
  - a test *script* is not a *test*
  - doing, rather than planning, produces results
- Determine where your documentation is on the continuum: product or tool?
  - Keep your *tools* sharp and lightweight
  - Obtain consensus from others as to what's necessary and what's excess in *products*
- Ask whether *reporting* test results takes priority over *obtaining* test results
  - note that in some contexts, it might
- Eliminate unnecessary clerical work

Visualizing Test Progress



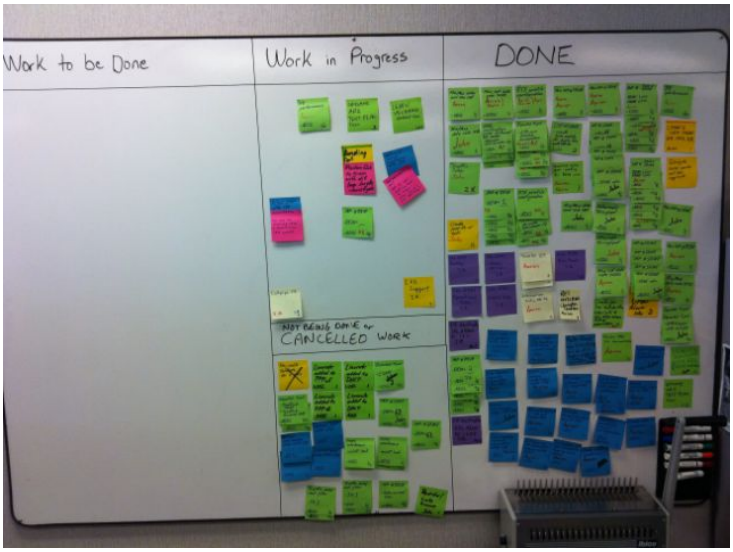
See "A Sticky Situation", Better Software, February 2012
http://www.developsense.com/articles/2012-02-AStickySituation.pdf

Visualizing Test Progress



See "A Sticky Situation", Better Software, February 2012
http://www.developsense.com/articles/2012-02-AStickySituation.pdf

Visualizing Test Progress



See "A Sticky Situation", Better Software, February 2012
http://www.developsense.com/articles/2012-02-AStickySituation.pdf

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?

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

Exploratory software testing is…

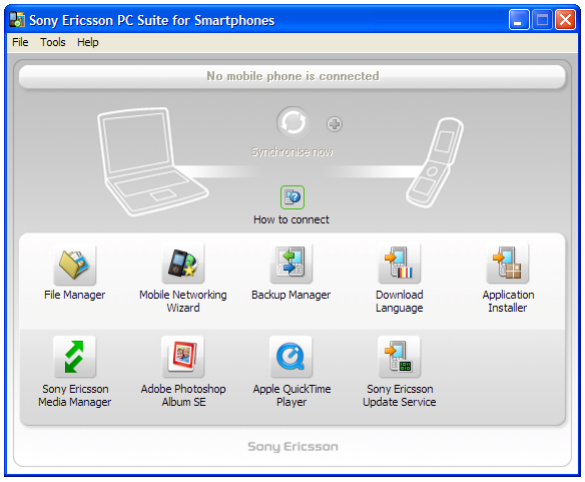
- a style of software testing
- that emphasizes the personal freedom and responsibility
- of the individual tester
- to continually optimize the value of his or her work
- by treating test design, test execution, test result interpretation, and test-related learning
- as mutually supportive activities
- that run in parallel
- throughout the project.

So 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

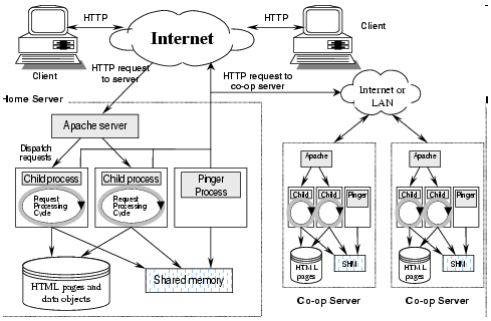
Why Exploratory Approaches?

- Systems are far more than collections of functions
- Systems typically depend upon and interact with many external systems



Why Exploratory Approaches?

- Systems are too complex for individuals to comprehend and describe
- Products evolve rapidly in ways that cannot be anticipated



In the future, developers will likely do more verification and validation at the unit level than they have done before.

Testers must explore, discover, investigate, and learn about the *system*.

Why Exploratory Approaches?

- Developers are using tools and frameworks that make programming more productive, but that may manifest more emergent behaviour.
- Developers are increasingly adopting unit testing and test-driven development.
- The traditional focus is on verification, validation, and confirmation.

The new focus must be on exploration, discovery, investigation, and learning.

Why Exploratory Approaches?

- We don't have time to waste
- preparing wastefully elaborate written plans
- for complex products
- built from many parts
- and interacting with many systems
- (many of which we don't understand…
- or control)
- where everything is changing over time
- and there's *so much learning* to be done
- and the *result*, not the plan, is paramount.

Exploratory Testing

The way we practice and teach it, exploratory testing…

- **IS NOT** "random testing" (or sloppy, or slapdash testing)
- **IS NOT** "unstructured testing"
- **IS NOT** procedurally structured
- **IS NOT** unteachable
- **IS NOT** unmanageable
- **IS NOT** scripted
- **IS NOT** a technique

- **IS** "ad hoc", in the dictionary sense, "to the purpose"
- **IS** structured and rigorous
- **IS** cognitively structured
- **IS** highly teachable
- **IS** highly manageable
- **IS** chartered
- **IS** an approach

What you do next is governed by what you're learning

Contrasting Approaches

Scripted Testing

- Is directed from elsewhere
- Is determined in advance
- Is about confirmation
- Is about controlling tests
- Emphasizes predictability
- Emphasizes decidability
- Like making a speech
- Like playing from a score

Exploratory Testing

- Is directed from within
- Is determined in the moment
- Is about investigation
- Is about improving test design
- Emphasizes adaptability
- Emphasizes learning
- Like having a conversation
- Like playing in a jam session

The tester's mind is in control, not the script.

Exploratory Testing IS Structured

- Exploratory testing, as we teach it, is a structured process conducted by a skilled tester, or by lesser skilled testers or users working under supervision.
- 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.

ET is a Structured Process

In excellent exploratory testing, one structure tends to dominate all the others:

The Testing Story

Exploratory testers construct a compelling story of their testing. It is this story that gives ET a backbone.

66

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…

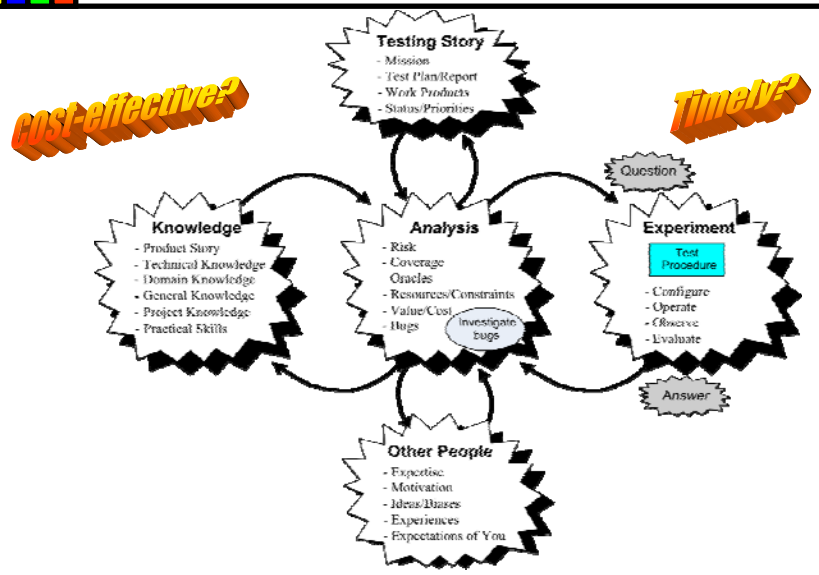
…what the risks and costs of testing are…

…how testable (or not) the product is…

…what you need and what you recommend.

67

One Story of Exploratory Testing



68

What does "taking advantage of resources" mean?

MIDTESTD

- Mission
  - *The problem we are here to solve for our customer.*
- Information
  - *Information about the product or project that is needed for testing.*
- Developer relations
  - *How you get along with the programmers.*
- Team
  - *Anyone who will perform or support testing.*
- Equipment & tools
  - *Hardware, software, or documents required to administer testing.*
- Schedule
  - *The sequence, duration, and synchronization of project events.*
- Test Items
  - *The product to be tested.*
- Deliverables
  - *The observable products of the test project.*

69

"Ways to test…"? *General Test Techniques*

FDSFSCURA

- Function testing
- Domain testing
- Stress testing
- Flow testing
- Scenario testing
- Claims testing
- User testing
- Risk testing
- Automatic checking

70

Exercise

WordPad Revisited

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

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

- Happy Path
- Tour the Product
  - *Sample Data*
  - *Variables*
  - *Files*
  - *Complexity*
  - *Menus & Windows*
  - *Keyboard & Mouse*
- Interruptions
- Undermining
- Adjustments
- Dog Piling
- Continuous Use
- Feature Interactions
- Click on Help

72

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

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

- Input Constraint Attack
- Click Frenzy
- Shoe Test
- Blink Test
- Error Message Hangover
- Resource Starvation
- Multiple Instances
- Crazy Configs
- Cheap Tools

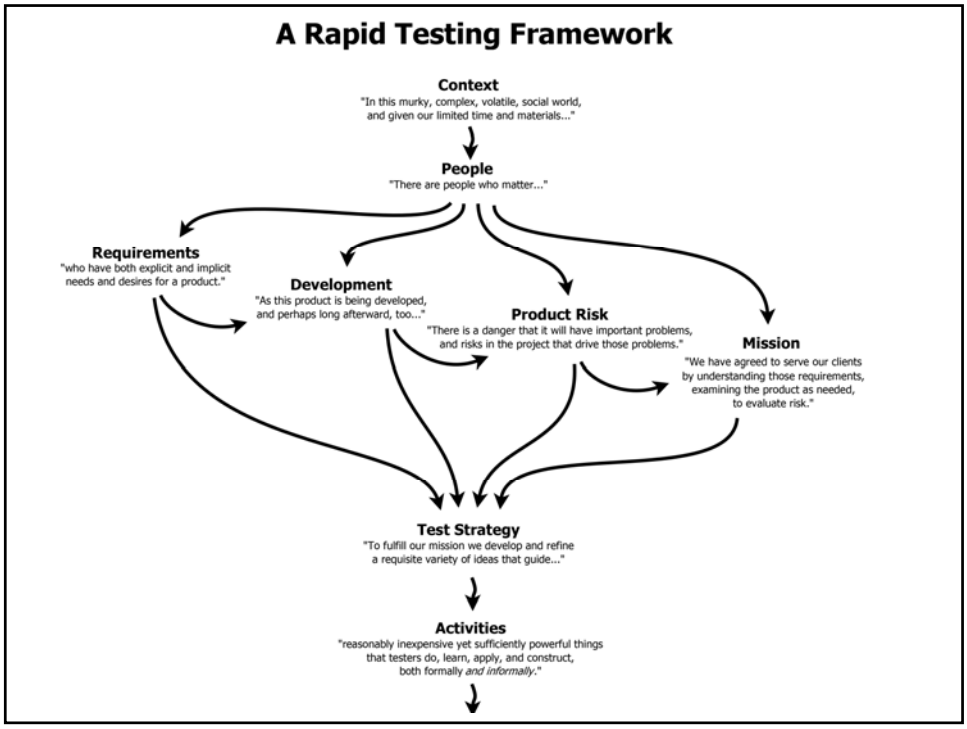
73

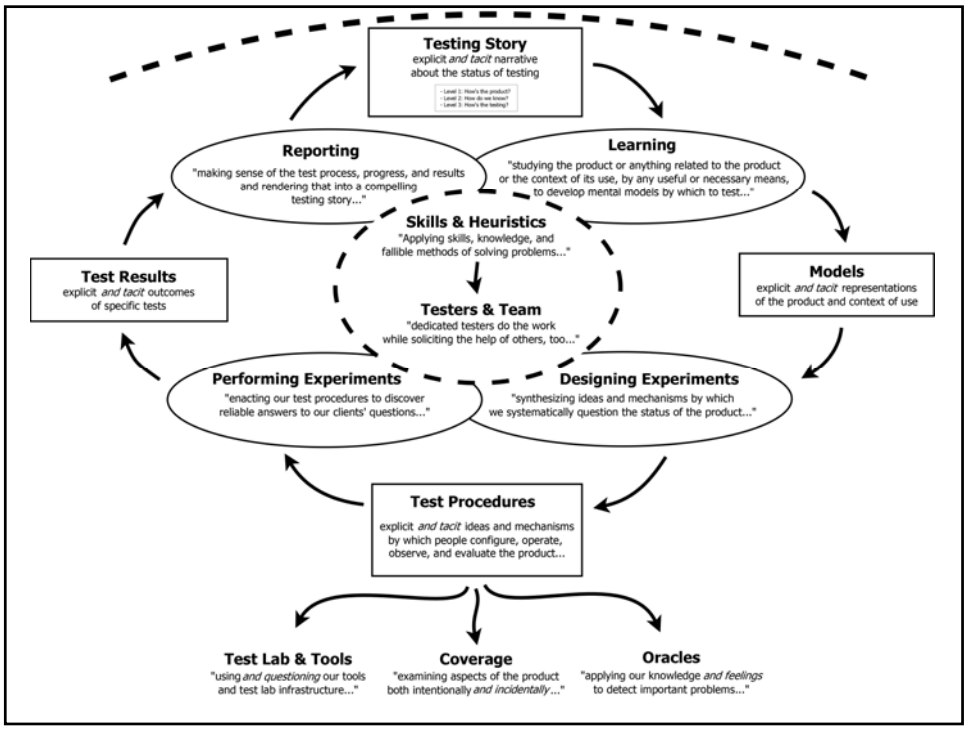
Touring the Product:
Mike Kelly's FCC CUTS VIDS

- Feature tour
- Complexity tour
- Claims tour
- Configuration tour
- User tour
- Testability tour
- Scenario tour
- Variability tour
- Interoperability tour
- Data tour
- Structure tour

Create your own list!

http://michaeldkelly.com/blog/2005/9/20/touring-heuristic.html


A Rapid Testing Framework
Context
"In this murky, complex, volatile, social world, and given our limited time and materials..."
People
"There are people who matter..."
Requirements
"who have both explicit and implicit needs and desires for a product."
Development
"As this product is being developed, and perhaps long afterward, too..."
Product Risk
"There is a danger that it will have important problems, and risks in the project that drive those problems."
Mission
"We have agreed to serve our clients by understanding those requirements, examining the product as needed, to evaluate risk."
Test Strategy
"To fulfill our mission we develop and refine a requisite variety of ideas that guide..."
Activities
"reasonably inexpensive yet sufficiently powerful things that testers do, learn, apply, and construct, both formally and informally."


Testing Story
explicit and tacit narrative about the status of testing
Reporting
"making sense of the test process, progress, and results and rendering that into a compelling testing story..."
Learning
"studying the product or anything related to the product or the context of its use, by any useful or necessary means, to develop mental models by which to test..."
Skills & Heuristics
"Applying skills, knowledge, and fallible methods of solving problems..."
Testers & Team
"dedicated testers do the work while soliciting the help of others, too..."
Test Results
explicit and tacit outcomes of specific tests
Models
explicit and tacit representations of the product and context of use
Performing Experiments
"enacting our test procedures to discover reliable answers to our clients' questions..."
Designing Experiments
"synthesizing ideas and mechanisms by which we systematically question the status of the product..."
Test Procedures
explicit and tacit ideas and mechanisms by which people configure, operate, observe, and evaluate the product...
Test Lab & Tools
"using and questioning our tools and test lab infrastructure..."
Coverage
"examining aspects of the product both intentionally and incidentally..."
Oracles
"applying our knowledge and feelings to detect important problems..."

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.

77

Resources

- http://www.satisfice.com
- http://www.developsense.com