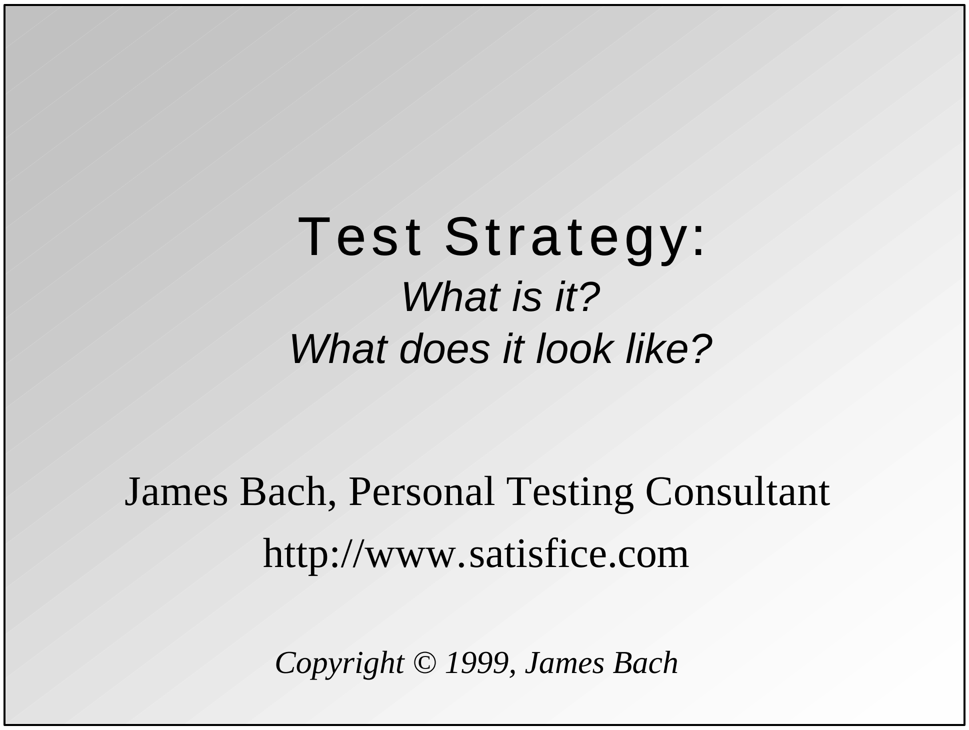
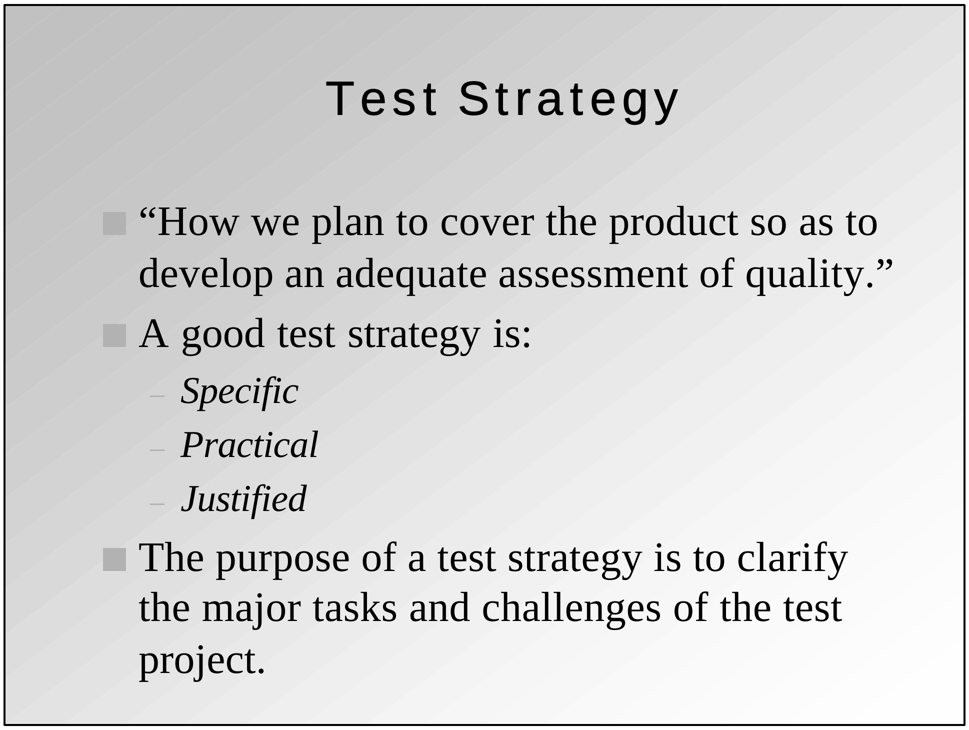


Test Strategy:

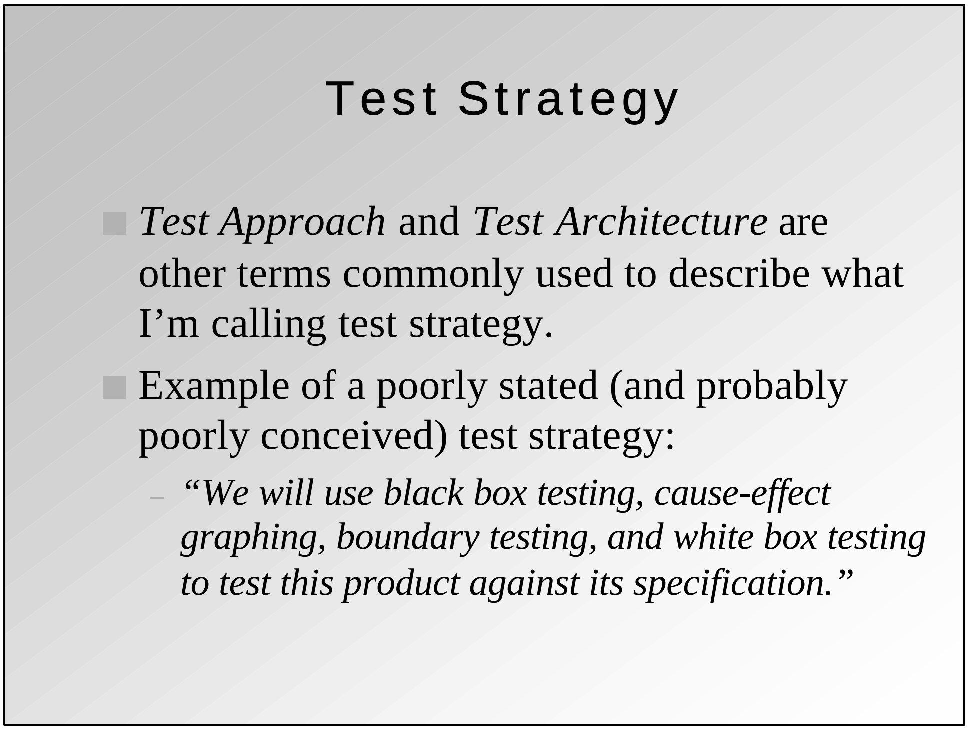
What is it?
What does it look like?

James Bach, Personal Testing Consultant
http://www.satisfice.com



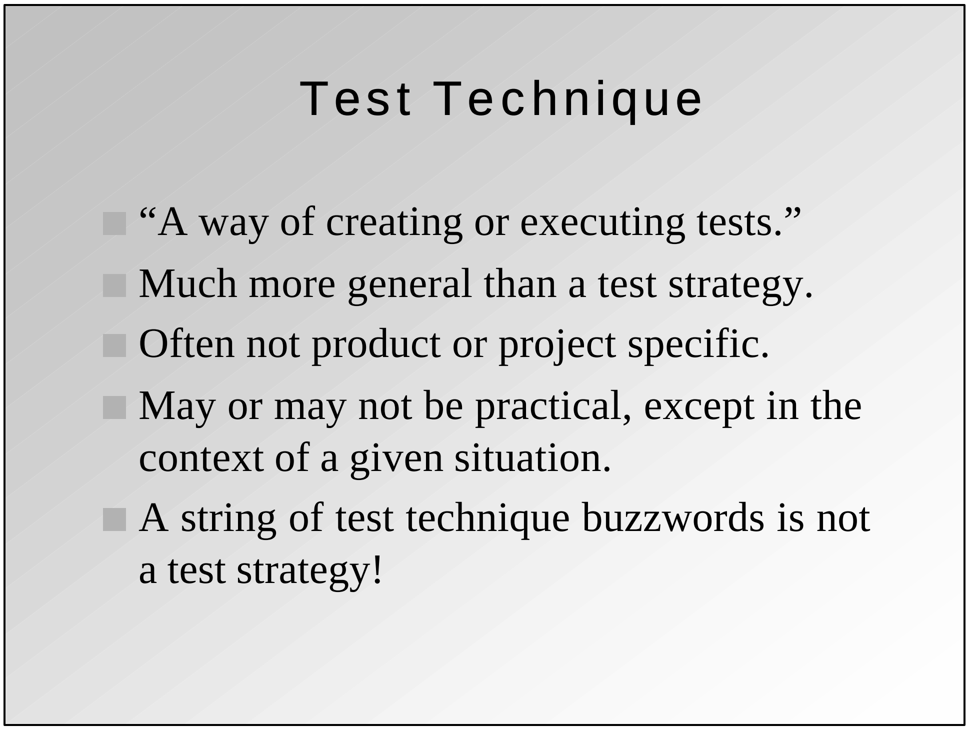
Test Strategy

- “How we plan to cover the product so as to develop an adequate assessment of quality.”
- A good test strategy is:
  - *Specific*
  - *Practical*
  - *Justified*
- The purpose of a test strategy is to clarify the major tasks and challenges of the test project.



Test Strategy

- *Test Approach* and *Test Architecture* are other terms commonly used to describe what I’m calling test strategy.
- Example of a poorly stated (and probably poorly conceived) test strategy:
  - *“We will use black box testing, cause-effect graphing, boundary testing, and white box testing to test this product against its specification.”*



Test Technique

- “A way of creating or executing tests.”
- Much more general than a test strategy.
- Often not product or project specific.
- May or may not be practical, except in the context of a given situation.
- A string of test technique buzzwords is not a test strategy!

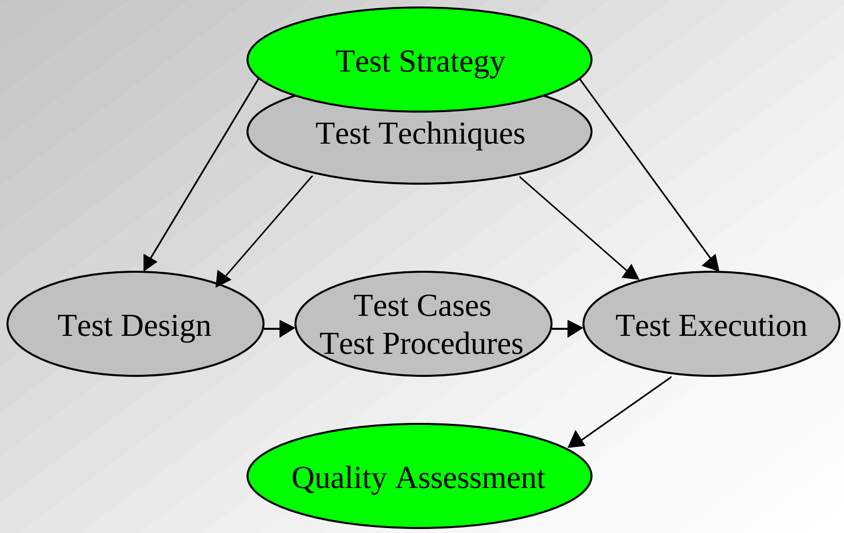
Test Cases/Procedures

- Test cases and procedures should manifest the test strategy.
- If your strategy is to “execute the test suite I got from Joe Third-Party”, how does that answer the prime strategic questions:
  - How will you *cover the product* and *assess quality*?
  - How is that *practical* and *justified* with respect to the *specifics* of this project and product?
- If you don’t know, then your real strategy is that you’re trusting things to work out.

How they go together

Test Strategy
Testing Happens….
Test
ution
Quality Assessment

How they go together



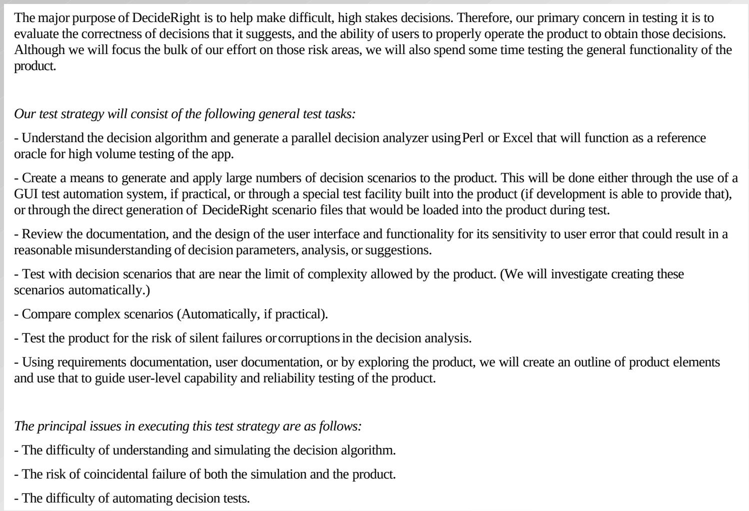
Test Strategy for DecideRight

- What is the product?
  - *An application to help people, and teams of people, make important decisions.*
- What are the key potential risks?
  - *It will suggest the wrong decisions.*
  - *People will use the product incorrectly.*
  - *It will incorrectly compare scenarios.*
  - *Scenarios may become corrupted.*
  - *It will not be able to handle complex decisions.*

Test Strategy for DecideRight

- How could we test the product so as to evaluate the *actual* risks associated with it?
  - *Understand the underlying algorithm.*
  - *Simulate the algorithm in parallel.*
  - *Capability test each major function.*
  - *Generate large numbers of decision scenarios.*
  - *Create complex scenarios and compare them.*
  - *Review documentation and help.*
  - *Test for sensitivity to user error.*

Test Strategy Text



The major purpose of DecideRight is to help make difficult, high stakes decisions. Therefore, our primary concern in testing it is to evaluate the correctness of decisions that it suggests, and the ability of users to properly operate the product to obtain those decisions. Although we will focus the bulk of our effort on those risk areas, we will also spend some time testing the general functionality of the product.

Our test strategy will consist of the following general test tasks:

- Understand the decision algorithm and generate a parallel decision analyzer using Perl or Excel that will function as a reference oracle for high volume testing of the app.

- Create a means to generate and apply large numbers of decision scenarios to the product. This will be done either through the use of a GUI test automation system, if practical, or through a special test facility built into the product (if development is able to provide that), or through the direct generation of DecideRight scenario files that would be loaded into the product during test.

- Review the documentation, and the design of the user interface and functionality for its sensitivity to user error that could result in a reasonable misunderstanding of decision parameters, analysis, or suggestions.

- Test with decision scenarios that are near the limit of complexity allowed by the product. (We will investigate creating these scenarios automatically.)

- Compare complex scenarios (Automatically, if practical).

- Test the product for the risk of silent failures or corruptions in the decision analysis.

- Using requirements documentation, user documentation, or by exploring the product, we will create an outline of product elements and use that to guide user-level capability and reliability testing of the product.

The principal issues in executing this test strategy are as follows:

- The difficulty of understanding and simulating the decision algorithm.

- The risk of coincidental failure of both the simulation and the product.

- The difficulty of automating decision tests.