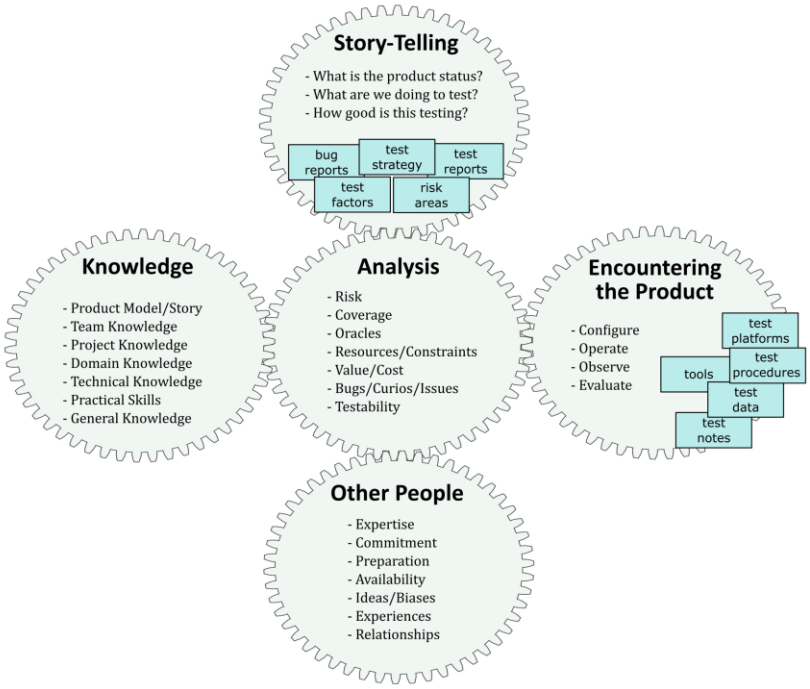
Elements of Excellent Testing

Created by James Bach, Jonathan Bach, and Michael Bolton *v4.1*

Science is testing; and *testing is science*. The world of commercial product testing differs from the world of science mainly in its object, not its subject or its process: we who test products apply ourselves to the study of an ephemeral human contrivance rather than the natural world. This is a human learning process.



Testing, like science, is an exploratory process that also makes use of scripted elements. The "gears" in the diagram to the left represent activities that evolve over time, feeding each other.

The rectangles are artifacts that result from those activities.

At the center, analysis drives the whole process. The four connection points to analysis are worth examining closer.

Learning: The connections between analysis and knowledge might be called the learning loop. In this interaction the tester is reviewing and thinking about, and applying what he knows.

Testing: The connection between analysis and experiment might be called the testing loop (in the sense of performing tests). It is dominated by questions about the status of the product. It may involve algorithmic processes such as automated output checking.

Collaboration: The connections between analysis and other people might be called the collaboration loop. Testing is always a social process to some degree, and group testing can be very energizing.

Self-management: The connection between analysis and the testing story is self-management, by which the whole process is regulated. Self-management is driven by stories we tell ourselves.

Evolving Work Products

As testing proceeds, look for any of the following to be created, refined, and possibly documented during the process.

	Test Ideas. Any idea or part of an idea, written or unwritten, that may guide the performance of a test or check.
	Output Checks. Mechanized or mechanizable processes for gathering product observations and evaluating them. A test is always human-guided, whereas a check, by definition, can be completely automated. A test often includes one or more checks, but a check cannot include a test.
	Testability Ideas. How can the product be made easier to test?
	Test Results. We may need to maintain or update test results as a baseline or historical record.
	Bug Reports. Reports regarding anything about the *product* that threatens its value.
	Issue Reports. Reports regarding anything about the *project* that threatens its value.
	Test Conditions (Product Coverage Outline). A *test condition* is anything about the product we might want to examine with a test. A *product coverage outline* is an outline, list or table of interesting test conditions.
	Product Risks. Any potential areas of bugginess or types of bug.
	Test Data. Any data available for use in testing.
	Test Tools. Any tools acquired or developed to aid testing (includes automated output checks).
	Test Strategy. The set of ideas that guide our test design.
	Test Infrastructure and Lab Procedures. General practices, protocols, controls, and systems that provide a basis for excellent testing.
	Test Estimation. Ideas about what we need and how much time we need and what obstacles might be in our way.
	Testing Story. What we know about our testing, so far.
	Product Story. What we know about the status of the product, so far.
	Test Process Assessment. Our own assessment of the quality of our test process.
	Tester and Team. The tester and the team evolves over the course of the project.
	Technical and Domain Knowledge. Our knowledge about how the product works and how it is used.

Testing Skills and Tactics

These are the skills (or tactics that involve skill) needed for professional and cost effective product testing. Each is distinctly observable and learnable, and each is necessary for excellent exploratory work.

Self-Management Skills and Tactics

	Chartering your work. Making decisions about what you will work on and how you will work. Deciding the testing story you want to manifest; knowing your client's needs, the problems you must solve, and assuring that your work is on target.
	Establishing procedures and protocols. Designing ways of working that allow you to manage your study productively. This also means becoming aware of critical patterns, habits, and behaviors that may be intuitive and bringing them under control.
	Establishing the conditions you need to succeed. Wherever feasible and to the extent feasible, establish control over the surrounding environment such that your tests and observations will not be disturbed by extraneous and uncontrolled factors.
	Self-care. Monitoring your emotional, physical, and mental states as they influence your testing; taking effective action to manage your energy and maintain a positive outlook; self-forgiveness.
	Self-criticism. Finding problems in your work and correcting them; awareness and acknowledgement of your strengths and weaknesses as a tester.
	Test status evaluation. Maintaining an awareness of problems, obstacles, limitations and biases in your testing; understanding the cost vs. value of the work; constructing the testing story.
	Ethics. Understanding your ethical code and fulfilling your responsibilities under it as you work.
	Branching your work and backtracking. Allowing yourself to be productively distracted from a course of action to explore an unanticipated new idea; identifying opportunities and pursuing them without losing track of your process.
	Focusing your work. Isolating and controlling factors to be studied; repeating experiments; limiting change; precise observation; defining and documenting procedures; optimizing effort; using focusing heuristics.
	De-focusing your work. Expanding the scope of your study; diversifying your work; changing many factors at once; broad observation; trying new procedures; using defocusing heuristics.
	Alternating activities to improve productivity. Switching among complementary activities or perspectives to create or relieve productive tension and make faster progress. See *Exploratory Testing Polarities*.
	Maintaining useful and concise records. Preserving information about your process, progress, and findings.
	Knowing when to stop. Selecting and applying stopping heuristics to determine when you have achieved good enough progress and results, or when your exploration is no longer worthwhile.
	Developing and maintaining credibility. No one will listen to you if they think what you say is not interesting or important. Remember, a tester has very little visible work, so your reputation is paramount.

Collaboration Skills and Tactics

	Getting to know people. Meeting and learning about the people around you who might be helpful, or whom you might help; developing a collegial network within your project and beyond.
	Conversation. Talking through and elaborating ideas with other people.
	Serving other testers. Performing services that support other testers on their own terms.
	Guiding other testers. Supervising testers who support your explorations; coaching testers.
	Asking for help. Articulating your needs; negotiating for assistance.
	Role visiting. Where feasible and applicable, spending time performing non-testing roles that may give you perspective or practice that makes you a better tester.
	Telling the story of your testing. Making a credible, professional report of your work to your clients in oral and written form that explains and justifies what you did.
	Telling the product story. Making a credible, relevant account of the status of the product you are studying, including bugs found. This is the ultimate goal for most test projects.

Learning Skills and Tactics

	Discovering and developing resources. Obtaining information or facilities to support your effort. Exploring those resources.
	Using the Web. Of course, there are many ways to perform research on the Internet. But, acquiring the technical information you need often begins with Google or Wikipedia.
	Considering history. Reviewing what's been done before and mining that resource for better ideas.
	Reading and analyzing documents. Reading carefully and analyzing the logic and ideas within documents that pertain to your subject.
	Interviewing. Identifying missing information, conceiving of questions, and asking questions in a way that elicits the information you seek.
	Pursuing an inquiry. A line of inquiry is a structure that organizes reading, questioning, conversation, testing, or any other information gathering tactic. It is investigation oriented around a *specific* goal. Many lines of inquiry may be served during exploration. This is, in a sense, the opposite of practicing curiosity.
	Indulging curiosity. Curiosity is investigation oriented around this *general* goal: to learn something that might be useful, at some later time. This is, in a sense, the opposite of pursuing a line of inquiry.
	Generating and elaborating a requisite variety of ideas. Working quickly in a manner good enough for the circumstances. Revisiting the solution later to extend, refine, refactor or correct it.
	Overproducing ideas for better selection. Producing many different speculative ideas and making speculative experiments, more than you can elaborate upon in the time you have. Examples are brainstorming, trial and error, genetic algorithms, free market dynamics.
	Abandoning ideas for faster progress. Letting go of some ideas in order to focus and make progress with other ones.
	Recovering or reusing ideas for better economy. Revisiting your old ideas, models, questions or conjectures; or discovering them already made by someone else.

Test Performance Skills and Tactics

	Encountering the product. Making and managing contact with the subject of your study; for technology, configuring and operating it so that it demonstrates what it can do.
	Sensemaking. Determining the meaning and significance of what you encounter; considering multiple, incompatible explanations that account for the same facts; inference to the best explanation.
	Modeling and factoring. Modeling means composing, decomposing, describing, and working with mental representations of the things you are exploring; factoring means identifying relevant dimensions, variables, and dynamics that should be tested. There are lots of formal modeling methods, as well.
	Analyzing product risk. Enabling new kinds of work or improving existing work by developing and deploying tools.
	Experiment design. As you develop ideas about what's going on, creating and performing tests designed to disconfirm those beliefs, rather than repeating the tests that merely confirm them.
	Literate observation. Making *relevant* observations guided by your various mental models; gathering different kinds of empirical data, or data about different aspects of the object; establishing procedures for rigorous observations; noticing strange things; noticing what you are *not* seeing.
	Detecting potential problems. Designing and applying oracles to detect behaviors and attributes that may be trouble.
	Assessing validity. Analyzing, monitoring, and correcting for factors that may distort or invalidate the tests.
	Notetaking. Recording observations, ideas, and progress as you test; recording useful information without unduly disturbing the test process itself.
	Data wrangling. Synthesizing, modifying, moving, and reformatting test data.
	Bug reporting and advocacy. Explaining problems in a compelling and respectful way.
	Applying tools. Enabling new kinds of work or improving existing work by developing and deploying tools.
	Testability advocacy. Analyzing and negotiating for the conditions that make testing easier and more effective.
	Protocol design. Creating and following procedures and practices that increase the reliability of the test process.
	Lab management. Creating and maintaining the systems, tools, databases, and spaces that you need to test well.

Knowledge that Helps

In addition to the skills and tactics of testing, there's lots of things we might need to know.

	Product Knowledge. What does your product do and how does it work?
	Technology Knowledge. What technology is your product built from? What other technologies can help you test it?
	Project Knowledge. What's going on in your project? What's the schedule? Who is working on it?
	Domain Knowledge. Who are the users? How do they think? What sort of process does your product support?
	General systems knowledge. This refers to the whole field of general systems theory and systems thinking. In short, it consists of the heuristics and know-how about how dynamic systems behave.
	Tool Knowledge. Physical or software-based tools that can help testing. This does not only mean tools that are called "test tools" but rather ANY tool that may help ANY aspect of the test process.
	Test Technique Knowledge. There are many kinds of testing; many specific testing heuristics you might use.
	Resource Knowledge. In addition to things that you think of as tools, you need to be aware of any resource (i.e. facility, material, or service) that is available to help you get the job done.
	People Knowledge. Who can help you? What skills do they have that you need? How do you approach them? How specifically might they contribute to the test project?
	Role Knowledge. What do the other people do who make the project work? How does their work impact yours? How might knowing more about their roles help you do your job better? How do you serve them as a tester?
	History Knowledge. What is the history of this project? This product line? The market? This company? What trouble has happened in the past that we don't want repeated?
	Business and Market Knowledge. Who are your competitors? What are those competing products? Are there similar or complementary products as well? How does quality affect your bottom line?

Helpful Skills Some Testers Have

In addition to the defining skills of testing, there are other skills and knowledge areas that testers may have.

	Coding Skill. In some companies, coding skills are a requirement. In general, the ability to build your own tools brings great power to testing. However, people with coding skills may also think too much like coders and lose empathy for users.
	Design Skill. Product design, and especially user interface design, can help sharpen your bug reports and improve your bug detection ability.
	Social Science Skills. The social sciences are about studying extremely complex, socially situated phenomena. The analytical methods and standards of social science helps testers better understand the limits of software testing and to better study and improve testing processes.
	Specification Writing Skills. Sometimes it helps for testers to help write specifications, or to suggest rewrites. Writing a spec is one great way of preparing to design tests for that product.
	Mathematics and Logic Skills. Statistics, combinatorics, and formal logic are often useful to design deep tests and characterize test coverage.
	Cognitive Science Skills. If you understand the patterns and limitations of human perception, you can better appreciate how to avoid common pitfalls of self-deception, and to design test procedures that are more reliable.

Exploratory Polarities

To develop ideas or search a complex space quickly yet thoroughly, not only must you look at the world from many points of view and perform many kinds of activities (which may be polar opposites), but your mind may get sharper from the very act of switching from one kind of activity to another. Here is a partial list of polarities:

	Warming up vs. cruising vs. cooling down
	Doing vs. describing
	Doing vs. thinking
	Deliberate vs. spontaneous
	Data gathering vs. data analysis
	Working with the product vs. reading about the product
	Working with the product vs. working with the developer
	Training (or learning) vs. performing
	Product focus vs. project focus
	Solo work vs. team effort
	Your ideas vs. other peoples' ideas
	Lab conditions vs. field conditions
	Current version vs. old versions
	Feature vs. feature
	Requirement vs. requirement
	Coverage vs. oracles
	Testing vs. touring
	Individual tests vs. general lab procedures and infrastructure
	Testing vs. resting
	Playful vs. serious

Test Strategy

This is a compressed version of the Satisfice Heuristic Test Strategy model. It's a set of considerations designed to help you test robustly or evaluate someone else's testing.

Project Environment

- ❑ *Mission.* The problems you are commissioned to solve for your customer.
- ❑ *Information.* Information about the product or project that is needed for testing.
- ❑ *Developer Relations.* How you get along with the programmers.
- ❑ *Test Team.* Anyone who will perform or support testing.
- ❑ *Equipment & Tools.* Hardware, software, or documents required to administer testing.
- ❑ *Schedules.* The sequence, duration, and synchronization of project events.
- ❑ *Test Items.* The product to be tested.
- ❑ *Deliverables.* The observable products of the test project.

Product Elements

- ❑ *Structure.* Everything that comprises the physical product.
- ❑ *Functions.* Everything that the product does.
- ❑ *Data.* Everything that the product processes.
- ❑ *Interfaces.* Every conduit by which the product is accessed or expressed.
- ❑ *Platform.* Everything on which the product depends (and that is outside your project).
- ❑ *Operations.* How the product will be used.
- ❑ *Time.* Any relationship between the product and time.

Quality Criteria Categories

- ❑ *Capability.* Can it perform the required functions?
- ❑ *Reliability.* Will it work well and resist failure in all required situations?
- ❑ *Usability.* How easy is it for a real user to use the product?
- ❑ *Charisma.* How appealing is the product?
- ❑ *Security.* How well is the product protected against unauthorized use or intrusion?
- ❑ *Scalability.* How well does the deployment of the product scale up or down?
- ❑ *Compatibility.* How well does it work with external components & configurations?
- ❑ *Performance.* How speedy and responsive is it?
- ❑ *Installability.* How easily can it be installed onto it target platform?
- ❑ *Development.* How well can we create, test, and modify it?

General Test Techniques

- ❑ *Function Testing.* Test what it can do.
- ❑ *Domain Testing.* Divide and conquer the data.
- ❑ *Stress Testing.* Overwhelm the product.
- ❑ *Flow Testing.* Do one thing after another.
- ❑ *Scenario Testing.* Test to a compelling story.
- ❑ *Claims Testing.* Verify every claim.
- ❑ *User Testing.* Involve the users.
- ❑ *Risk Testing.* Imagine a problem, then find it.
- ❑ *Automatic Checking.* Write a program to generate and run a zillion checks.