

A Master Class in  
Exploratory Testing

Michael Bolton  
DevelopSense  
St. Petersburg, Russia  
November 2010

KEY IDEA

*How do you test a whole product?*

**Use a diversified risk-based strategy.**

2

Test Strategy

- **Strategy:** “The set of ideas that guide your **test design.**”
- **Logistics:** “The set of ideas that guide your **application of resources** to fulfilling the test strategy.”
- **Plan:** “The set of ideas that guide your **test project.**”
- A good test strategy is:
  - Product-Specific
  - Risk-focused
  - Diversified
  - Practical

**plan = strategy + logistics**

3

Visual

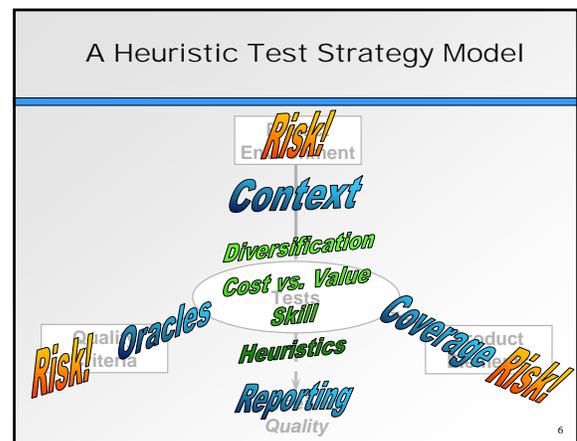
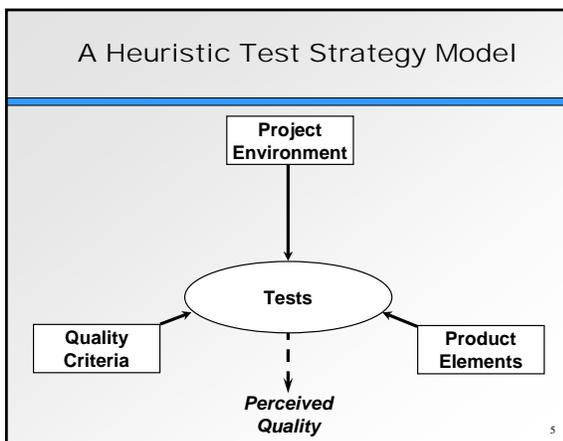
One way to make a strategy...

1. Learn the product.
2. Think of important potential problems.
3. Think of how to search the product for *those problems*.
4. Think of how to search the product, *in general*.

Think of ways that:

- will take advantage of the *resources* you have.
- comprise a *mix* of different techniques.
- comprise something that you really *can* actually do.
- serve the *specific* mission you are expected to fulfill.

4



### Project Environment Ways to Understand Our Context

**"CIDTESTD -- Mother Approved"**

- Customers
  - Anyone who is a client of the test project.
- 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.

7

### Quality Criteria Identifying Value and Threats To It

**CRUSSPIC STMP**

- Capability
- Reliability
- Usability
- Security
- Scalability
- Performance
- Installability
- Compatibility
- Supportability
- Testability
- Maintainability
- Portability
- Localizability

Many test approaches focus on Capability (functionality) and underemphasize the other criteria 8

### Product Elements Ways to Model and Cover The Product

**"SDFPOT - San Francisco DePOT"**

- Structure
  - What are the pieces and how do they fit together?
- Function
  - What does the product do?
- Data
  - What does the product do things to?
- Platform
  - What does the product depend upon?
- Operations
  - How do people actually use the program?
- Time
  - How is the product affected by time?

9

### Test Techniques General Ways to Test

**FDSFSCURA**

- Function testing
  - Test what it does
- Domain testing
  - Divide and conquer the data
- Stress testing
  - Overwhelm or starve the product
- Flow testing
  - Do one thing after another after another
- Scenario testing
  - Test to a compelling story

10

### Test Techniques General Ways to Test

**FDSFSCURA**

- Claims testing
  - Test everything that people say it should do
- User testing
  - Involve the users (or systematically simulate them)
- Risk testing
  - Imagine a problem, and then look for it
- Automatic testing
  - Perform zillions of tests, aided by machines

11

### Thirty-Six Testing Heuristics

**"cidtestdsfdpotcrusspicstmpifdsfscura"**

Customers	Structures	Capability	Function testing
Information	Functions	Reliability	Domain testing
Developer relations	Data	Usability	Stress testing
Team	Platforms	Security	Flow testing
Equipment & tools	Operations	Scalability	Scenario testing
Schedule	Time	Performance	Claims testing
Test Items		Installability	User testing
Deliverables		Compatibility	Risk testing
		Supportability	Automatic testing
		Testability	
		Maintainability	
		Portability	
		Localizability	

Project Environment

Product Elements

Quality Criteria

Test Techniques

### How much is enough? *Diverse Half-Measures*

- There is no single technique that finds all bugs.
- We can't do any technique perfectly.
- We can't do all conceivable techniques.

Use "**diverse half-measures**"-- lots of different points of view, approaches, techniques, even if no one strategy is performed completely.

13

### The Four-Part Risk Story

- Some *person*
- might suffer *harm or loss*
- because of a *vulnerability* in the product
- triggered by some *threat*.

Excellent **checking** reduces risks of unexpected implementation behaviour and of change.

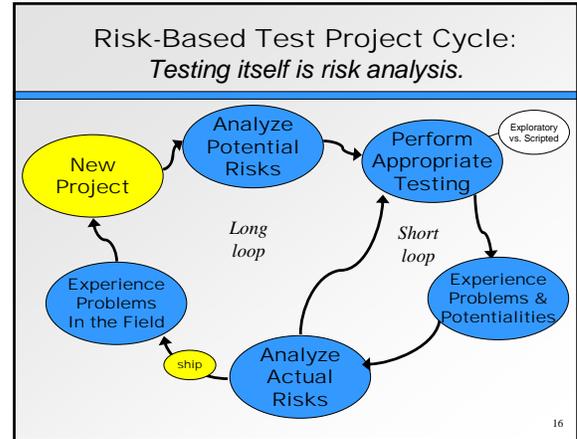
Excellent (risk-based) **testing** is less about anticipating and calculating, and more about learning.

### Value (or Risk) as a Simplifying Factor *Find problems that matter*

- In general it can vastly simplify testing if we focus on whether the product has a problem that matters, rather than whether the product merely satisfies all relevant standards.
- Effective testing requires that we understand standards as they relate to how our clients value the product.

Instead of thinking pass vs. fail, think problem vs. no problem.

15



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

## FOCUS!

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 helps).

17

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

## De-FOCUS!

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.

18

### Can tests be repeated?

- You can't be certain that you control all the factors that might matter to fulfill the purpose of a test.
- So, to "repeat a test" means that you believe you are repeating *some part of a test that matters*, while other parts of that test may not be repeated.
- Even if you repeat "just 1% of a test", it may be fair to say that you have repeated that test *in every way that matters*.

***Exact repetition is not an option.***

19

### Should tests be repeated?

- Consider cost vs. value.
- Consider what could be gained from some of the many tests you have not yet *ever performed*.
- To test is to question:
  - Why ask the same question again?
  - Why ask the same question again?
  - Why ask the same question again?
  - Why ask the same question again?
- You should know why, and why *not*, to repeat tests.

20

### Don't Repeat Tests: Debate

- I will now argue why, any time you are tempted to repeat even part of a test, you should not.
- I want **you** to argue the other side: provide reasons why it might be a *good thing* to repeat a test.

### The Fallacy of Repeated Tests: Clearing Mines

◆ mines

### Totally Repeatable Tests Won't Clear the Minefield

◆ mines    ● fixes

### Variable Tests are Therefore More Effective

◆ mines    ● fixes

### Reasons to Repeat

<ol style="list-style-type: none"> <li>1. Recharge</li> <li>2. Retry</li> <li>3. Intermittence</li> <li>4. Mutation</li> <li>5. Benchmark</li> <li>6. Importance</li> <li>7. Inexpensiveness</li> <li>8. Mandated</li> <li>9. Enoughness</li> <li>10. Avoidance/Indifference</li> </ol>	<span style="font-size: 3em;">}</span>	<h2 style="margin: 0;">Technical</h2>           <h2 style="margin: 0;">Business</h2>
---	--	---

### Why Do You Think You Need to Repeat Tests?

- You need to repeat tests to detect unexpected and undesired change, yet...
- If your regression tests are consistently revealing bugs, those bugs are only symptoms of a larger problem
- The larger problem is that people are making changes without being aware of the consequences of the changes
- A large enough regression suite is *incomprehensible*

Wordpad

### Exploiting Variation To Find More Bugs

- **Micro-behaviors:** Unreliable and distractible humans make each test a little bit new each time through.
- **Randomness:** Can protect you from unconscious bias.
- **Data Substitution:** The same actions may have dramatically different results when tried on a different database, or with different input.
- **Platform Substitution:** Supposedly equivalent platforms may not be.
- **Timing/Concurrency Variations:** The same actions may have different results depending on the time frame in which they occur and other concurrent events.
- **Scenario Variation:** The same functions may operate differently when employed in a different flow or context.
- **State Pollution:** Hidden variables of all kinds frequently exert influence in a complex system. By varying the order, magnitude, and types of actions, we may accelerate state pollution, and discover otherwise rare bugs.
- **Sensitivities and Expectations:** Different testers may be sensitive to different factors, or make different observations. The same tester may see different things at different times or when intentionally shifting focus to different things.

27

### KEY IDEA

How do you invent the right tests,  
at the right time?

Evolve them with an exploratory strategy.

28

### Exploratory Testing Is...

- an *approach* to software testing...  
*(applicable to any test technique)*
- that emphasizes the personal freedom and responsibility of each tester to continually optimize the value of his work...  
*(optimize how?)*
- by treating learning, test design and test execution as mutually supportive activities that run in parallel throughout the project.

29

### Questions About Exploration...

arrows and cycles

(value seeking)

Questions About Scripts...  
*arrows and cycles*

---

What happens when the unexpected happens during a script?

Where do scripts come from?

Will everyone follow the same script the same way?

What do we do with what we learn?

(task performing)

Answers About Scripts...  
*arrows and cycles*

---

What happens when the unexpected happens during a script?

Where do scripts come from?

Will everyone follow the same script the same way?

What do we do with what we learn?

(task performing)

Questions About Exploration...  
*arrows and cycles*

---

Where does exploration come from?

What happens when the unexpected happens during exploration?

What do we do with what we learn?

Will everyone explore the same way?

(value seeking)

Exploration is Not Just Action  
*arrows and cycles*

---

Exploratory behavior acts on itself.  
Scripted behavior does not.

You can put them together!  
*arrows and cycles*

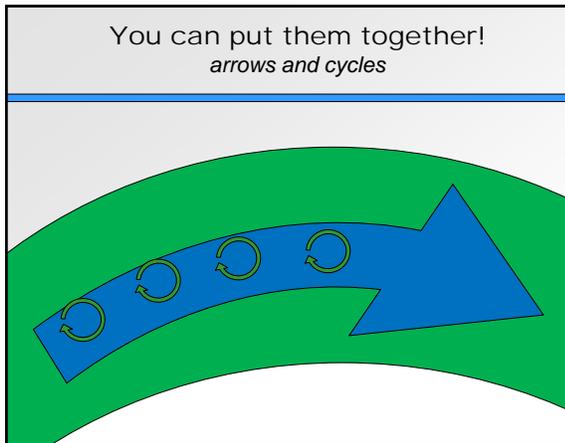
---

**EXPLORATORY**  
(value seeking)

**SCRIPTED**  
(task performing)

You can put them together!  
*arrows and cycles*

---



- How do you do this well?
- With...
    - Skills
    - Heuristics
    - Diversity
    - Leadership
    - Good notes and automatic logging
    - ...oh and sometimes... with *scripting*.

- IP Address
- ET is a Structured Process
- Exploratory testing, as we talk about 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
- In other words, it's not "random", but systematic.
- See "Exploratory Dynamics" in the Appendices.
- 39

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.*

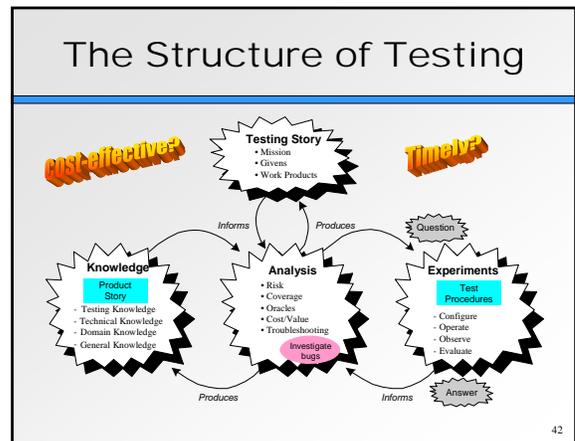
40

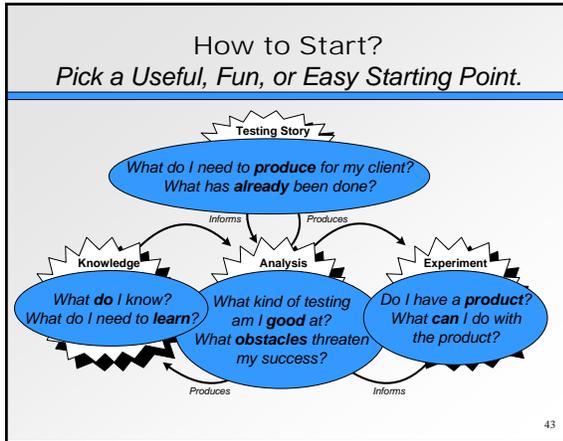
To test is to compose, edit, narrate, and justify two parallel stories.

You must tell a story about the product...  
...about how it failed, and how it *might* fail...  
...in ways that matter to your various clients.

But also tell a story about testing...  
...how you configured, operated and observed it...  
...about what you haven't tested, yet...  
...or won't test, at all...  
...and about why what you did was good enough.

41





### Boundary Testing

### How to Find an Elusive Bug

# DE-FOCUS!

1. Look over your recent tests and find a pattern there.
2. With your next few tests, *violate* the old pattern.
3. Prefer **MFAT** (**m**ultiple **f**actors **a**t a time).
4. Broaden and vary your observations.

44

### Dice Game

### What to Do if You are Confused

# FOCUS!

1. *Simplify* your tests.
2. *Conserve* states.
3. Frequently *repeat* your actions.
4. Frequently return to a *known* state.
5. Prefer **OFAT** heuristic (**o**ne factor **a**t a time).
6. Make *precise* observations.

45

### Test Design

### Testing to Learn vs. Testing to Search

- Testing (primarily) to Learn
  - Forming a mental model of the product.
  - Learning what the product is supposed to do.
  - Inventorying the product elements you may want to test.
  - Looking at consistency relationships and trying various oracles.
  - Generating test data.
  - Considering testability and trying potentially useful tools.
  - Experimenting with different ways of testing it.
  - Reporting bugs that you find.
- Testing (primarily) to Search
  - Using your detailed product knowledge, and any relevant tools, to systematically exercise the product in every important way.
  - Using careful observation and good oracles to detect important bugs.
  - Modifying and diversifying your tests to make them more powerful.
  - Reporting bugs that you find.

46

### Test Design Motivations

### Testing to Learn vs. Testing to Search

#### Compare these Situations:

- L** **S** - Starting a new project.
- L** **S** - Reflective thought or research about test design
- L** **S** - Seeing a new feature for the first time.
- L** **S** - Smoke or sanity tests on a new build
- L** **S** - Testing a product deeply to reveal important bugs.
- L** **S** - Investigating a particular bug.
- L** **S** - Re-testing a product after a change.
- S** - Repeated execution of detailed procedural test scripts.

47

### Lyndsay Machine

### High Learning ET: Explore These Things

- **Composition**
  - **Affordances:** Interfaces to the product.
  - **Dimensions & Variables:** Product space and what changes within it.
  - **Relationships & Interactions:** functions that cooperate or interfere.
  - **Navigation:** Where things are and how to get to them.
- **Conformance**
  - **Benefits:** What the product is good for-- when it has no bugs in it.
  - **Consistencies:** Fulfillment of logical, factual, and cultural expectations.
  - **Oracles:** Specific mechanisms or principles by which you can spot bugs.
  - **Bugs and Risks:** Specific problems and potential problems that matter.
- **Context (of the Product)**
  - **History:** Where the product has been, and how it came to be.
  - **Operations:** Its users and the conditions under which it will be used.
- **Conditions (of Testing)**
  - **Attitudes:** What your clients care about and what they want from you.
  - **Complexities & Challenges:** Discover the hardest things to test.
  - **Resources:** Discover tools and information that might help you test.

48

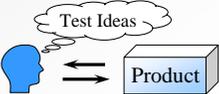
### Contrasting Approaches

In *scripted* testing, tests are first designed and recorded. Then they may be executed at some later time or by a different tester.



**vs.**

In *exploratory* testing, tests are designed and executed at the same time, and they are not necessarily recorded, but may be.



49

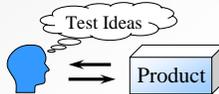
### Contrasting Approaches

Scripted testing is about *controlling test execution*.



**vs.**

Exploratory testing is about *improving test design*.



50

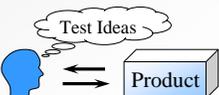
### Contrasting Approaches

Scripted testing is like *making a prepared speech*. It is guided by pre-conceived ideas.



**vs.**

Exploratory testing is like *having a conversation*. It is self-guided.



51

### Exploratory Skills and Tactics

- Modeling
- Resourcing
- Questioning
- Chartering
- Observing
- Manipulating
- Collaborating
- Generating & Elaborating
- Overproduction & Abandonment
- Abandonment & Recovery
- Refocusing
- Alternating
- Branching & Backtracking
- Conjecturing
- Recording
- Reporting

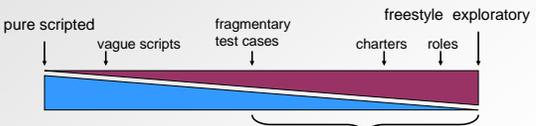
52

### Exploratory Dynamics

Testing vs. resting	Warming up vs. cruising vs. cooling down
Solo work vs. team effort	Your ideas vs. other peoples' ideas
Testing vs. touring	Individual tests vs. general lab procedures and infrastructure
Feature vs. feature	
Working with the product vs. working with the developer	Test design vs. execution
	Product vs. project
Current version vs. old versions	Requirement vs. requirement
Working with the product vs. reading about the product	Data gathering vs. data analysis
	Doing vs. thinking
Careful vs. quick	Doing vs. describing
Lab conditions vs. field conditions	Coverage vs. oracles

53

### Mixing Scripting and Exploration



*When we say "exploratory testing" and don't qualify it, we mean anything on the exploratory side of this continuum.*

54

### Blending Scripted & Exploratory

- **Generic scripts:** specify general test procedures and apply them to different parts of a test coverage outline.
- **Vague scripts:** specify a test step-by-step, but leave out any detail that does not absolutely need to be pre-specified.
- **Improvisation:** have scripts, but encourage deviation from them, too.
- **Fragmentary cases:** specify tests as single sentences or phrases.
- **Test Coverage Outline:** use outline of product elements and have tester construct tests from it on the fly.
- **Risk Catalog:** specify types of problems to look for, then construct tests on the fly to find each one.
- **Exploratory Charters:** specify 90 minutes of testing in two sentences or less.
- **Roles:** Give each tester a standing role to test a certain part of the product. Leave the rest up to them.
- **Heuristics:** Train exploratory testers to use standardized test design heuristics.
- **SBTM:** Consider using Session-Based Test Management, a formalized method of exploratory test management. (<http://www.satisfice.com/sbtm>).

55

### Test Design and Execution

*Achieve excellent test design by exploring different test designs while actually testing*

*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.*

56

### Allow some *disposable time* Self-management is good!

- How often do you account for your progress?



*Every minute?  
Every hour?  
Every day?*

- If you have *any autonomy at all*, you can risk investing *some time* in
  - learning
  - thinking
  - refining approaches
  - better tests

57

### Allow some *disposable time*

- ☹ If it turns out that you've made a bad investment...*oh well*
- 😊 If it turns out that you've made a *good* investment, you might have
  - learned something about the product
  - invented a more powerful test
  - found a bug
  - done a better job
  - surprised and impressed your manager

58

### “Plunge in and Quit” Heuristic

Whenever you are called upon to test something very complex or frightening, plunge in!  
After a little while, if you are very confused or find yourself stuck, quit!

- This benefits from *disposable time*– that part of your work not scrutinized in detail.
- Plunge in and quit means you can start something without *committing* to finish it successfully, and therefore *you don't need a plan*.
- Several cycles of this is a great way to *discover* a plan.

59

### Exploratory Branching: *Distractions are good!*

*New test ideas occur continually during an ET session.*

60

### "Long Leash" Heuristic

Let yourself be distracted...

'cause you never know what you'll find...

but periodically take stock of your status against your mission

61

### Important Questions

## Why run that test?

- Variations:
  - Why are you planning to run that test?
  - Why are you running that test *right now*?
  - Why did you run that test?

### Important Questions

## Why NOT run THAT test?

- Variations:
  - Why aren't you planning to run that test?
  - Why aren't you running that test *right now*?
  - Why didn't you run that test?

### Important Questions

## Why didn't you find that bug?

- Variations:
  - Why didn't you find that bug earlier?
  - Why did you apparently ignore that requirement?

### Important Questions

## Why do you think that's a bug?

- Variations:
  - Why do you say that this isn't working properly?
  - What requirement is being left unfulfilled here?
  - Why do you think that's a requirement?
  - For whom might this be a problem?
  - Do you think a user would ever do that?

### Even more generally...

## Why are you doing this?

- Variations:
  - Why are you not doing that?
  - How does this test relate to a requirement?
  - How does this test relate to a risk?
  - How does this test relate to your mission?

## 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?*

## Much More Traceability

1. Product traces to specifications.
2. Specifications trace to standards.
3. Test sessions trace to product versions.
4. Test sessions trace to specifications.
5. Test sessions trace to logs which trace to product, playbook and specifications.
6. Test sessions trace to charters and charters to playbook.
7. Playbook traces to standards.
8. Playbook traces to specifications.
9. Playbook traces to risks which trace to specifications...
10. Tests trace to risk...
11. Tests trace to implicit requirements...
12. Tests trace to other tests...

## Vocabulary

- **structure**
  - that which forms the unchanging parts and relationships of a system; “that which remains”
- **logic**
  - A means of convincing or proving e.g. “the logic of the situation”, the facts which dictate what action is rationally to be taken
- **narration**
  - telling a story that fits in time
- **framing**
  - placing the test, via logic and narrative, in logical relationship with the structures that inform it

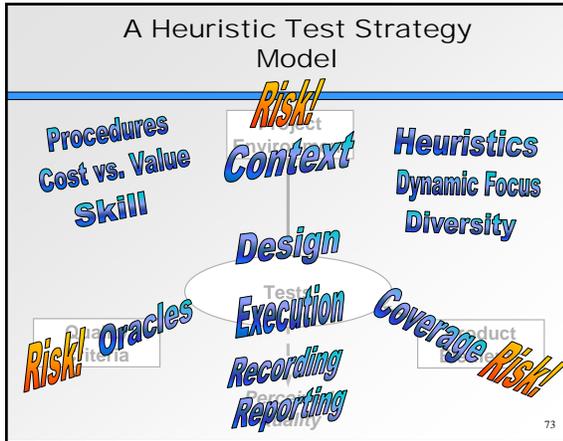
## Vocabulary

- **galumphing**
  - exploiting variability by doing something in a deliberately over-elaborate way
  - adding lots of unnecessary but inert actions that are inexpensive and shouldn't (in theory) affect the test outcome
  - ...but sometimes they do affect it!

## To test is to compose, edit, narrate, and justify two parallel stories.

*You must tell a story about the product...*  
...about how it failed, and how it *might* fail...  
...in ways that matter to your various clients.

*But also tell a story about testing...*  
...how you configured, operated and observed it...  
...about what you haven't tested, yet...  
...or won't test, at all...  
...and about why what you did was good enough.



What if you have an unframed test?

**Try framing it!**

But if you can't do it perfectly, or right away, that might be okay. Why?

How can you justify an unframed test?

**All of the hidden frames!**