

Black Box Software Testing

2004 Academic Edition

PART 17 -- EXPLORATORY TESTING

by

Cem Kaner, J.D., Ph.D.

Professor of Software Engineering

Florida Institute of Technology

and

James Bach

Principal, Satisfice Inc.

These notes are partially based on research that was supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Kaner & Bach grant permission to make digital or hard copies of this work for personal or classroom use, including use in commercial courses, provided that (a) Copies are not made or distributed outside of classroom use for profit or commercial advantage, (b) Copies bear this notice and full citation on the front page, and if you distribute the work in portions, the notice and citation must appear on the first page of each portion. Abstracting with credit is permitted. The proper citation for this work is "Black Box Software Testing (Course Notes, Academic Version, 2004) www.testingeducation.org", (c) Each page that you use from this work must bear the notice "Copyright (c) Cem Kaner and James Bach, kaner@kaner.com", or if you modify the page, "Modified slide, originally from Cem Kaner and James Bach", and (d) If a substantial portion of a course that you teach is derived from these notes, advertisements of that course should include the statement, "Partially based on materials provided by Cem Kaner and James Bach." To copy otherwise, to republish or post on servers, or to distribute to lists requires prior specific permission and a fee. Request permission to republish from Cem Kaner, kaner@kaner.com.

Exploratory Testing

- Simultaneously:
 - Learn about the product
 - Learn about the market
 - Learn about the ways the product could fail
 - Learn about the weaknesses of the product
 - Learn about how to test the product
 - Test the product
 - Report the problems
 - Advocate for repairs
 - *Develop new tests based on what you have learned so far.*

Exploratory Testing

- Exploratory testing is not a testing technique. It's a way of thinking about testing.

- Any technique can be used in any exploratory way.
- Any competent tester does *some* exploratory testing.

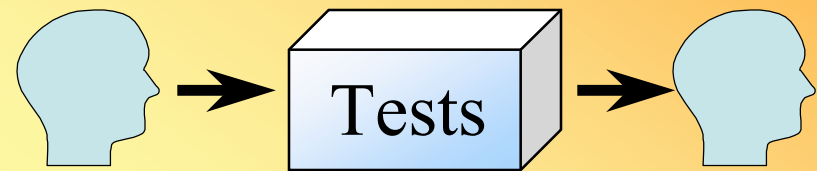
- Example -- bug regression

Report a bug, the programmer claims she fixed the bug, so you test the fix.

- Start by reproducing the steps you used in the bug report to expose the failure.
- Then vary your testing to search for side effects.
 - » These variations are not predesigned. This is an example of chartered exploratory testing.

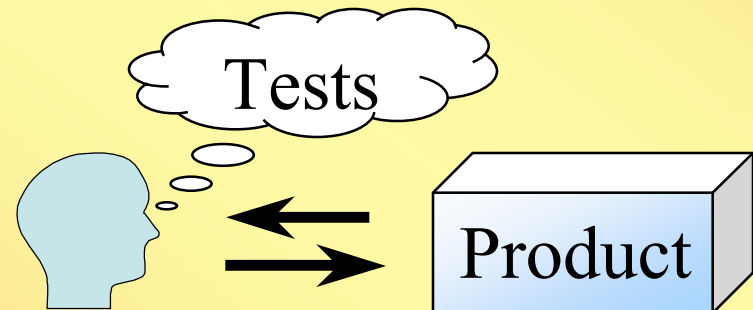
Contrasting Approaches

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



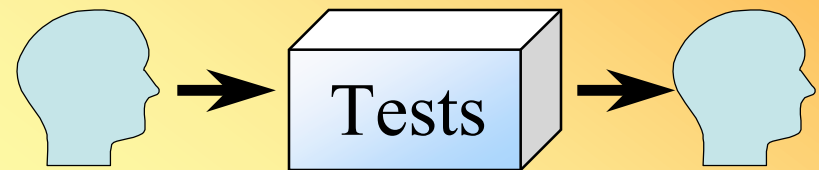
vs.

In *exploratory* testing, tests are designed and executed at the same time, and they often are not recorded.



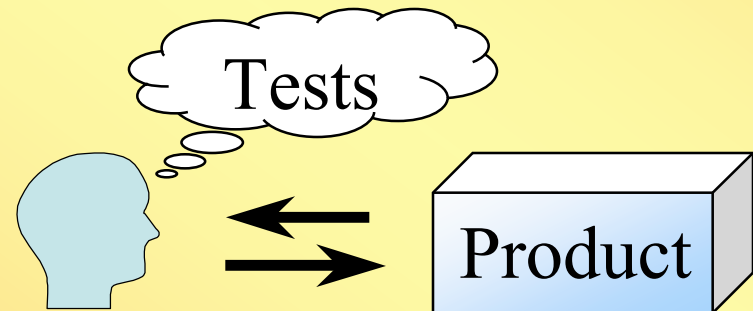
Contrasting Approaches

Scripted testing emphasizes
accountability and *decidability*.



vs.

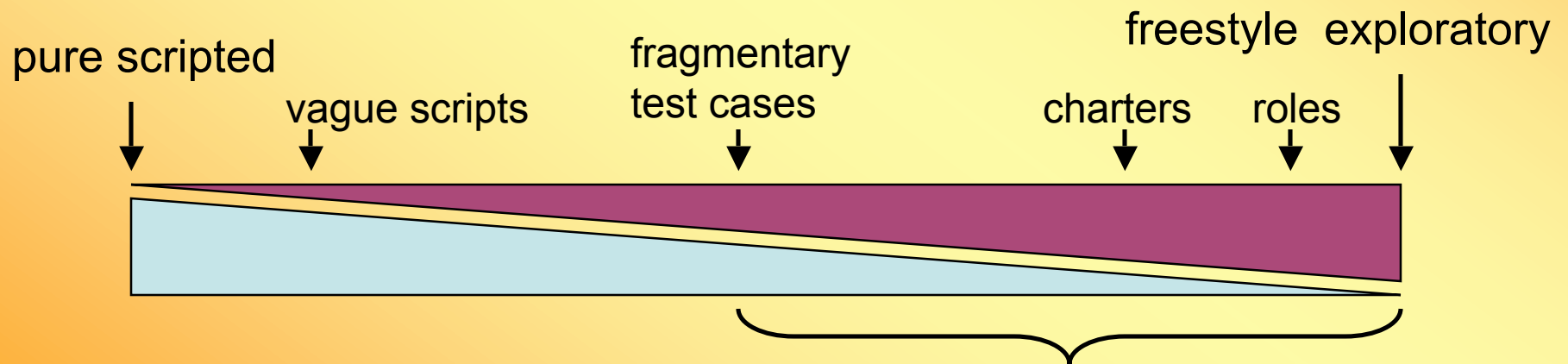
Exploratory testing emphasizes
adaptability and *learning*.



Exploratory Testing Defined

Definition


Exploratory testing is simultaneous *learning, test design, and test execution.*





When I say “exploratory testing” and don’t qualify it, I mean anything on the exploratory side of this continuum.

Exploratory Testing Tasks

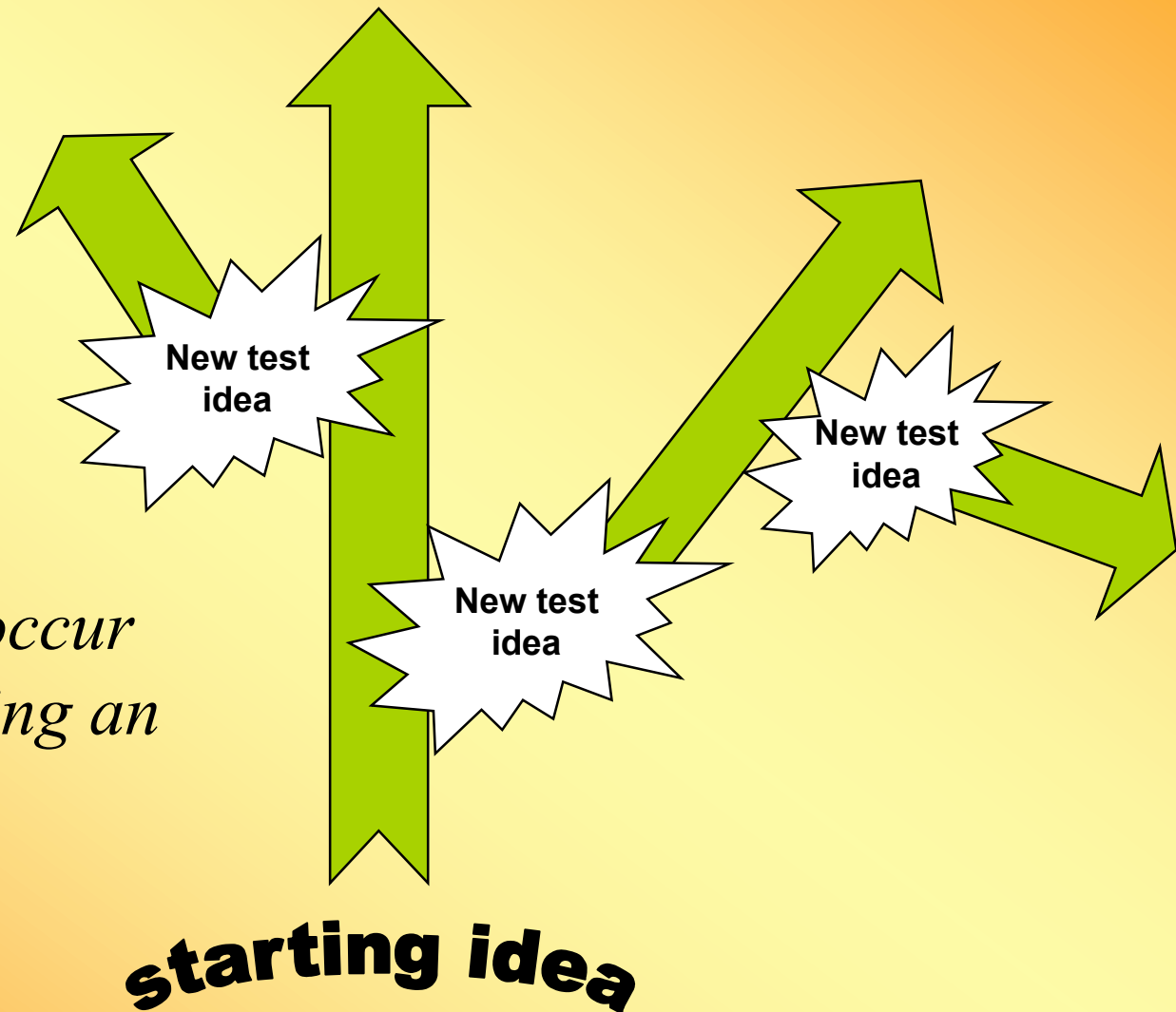
	Learning	Design Tests	Execute Tests
Product <i>(coverage)</i>	Discover the elements of the product.	Decide which elements to test.	Observe product behavior.
Quality <i>(oracles)</i>	Discover how the product should work.	Speculate about possible quality problems.	Evaluate behavior against expectations.
Techniques	Discover test design techniques that can be used.	Select & apply test design techniques.	Configure & operate the product.

**Testing notes**

**Tests**

**Problems Found**

Exploratory Forks



New test ideas occur continually during an ET session.

Exploratory Testing

- You know all the *techniques* you need to do good exploratory testing.
- The challenge is to develop your *thinking* about testing, so that you apply useful techniques at appropriate times.

By our *thinking*, we can compensate for a difficult project environment.

Instead of this... consider this.

- | | |
|--------------------------|------------------------------|
| – complete specs | – implicit specs & inference |
| – quantifiable criteria | – meaningful criteria |
| – protected schedule | – risk-driven iterations |
| – early involvement | – good working relationship |
| – zero defect philosophy | – good enough quality |
| – complete test coverage | – enough information |

Heuristic thinking

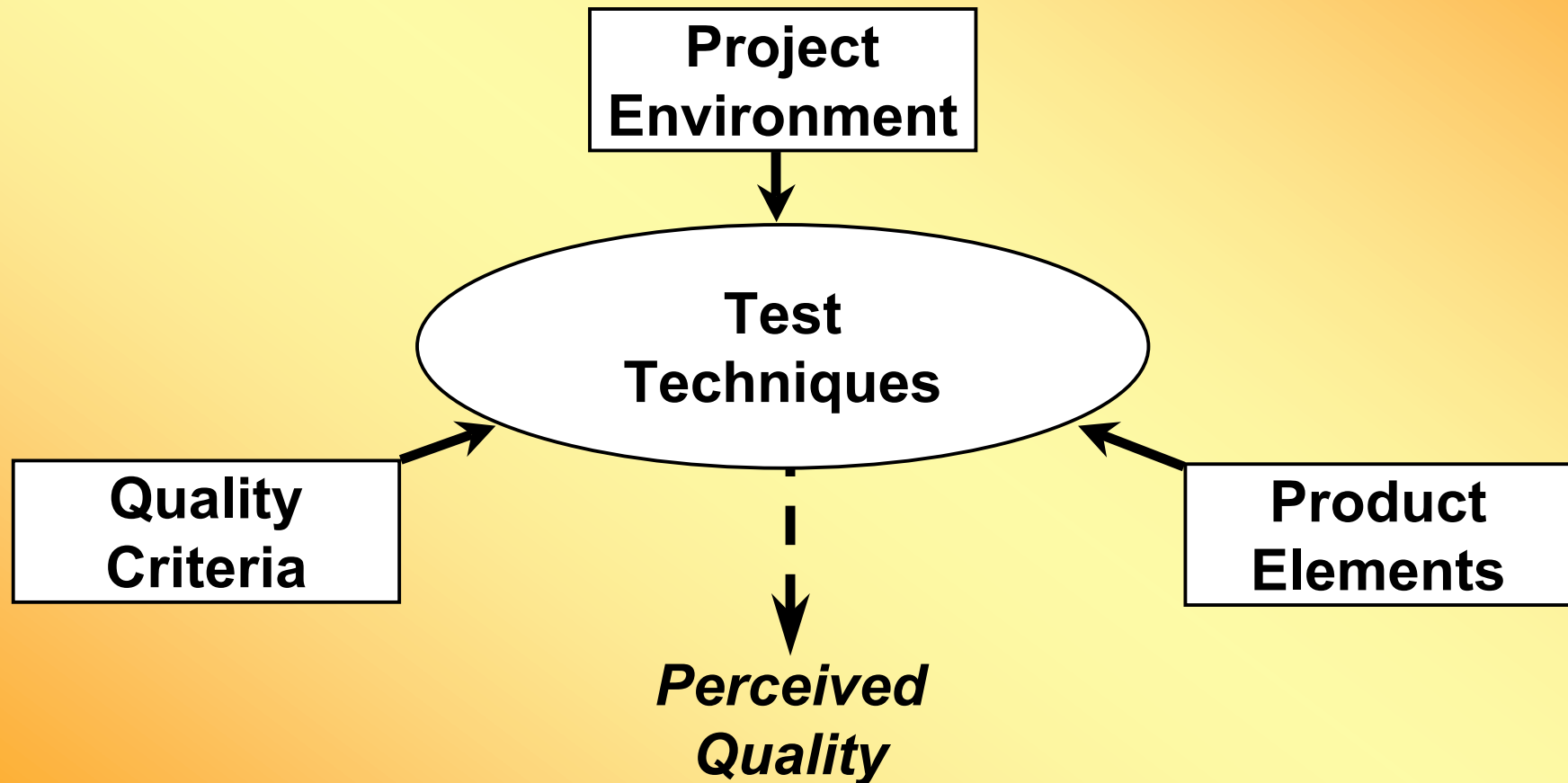
- A heuristic is a fallible idea or method that may help solve a problem.
- We've worked with several heuristics so far, including heuristics for:
 - Determining whether the program passed or failed a test (*all oracles are heuristics*).
 - Explaining why a behavior appears wrong (*the consistency heuristics*)
 - Organizing thinking about a project as a whole (*the heuristic test strategy model, for organizing risk-based testing and specification analysis, and soon, for test selection*)
 - Extending domain-testing thinking to non-ordered domains (*the printer testing example illustrates this approach*)
- The attacks are all examples of a heuristic approach to testing. They are educated guesses about what would be useful (given a fallible theory of error, here is a type of test that will probably expose the anticipated problem.)

A fundamental heuristic:

Testing is the process of asking questions of the program

- Product
 - What is this product?
 - What can I control and observe?
 - What should I test?
- Tests
 - What would constitute a diversified and practical test strategy?
 - How can I improve my understanding of how well or poorly this product works?
 - If there were an important problem here, how would I uncover it?
 - What document to load? Which button to push? What number to enter?
 - How powerful is this test?
 - What have I learned from this test that helps me perform powerful new tests?
 - What just happened? How do I examine that more closely?
- Problems
 - What quality criteria matter?
 - What kinds of problems might I find in this product?
 - Is what I see, here, a problem? If so, why?
 - How important is this problem? Why should it be fixed?

Heuristic Model for Test Design



Heuristic Model for Test Design

- We've used this model to organize failure-modes and to analyze specifications.
- We can **also** use it to guide test design overall.
 - If you're out of ideas about what to test,
 - Pick the focus of your tests by sampling your *Product's Elements* and considering which ones (or which interesting combinations) haven't yet been sufficiently tested, OR
 - Pick a theme for a set of tests by reviewing the *Quality Attributes* and considering which ones are important for the project that haven't yet been thoroughly tested. (Then pick the product elements that are interesting to test for this attribute, such as testing the product Preferences dialog (an element) for accessibility (a quality attribute).
 - If you're not sure what to test *for*
 - Review a failure mode catalog for examples of ways your product might fail.
 - When considering *how to test*,
 - Look carefully at your *Project Factors*, for suggestions of factors that might make one approach easier or less expensive or in some other way more appropriate than another.

The 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 is a tool for overcoming fear of complexity.
- Often, a testing problem isn't as hard as it looks.
- Sometimes it *is* as hard as it looks, and you need to quit for a while and consider how to tackle the problem.
- It may take several plunge & quit cycles to do the job.

Exploration Trigger Heuristic: No Questions

If you find yourself without any questions, ask yourself “Why don’t I have any questions?”

If you don’t have any *issues or concerns* about product testability or test environment, that itself may be a critical issue.

When you are uncautious, uncritical, or uncurious, that’s when you are most likely to let important problems slip by. *Don’t fall asleep!*

Observation vs. Inference

- Observation and inference are easily confused.
- Observation is direct sensory data, but on a very low level it is guided and manipulated by inferences and heuristics.
- You sense very little of what there is to sense.
- You remember little of what you actually sense.
- Some things you think you see in one instance may be confused with memories of other things you saw at other times.
- **It's easy to miss bugs that occur right in front of your eyes.**
- **It's easy to think you “saw” a thing when in fact you merely inferred that you must have seen it.**

What can you do about it?

Observation vs. Inference

Here's what:

- Accept that we're all fallible, but that we can learn to be better observers by learning from mistakes.
- Pay special attention to incidents where someone notices something you could have noticed, but did not.
- Don't strongly commit to a belief about any important evidence you've seen only once.
- Whenever you describe what you experienced, notice where you're saying what you saw and heard, and where you are instead jumping to a conclusion about "what was really going on."
- Where feasible, look at things in more than one way, and collect more than one kind of information about what happened (such as repeated testing, paired testing, loggers and log files, or video cameras).

ET Done Well is a Structured Process

- Exploratory testing, as we teach it, is a structured process conducted by a skilled tester, or by less skilled testers or users working under reasonable supervision.
- The structure of ET comes from:
 - 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 reasoned.**

ET is an Adaptive Process

- Exploratory testing decentralizes the testing problem.
- Instead of trying to solve it:
 - only before test execution begins.
 - by investing in expensive test documentation that tends to reduce the total number of tests that can be created.
 - only via a designer who is not necessarily the tester.
 - while trying to eliminate the variations among testers.
 - completely, and all at once.
- It is solved:
 - over the course of the project.
 - by minimizing the need for expensive test documentation so that more tests and more complex tests can be created with the same effort.
 - via testers who may also be test designers.
 - by taking maximum advantage of variations among testers.
 - incrementally and cyclically.

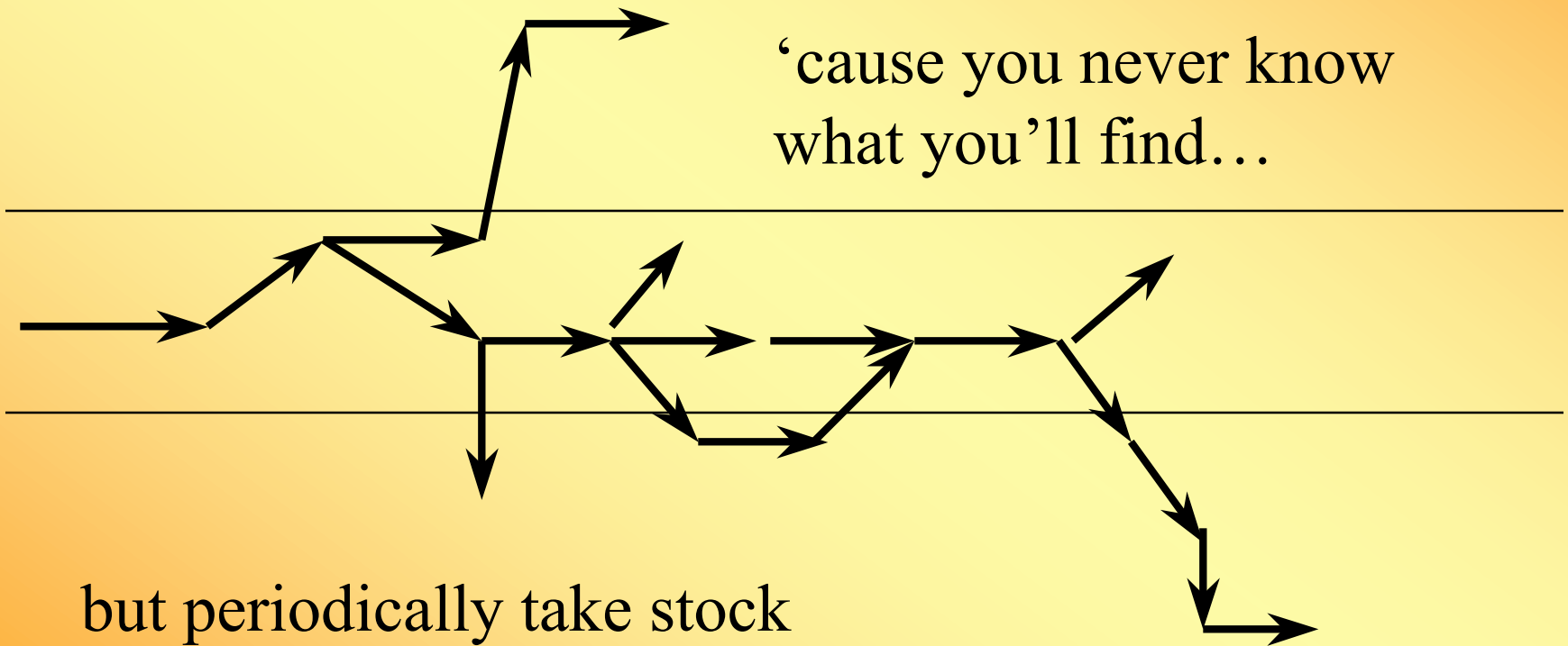
Thinking About Coverage

- Testers with less expertise...
 - Think about coverage mostly in terms of what they can see.
 - Cover the product indiscriminately.
 - Avoid questions about the completeness of their testing.
 - Can't reason about how much testing is enough.
- Better testers are more likely to...
 - Think about coverage in many dimensions.
 - Maximize diversity of tests while focusing on areas of risk.
 - Invite questions about the completeness of their testing.
 - Lead discussions on what testing is needed.

Lateral Thinking

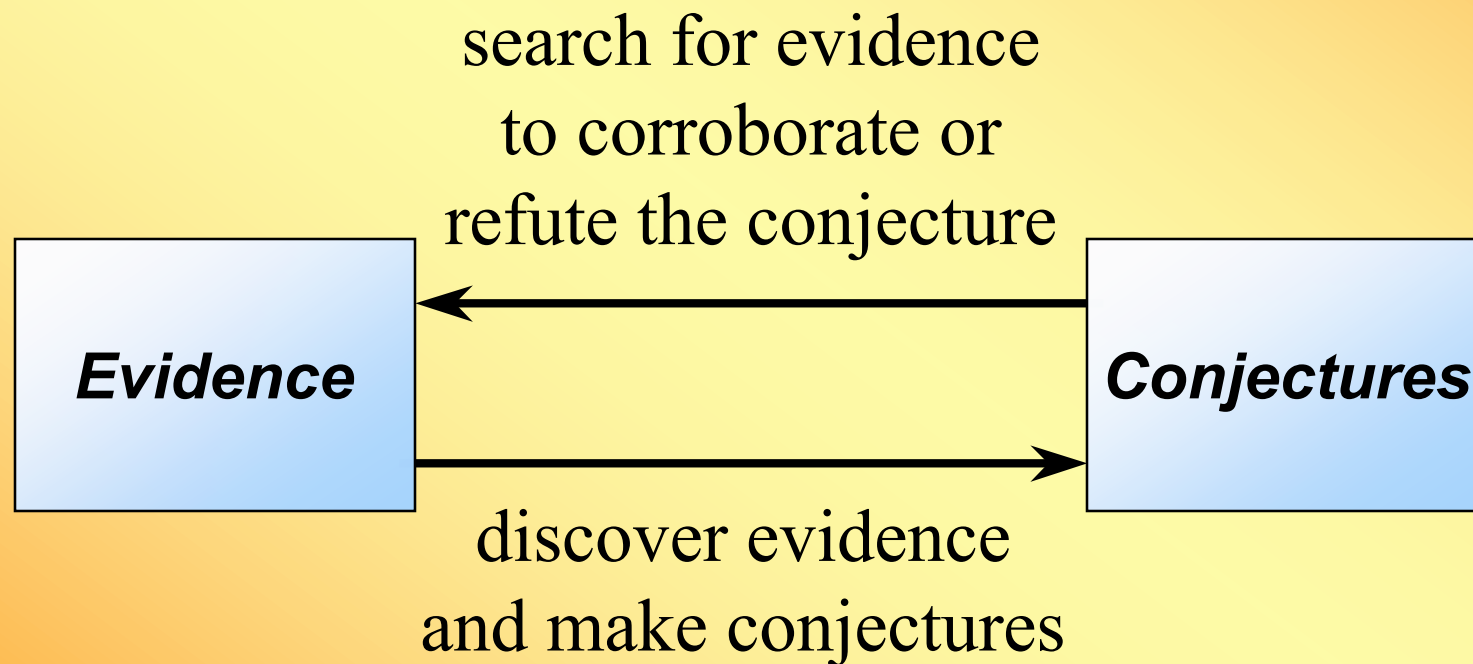
Let yourself be distracted...

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



but periodically take stock
of your status against your mission

Forward-Backward Thinking



Common Concerns About ET

- **Concerns**

- We have a long-life product and many versions, and we want a good corporate memory of key tests and techniques. Corporate memory is at risk because of the lack of documentation.
- The regulators would excommunicate us. The lawyers would massacre us. The auditors would reject us.
- We have specific tests that should be rerun regularly.

- **Replies**

- So, use a balanced approach, not purely exploratory.
- Even if you script all tests, you needn't outlaw exploratory behavior.
- Let no regulation or formalism be an excuse for bad testing.

Common Concerns About ET

- **Concerns**

- Some tests are too complex to be kept in the tester's head. The tester has to write stuff down or he will not do a thorough or deep job.

- **Replies**

- There is no inherent conflict between ET and documentation.
- There *is* often a conflict between writing *high quality* documentation and doing ET when both must be done at the *same moment*. But why do that?
- Automatic logging tools can solve part of the problem.
- Exploratory testing can be aided by any manner of test tool, document, or checklist. It can even be done from detailed test scripts.

Common Concerns About ET

- **Concerns**

- ET works well for expert testers, but we don't have any.

- **Replies**

- Detailed test procedures do not solve that problem, they merely obscure it, like perfume on a rotten egg.
- Our goal as test managers should be to develop skilled testers so that this problem disappears, over time.
- Since ET requires test design skill in some measure, ET management must constrain the testing problem to fit the level and type of test design skill possessed by the tester.
- We constrain the testing problem by personally supervising the testers, and making use of concise documentation, NOT by using detailed test scripts. *Humans make poor robots.*

Common Concerns About ET

- **Concerns**

- How do I tell the difference between bluffing and exploratory testing?
- If I send a scout and he comes back without finding anything, how do I know he didn't just go to sleep behind some tree?

- **Replies**

- You never know for sure— just as you don't know if a tester truly followed a test procedure.
- It's about reputation and relationships. Managing testers is like managing executives, not factory workers.
- Give novice testers short leashes; better testers long leashes. An expert tester may not need a leash at all.
- Work *closely* with your testers, and these problems go away.

Better Thinking

- *Conjecture and Refutation*: reasoning without certainty.
- *Abductive Inference*: finding the best explanation among alternatives.
- *Lateral Thinking*: the art of being distractible.
- *Forward-backward thinking*: connecting your observations to your imagination.
- *Heuristics*: applying helpful problem-solving short cuts.
- *De-biasing*: managing unhelpful short cuts.
- *Pairing*: two testers, one computer.
- *Study other fields*. Example: Information Theory.