




Test Framing


Michael Bolton
DevelopSense
<http://www.developsense.com>
STAR East
May, 2011



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?

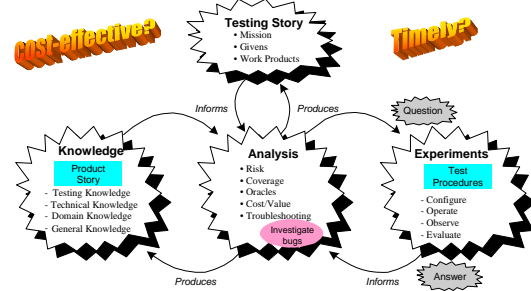
To test is to compose, edit, narrate, and justify two parallel stories.

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.

7

One Structure of Testing



8

What is test framing?

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

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

Framing ~ = Traceability

- Framing is, in essence, traceability...
- ...but typically we see an impoverished view of traceability: between *tests* and requirements *documents*—explicit requirements.
- *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...

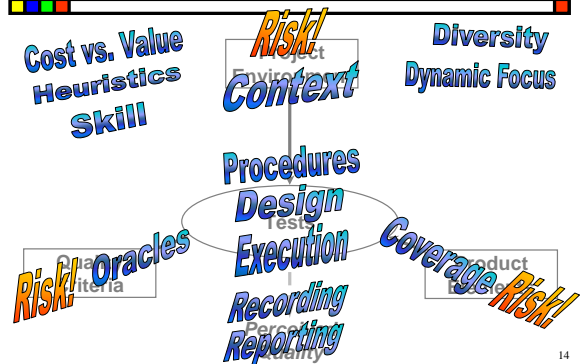
How Do We Know What "Is"?

We see the signs!

"If I see X, then probably Y, because probably A, B, C, D, etc."

- THIS CAN FAIL:
 - Getting into a car– oops, not my car.
 - Bad driving– Why?
 - Bad work– Why?
 - Ignored people at my going away party– Why?
 - I can never find the sugar– Why?
 - Ordered orange juice at seafood restaurant– waitress misunderstood

A Heuristic Test Strategy Model



14

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.

15

Quality Criteria Identifying Value and Threats To It

CRUSSPIC STMPL

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

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

16

Product Elements Ways to Model and Cover The Product

"SFDPOT - 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 does the product interact with time?

17

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

18

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*

19

Thirty-Six Testing Heuristics

"cidtestdsfdpotcrusspicstmpfldsfscura"

| | | | |
|----------------------------|-------------------------|-------------------------|------------------------|
| 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 | Product Elements | Compatibility | Risk testing |
| | | Supportability | Automatic testing |
| | | Testability | |
| | | Maintainability | Test Techniques |
| | | Portability | |
| | | Localizability | |
| Project Environment | | Quality Criteria | |

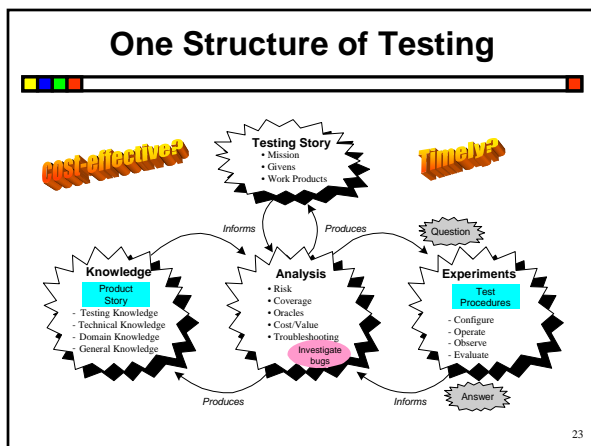
What if you have an unframed test?

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!



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).

To find *unexpected problems*,
elusive problems that may occur in the field,
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.

Galumphing

Exploiting Variability

- 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
- sometimes they do affect it!

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.