

Explaining Testing to Anybody

James Bach, Satisfice, Inc.

James@satisfice.com

www.satisfice.com

"I'm confused. Can you explain...?"

"Let's {do my crazy plan}, okay?"

Uh oh...



How do you respond when someone you work with doesn't understand how testing works?

“In the testing process, test plans are made, test cases specified, tests databases and files set up, test output created, and so on. It is important to regard this testware not as a disposable product, but as one that has to be filed, managed, and handed over for reuse upon—**are you listening?**”



*head... heavy...
strength... fading...
ZZZZZZZZ...*

Basics of Explaining

- **Be quick**, then stop and check in.
- **Be humble**. Exceptions and alternatives abound.
- You can **be wrong**, as long as you're thoughtful.
- **Be real**. Own the explanation.
- **Be respectful**. Honor objections and questions.
- **Be patient**. Often, experience must come first.
- **Be relevant**. How does it relate to them? What's the bottom-line? What do you want them to do?

Basics of Explaining

- **Be prepared**

- *Quick point* (a few sentences at most.)
- *Simile & Metaphor* (relate to something familiar)
- *Vocabulary check* (are you using words the same way?)
- *5-minute whiteboard talk* (diagrams are powerful)
- *Concrete examples or demonstration* (in-house data)
- *Anticipate common questions and objections*
- *References & supporting material* (have it handy)

Explanations

- **Confusions**

- “What is testing?”
- “What’s so hard about testing?”
- “How do you know when you’re done?”
- “Why didn’t you find that bug?”

- **“Let’s {do my crazy plan},okay?”**

- “...change the product at the last minute...”
- “...specify those tests in advance [and automate them]...”

“What is Testing?”

- **Quick points (choose one):**

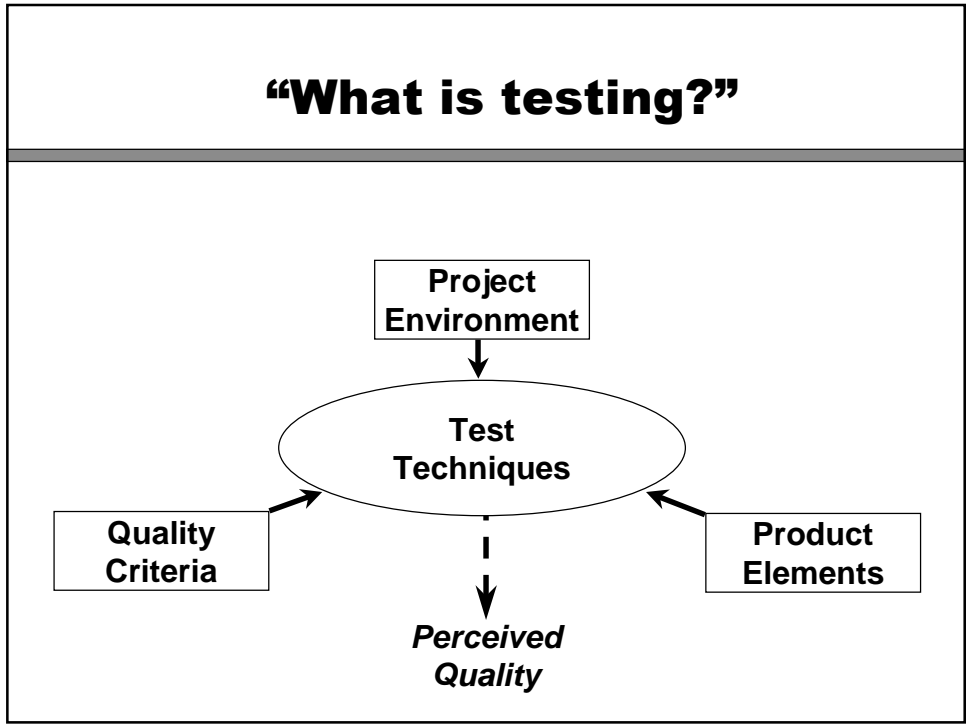
- Testing is organized skepticism. It’s the belief that things may not be as they seem; that things could be different.
- Testing is comparing the ambiguous to the invisible, so as to avoid the unthinkable happening to the unknown.
- Testing brings vital information to light, so we can see where we are and make better decisions.
- Testing is a support function that helps developers look good by finding their mistakes before anyone else does.

“What is Testing?”

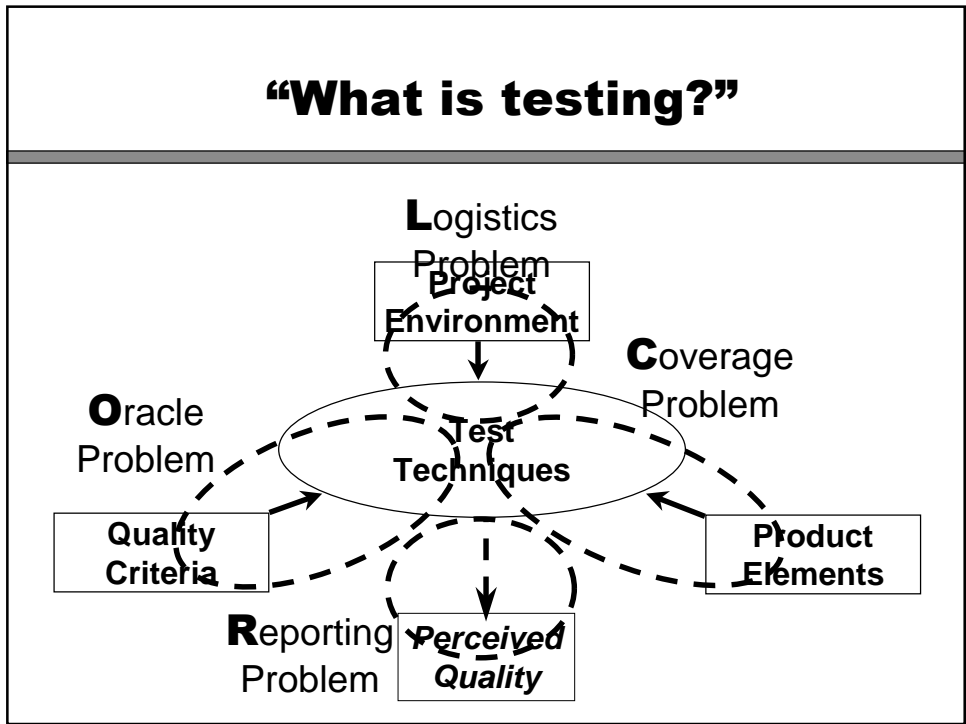
- **Similes and Metaphors**

- *Police patrol* (scanning for trouble in a complex world)
- *Secret Service* (making risky activities safer)
- *Radar* (threat detection while avoiding false alarms)
- *Copy editing* (saving authors from embarrassment)
- *Soccer/hockey defense* (we defend the goal)
- *Science* (conjecture and refutation)

“What is testing?”



“What is testing?”




“What is testing?”

▪ Supporting Material

- <http://www.kaner.com/imposs.htm>
Explains the impossibility of complete testing.
- http://www.satisfice.com/articles/test_automation_snake_oil.pdf
- <http://www.kaner.com/lawst1.htm>
Explain the challenges of regression test automation.
- http://www.satisfice.com/articles/good_enough_testing.pdf
Explains testing in the context of how much is enough.
- <http://www.rstcorp.com/marick/classic/mistakes.html>
A gold mine of marvelous little explanations.

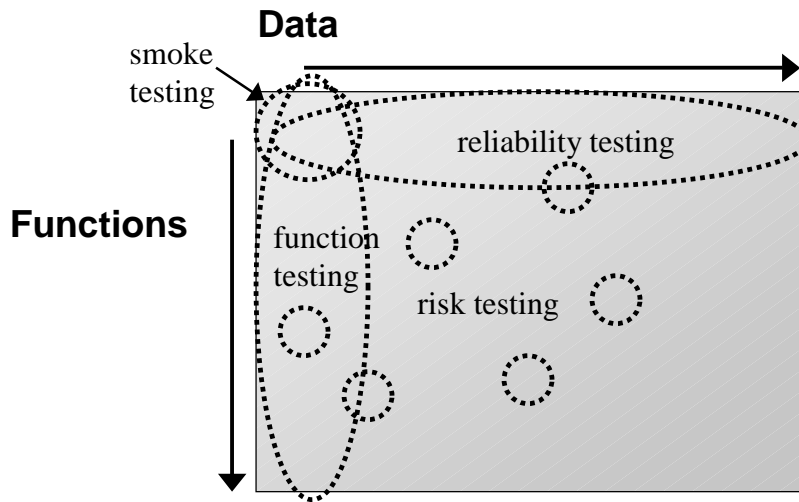
Explanation Hint



***Try highlighting one key dynamic,
instead of explaining the whole issue.***



“What’s so hard about testing?”



Explanation Hint

Try beginning by explaining things in terms of the mission, then get more concrete from there.



“How do you know when you’re done?”

1. When management has enough information to make an informed decision about the benefits and risks associated with the product.

AND

2. Management decides to release or cancel the project.

Both must be true or testing isn't done.

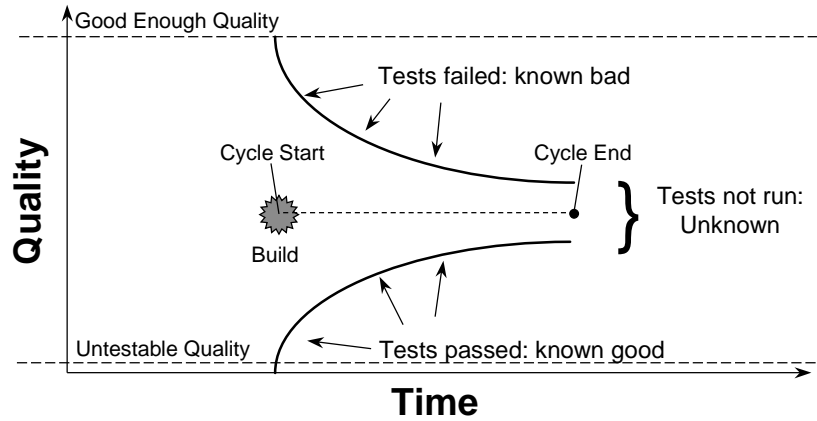
Explanation Hint

Use a diagram that helps you examine different scenarios and issues as part of the dialog.



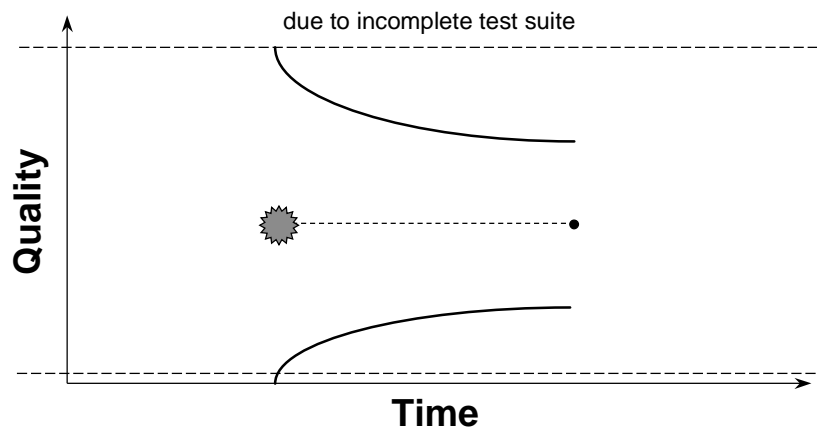
“How do you know when you’re done?”

Test Cycle Convergence



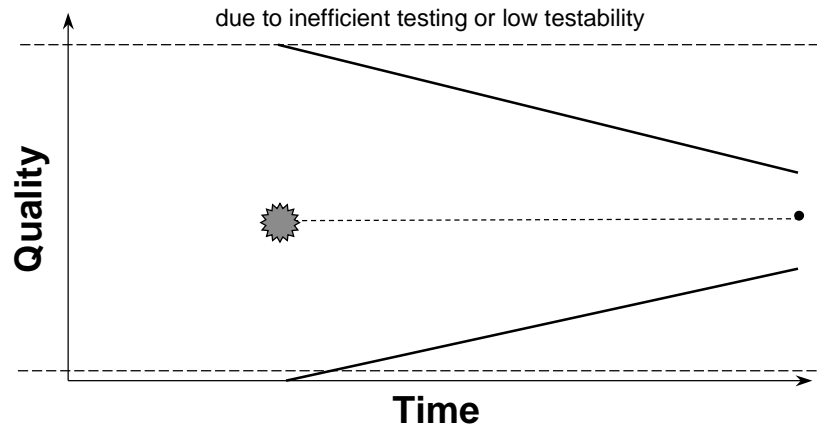
“How do you know when you’re done?”

Partial Convergence:



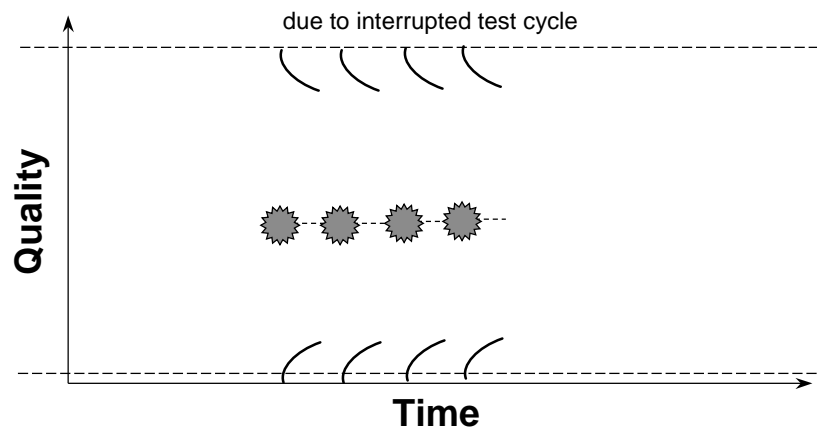
“How do you know when you’re done?”

Slow Convergence:



“How do you know when you’re done?”

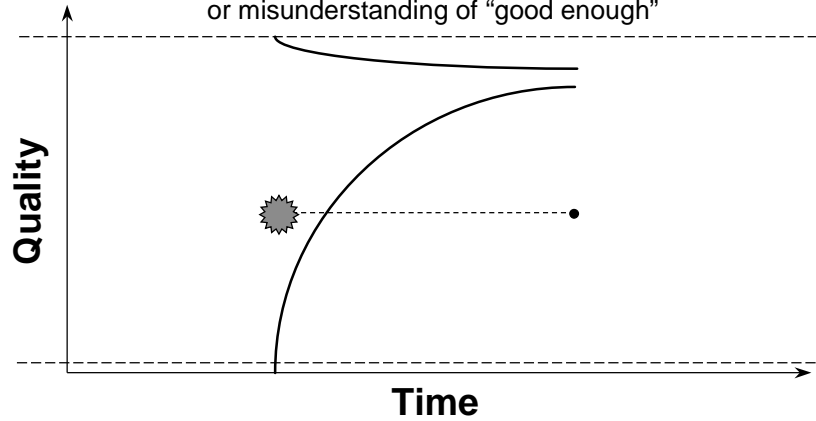
No Convergence:



“How do you know when you’re done?”

High-Biased False Convergence:

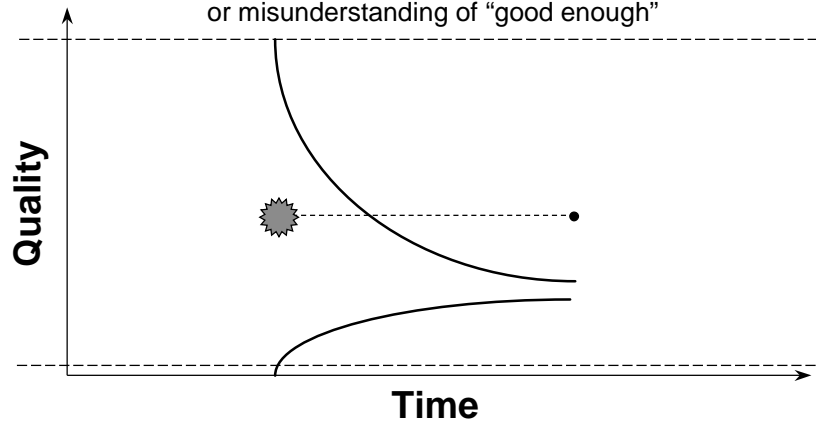
due to blissful optimism
or misunderstanding of “good enough”



“How do you know when you’re done?”

Low-Biased False Convergence:

due to negative attitudes
or misunderstanding of “good enough”




Explanation Hint



Be ready to explain things from more than one angle, in case they aren't receptive to your first try.




“Why didn't you find that bug?”

- **Highway Patrol analogy:** Do you realize how hard it would be to patrol the highways so that all speeders are always caught? 
- **Risk-based improvement argument:** Our goal is to find important problems fast. Was that bug important? Could we economically have anticipated it? If so, we'll modify our approach.
- **Testability reframe:** We didn't find it because the developers didn't make the bug obvious enough for us to notice. The developers should put better bugs into the code. *Let's make a more testable product.*

Responding to “Crazy Plans”

- **Discover the need** that motivates their plan.
- **Honor something good in it.** It’s good practice to find points in favor even of those things you don’t believe.
- **Say something like...**
 - “I’m confused about your idea. Let me explain my confusion and then you can help me see how your plan would work.”
 - “I don’t know how to do what you propose without also creating the following problems...”
 - “I’m concerned about your proposal, because I haven’t yet seen it work. What I *have* seen work is this...”
- **Then, explain the issue as you see it.**

Explanation Hint



Use analogies that come from everyday experience, when you can. Then you can appeal to common sense.



“...change the product at the last minute...”

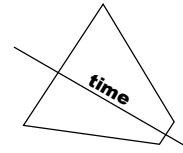
▪ Driving analogy

- When on the freeway, go 65mph.
- When on a suburban artery, go 40mph.
- When in a neighborhood, go 25mph.
- When in a parking lot, go 5mph.
- Park very slowly.

*More risks,
lower speeds.*

▪ Change Control Funnel

- The closer you are to shipping, the more reluctantly you should change code.
- To minimize retesting, carefully review proposed changes.
- “You can change what you like, but change may invalidate everything we thought we knew about quality.”

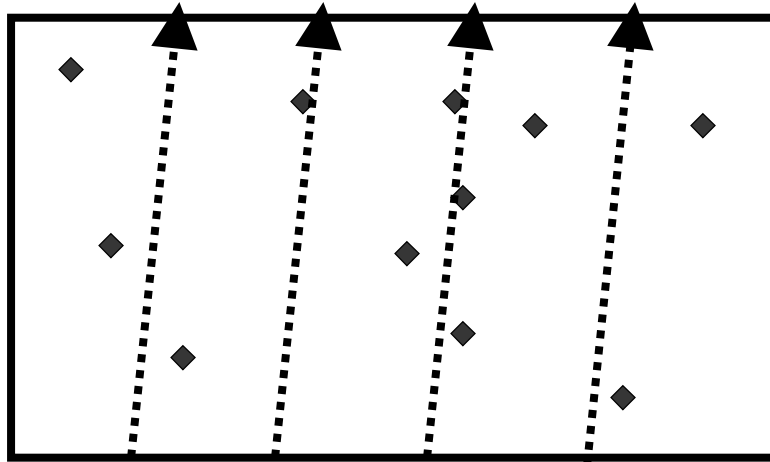


Explanation Hint

It helps to validate the crazy plan even while you're opposing it. For one thing, no plan is totally crazy. For another, it's how they know you understand it.

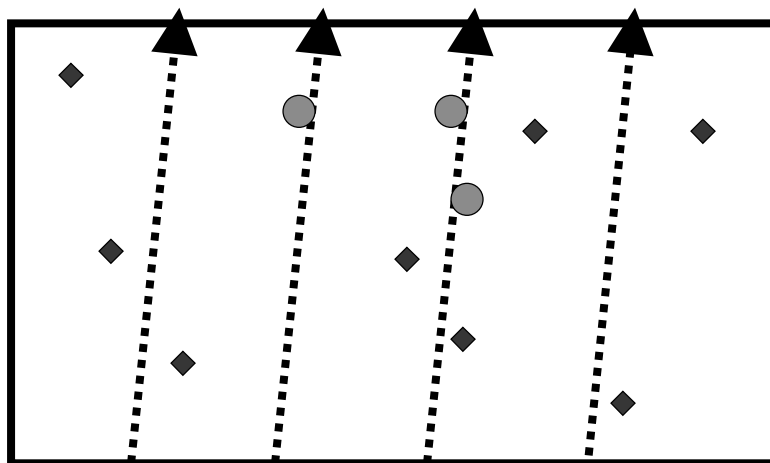


**“...specify those tests in advance
[and automate them]...”**



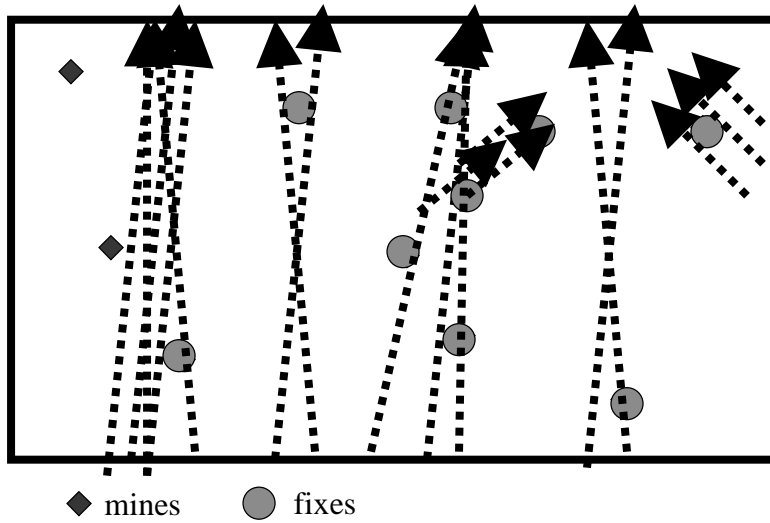
◆ mines

**“...specify those tests in advance
[and automate them]...”**

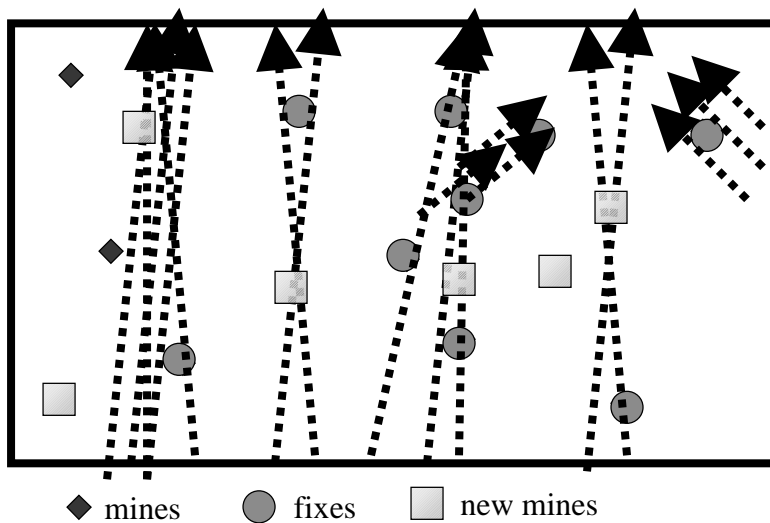


◆ mines ● fixes

**“...specify those tests in advance
[and automate them]...”**




**“...specify those tests in advance
[and automate them]...”**



“...specify those tests in advance [and automate them]...”

- **Mine clearing analogy**
 - To *avoid* mines, repeat steps exactly.
 - A repeatable test strategy can *minimize* the chance that you’ll discover new problems.
 - A variable test strategy will at least not *avoid* finding bugs.
- **Principle of diverse half-measures**
 - It’s usually better to do many different test techniques than it is to do one technique really well.
 - Maximize diversity in all dimensions your testing, if you want to find more problems and uncover more risks.
- **Kaner’s Tour Bus:** A test script is just a tour bus. Don’t fall asleep on the bus. Get off and look around.

Explanation Hint



***The success of your explanations
ultimately rests on your passion,
your curiosity, and your humanity,
not on your logic.***

(still... logic helps)

