Heuristic Test Strategy Model

James Bach, Satisfice, Inc.

james@satisfice.com

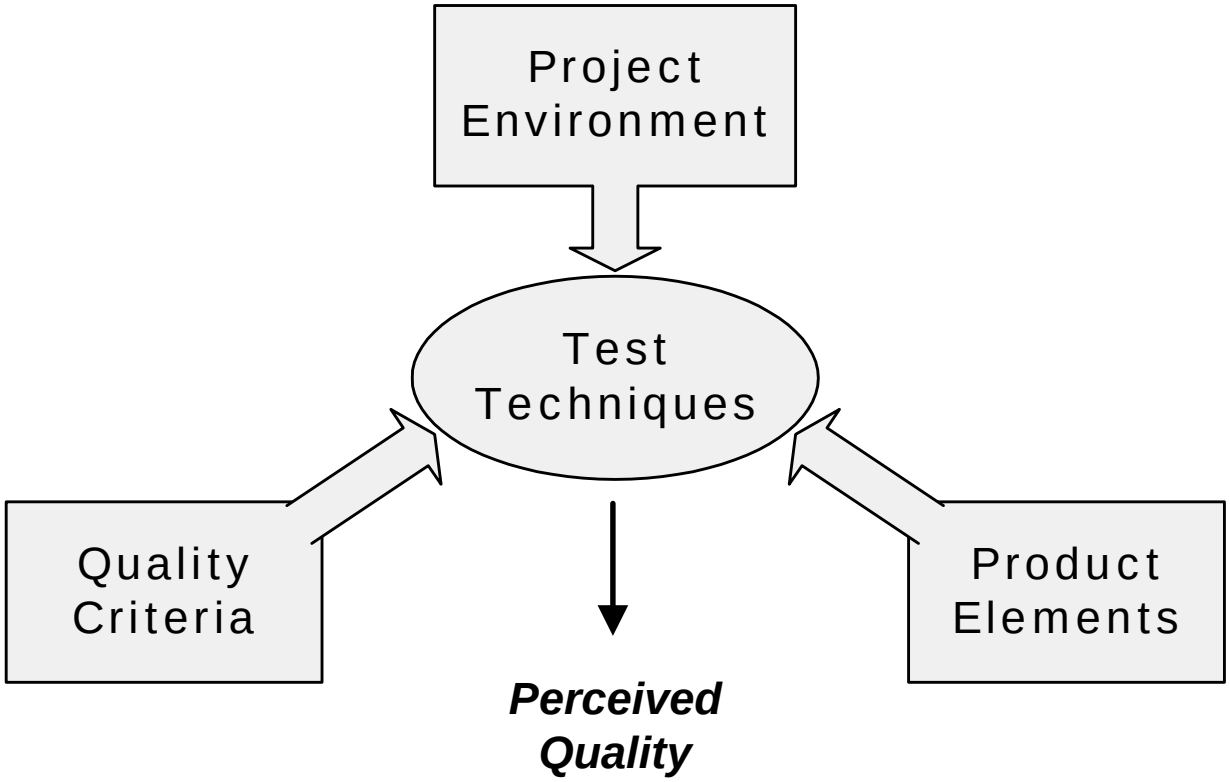
www.satisfice.com

(540) 631-0600

Designed by James Bach
james@satisfice.com
www.satisfice.com

Version 3.1
8/5/2001

Heuristic Test Strategy Model



The **Heuristic Test Strategy Model** is a set of patterns for designing a test strategy. The immediate purpose of this model is to remind testers of what to think about when they are creating tests. Ultimately, it is intended to be customized and used to facilitate dialog, self-directed learning, and more fully conscious testing among professional testers.

Project Environment includes resources, constraints, and other forces in the project that enable us to test, while also keeping us from doing a perfect job. Make sure that you make use of the resources you have available, while respecting your constraints.

Product Elements are things that you intend to test. Software is so complex and invisible that you should take special care to assure that you indeed examine all of the product that you need to examine.

Quality Criteria are the rules, values, and sources that allow you as a tester to determine if the product has problems. Quality criteria are multidimensional, and often hidden or self-contradictory.

Test Techniques are strategies for creating tests. All techniques involve some sort of analysis of project environment, product elements, and quality criteria.

Perceived Quality is the result of testing. You can never know the "actual" quality of a software product, but through the application of a variety of tests, you can derive an informed assessment of it.

General Test Techniques

A test technique is a way of creating tests. There are many interesting techniques. The list includes nine *general* techniques. Each of them is conceptually simple and often incorporated into other techniques.

Function — *Focus on functions*

1. Identify things that the product can do (functions and sub-functions).
2. Determine how you'd know if a function was capable of working.
3. Test each function.

Domain — *Focus on data*

1. Identify domains to test.
4. Analyze the limits and properties of each domain.
5. Identify combinations of domains to test.
6. Apply selection strategy:

   e.g., exhaustive, typical, boundaries, best representative

Stress — *Overwhelm the product*

1. Select test items and functions to stress.
2. Identify data and platform elements that relate to them.
3. Select or generate challenging data and platform configurations to test with:

   e.g., large or complex data structures, high loads, long test runs, many test cases, low memory conditions

Flow — *Do one thing after another*

1. Define test procedures or high level cases that incorporate multiple tests connected end-to-end.
2. Don't reset the system between tests.
3. Incorporate the element of time.
4. Combine with other techniques:

   e.g., user flow, stress flow, risk-based flow

User — *Strive for realism*

1. Identify categories of users.
2. Determine what each category of user will do, how they will do it, and what they value.
3. Get real user data, or bring real users in to test.
4. Otherwise, systematically simulate a user (be careful—it's easy to think you're like a user when you're not).

Regression — *Repeat testing after changes*

1. Identify what product elements changed.
2. Identify what elements could have been impacted by the changes.
3. Select what to test, such as recent bug fixes, past bug fixes, new code, sensitive code, or all code.

Risk — *Find big bugs first*

1. Analyze project factors, product elements, and quality criteria in order to identify sources of risk.
2. Focus testing on areas of highest potential risk.
3. Use test results to refine the risk analysis.
4. Be careful not to completely neglect low risk areas—your risk analysis might be wrong.

Claims — *Verify every claim*

1. Identify reference materials that include claims about the product (implicit or explicit).
2. Analyze individual claims, and clarify vague claims.
3. Verify each claim.
4. If you're testing from an explicit specification, expect it and the product to be brought into alignment.

Random — *Run a million different tests*

1. Look for opportunities to automatically generate a lot of tests.
2. Develop an automated, high speed evaluation mechanism.
3. Write a program to generate, execute, and evaluate the tests.

Project Environment

Creating and executing tests is the heart of the test project, and is what most people think of as "software testing." However, there are many factors in the project environment that are critical to the successful completion of a testing project. In each category, below, consider how that factor may help or hinder your testing. Be aware of what's going on in the project. Try to exploit the resources you have available, while minimizing the impact of constraints.

Customers

Anyone who is a client of the test project.

Information

Information about the product or project that is needed for testing.

- **Availability:** Do you have all the information from the developer and others that you need in order to test?
- **Volatility:** Is that information current? How are you apprised of new or change information?

Team

Anyone who will perform or support testing.

- **Size:** Do you have enough people to complete all planned testing within the desired time frame?
- **Expertise:** Do you have people with the right knowledge to satisfactorily complete the planned testing?
- **Organization:** Are the testers coordinating their efforts and pulling for the same goal?

Budget

Money needed to purchase testing resources and materials.

Equipment & Tools

Hardware, software, or documents required to administer testing.

- **Hardware:** Do we have all the equipment and platforms we need in order to execute the tests?
- **Automation:** Are any test automation tools needed? Are they available?
- **Probes:** Are any tools needed to aid in the observation of the product under test?
- **Matrices & Checklists:** Are any documents needed to track or record the progress of testing?

Processes

The tasks and events that comprise the test project.

Schedules

The sequence, duration, and synchronization of events.

- **Testing:** When will testing start and how long will it take?
- **Development:** When will builds be available for testing, features added, code frozen, etc.?
- **Documentation:** When will the user documentation be available for review?

Test Items

The product to be tested.

- **Availability:** Do you have the product to test?
- **Volatility:** Is the design and implementation of the product constantly changing?
- **Testability:** Is the product reliable enough that you can effectively test it?

Deliverables

The observable products of the test project.

- **Content:** What sort of reports will you have to make? Will you share your working notes, or just the end results?
- **Media:** How will you record and communicate your reports?

Product Elements

Ultimately a software product is an experience or solution provided to a customer. Software products have many dimensions. So, to test well, we must take care to examine a variety of those dimensions. Each category of elements, listed below, represents an important and unique aspect of a product. Testers who focus on only a few categories are likely to miss important bugs. This list is a generic starting point, but in any particular domain of technology you should be able to construct a more specific checklist of elements to examine.

Structures

Everything that comprises the physical product.

- **Code:** the code structures that comprise the product, from executables to individual routines.
- **Interfaces:** points of connection and communication between sub-systems.
- **Hardware:** any hardware component that is integral to the product.
- **Non-executable files:** any files other than multimedia or programs, like text files, sample data, or help files.
- **Ephemera and Collateral:** anything beyond software and hardware that is also part of the product, such as paper documents, web links and content, packaging, license agreements, etc..

Functions

Everything that the product does.

- **User Interface:** any functions that mediate the exchange of data with the user.
- **System Interface:** any functions that exchange data with something other than the user, such as with other programs, hard disk, network, printer, etc.
- **Application:** any function that defines or distinguishes the product or fulfills core requirements.
- **Multimedia:** Sounds, bitmaps, videos, or any graphical display embedded in the product.
- **Error Handling:** any functions that detect and recover from errors, including all error messages.
- **Interactions:** any interactions or interfaces between functions within the product.
- **Testability:** any functions provided to help test the product, such as diagnostics, log files, asserts, test menus, etc.

Data

Everything that the product processes.

- **Input:** any data that is processed by the product.
- **Output:** any data that results from processing by the product.
- **Preset:** any data that is supplied as part of the product, or otherwise built into it, such as prefabricated databases, default values, etc.
- **Persistent:** any data that is stored internally and expected to persist over multiple operations. This includes modes or states of the product, such as options settings, view modes, contents of documents, etc.
- **Temporal:** any relationship between data and time, such as the number of keystrokes per second, date stamps on files, or synchronization of distributed systems.
- **Invalid:** any data or state that should trigger an error handling function.

Platform

Everything on which the product depends.

- **External Hardware:** hardware components and configurations that are not part of the shipping product, but are required (or optional) in order for the product to work. Includes CPU's, memory, keyboards, peripheral boards, etc.
- **External Software:** software components and configurations that are not a part of the shipping product, but are required (or optional) in order for the product to work. Includes operating systems, concurrently executing applications, drivers, fonts, etc.

Operations

How the product will be used.

- **Usage Profile:** the pattern of usage, over time, including patterns of data that the product will typically process in the field. This varies by user and type of user.
- **Environment:** the physical environment in which the product will be operated, including such elements as noise, light, and distractions.

Quality Criteria Categories

When we say that a product is "high quality" we mean that it satisfies the quality criteria of its stakeholders to a "high" degree. We often have to test without knowing exactly what those criteria are, but we can use this list of criteria categories to brainstorm or otherwise ask pointed questions aimed at revealing what we need to know. For each category, determine if it is important to your project, then think how you would recognize if the product worked well or poorly in that regard. This list particularly benefits from customization. I recommend reviewing the ISO 9126 quality characteristics standard or the Encyclopedia of Software Engineering for other category ideas.

Operational Criteria

Capability ***Can it perform the required functions?***

Reliability ***Will it work well and resist failure in all required situations?***

- **Error handling:** the product resists failure in the case of errors, is graceful when it fails, and recovers readily.
- **Data Integrity:** the data in the system is protected from loss or corruption.
- **Security:** the product is protected from unauthorized use.
- **Safety:** the product will not fail in such a way as to harm life or property.

Usability ***How easy is it for a real user to use the product?***

- **Learnability:** the operation of the product can be rapidly mastered by the intended user.
- **Operability:** the product can be operated with minimum effort and fuss.

Performance ***How speedy and responsive is it?***

Installability ***How easily can it be installed onto its target platform?***

Compatibility ***How well does it work with external components & configurations?***

- **Application Compatibility:** the product works in conjunction with other software products.
- **Operating System Compatibility:** the product works with a particular operating system.
- **Hardware Compatibility:** the product works with particular hardware components and configurations.
- **Backward Compatibility:** the products works with earlier versions of itself.
- **Resource Usage:** the product doesn't unnecessarily hog memory, storage, or other system resources.

Development Criteria

Supportability ***How economical will it be to provide support to users of the product?***

Testability ***How effectively can the product be tested?***

Maintainability ***How economical is it to build, fix or enhance the product?***

Portability ***How economical will it be to port or reuse the technology elsewhere?***

Localizability ***How economical will it be to publish the product in another language?***