How Rapid Software Testing is Different from “Factory-Style” Testing

James Bach, Satisfice, Inc., james@satisfice.com
Michael Bolton, DevelopSense, michael@developsense.com

Rapid Software Testing is a particular mindset, which we developed over many years of consulting and teaching around the world, study of social science, and our own experiences as working test managers. This document lays out the important differences between that mindset and the popular (but in our opinion poorly conceived) view that we call the “Factory School of Testing” or “factory-style testing.”

In our analysis, factory-style testing is characterized most of all by a general de-humanization and de-skilling of the test process. Tools and documents are the trusted more than the people who maintain them. We see this embodied in such systems as ISTQB, TMap, TPI, ISO/IEC Standards, Six Sigma, TQM, CMM, and RUP, to name some popular ones.

If you look at this and think that we have mischaracterized factory-style testing, we’d like to hear from you. We want to be fair. We will correct any mischaracterizations that we become aware of. But being fair is not the same as uncritical acceptance of a marketing story.

	Factory-Style Testing	**Rapid Software Testing**
Basic Idea	Follow industry consensus (“best practices”) and principles of efficient manufacturing. Products are technological artifacts. Testing is a repetitive process of collecting facts about products. Humans are relatively unreliable, however, and skilled people are expensive and hard to find. It's therefore important to use a testing methodology that minimizes reliance on subjective factors and tester creativity. We can do this by structuring tasks in a manner reminiscent of a manufacturing plant—explicitly defining procedures and then monitoring adherence to those procedures.	Fulfill our mission for our clients by developing and applying skills and heuristics. Products are solutions that fulfill some need. Testing is an adaptive process of learning and analysis involving a great variety of experiments, observations and inferences about products. Although it is not possible to fully define or formalize that process, skilled humans are uniquely able to perform it. We therefore choose to use a methodology that maximizes freedom, self-regulation, and responsibility. We can do this by structuring training and culture in a manner reminiscent of hospitals, law firms, or elite military units—using practical drills, realistic missions, and other scalable training methods (along with expert mentoring) to build skills and create a culture of excellence. The people who do the work then structure their processes as needed.

Notion of Product Quality	A product is a system of elements and behaviors that fulfill explicitly defined requirements. The quality of a product is how well it conforms to those requirements. Quality should be measured objectively.	A product is a system of elements and behaviors, created by people, that creates a desirable experience or solution for other people. A product is always produced in some set of circumstances (we call that the context) and is delivered to some other context. Value has many aspects, some of which cannot be made explicit. Meanwhile, context may change over time. Therefore, there can no such thing as an objective or unchanging measure of quality. We can, however, discuss and construct a useful consensus about what value we think we're delivering.
Purpose of Testing	The purpose of testing is to detect non-conformances between a product and its specifications, so that they may be resolved. Specifications may exist on several levels, which leads to the concept of verification and validation. Verification means checking a component against its immediate spec, while validation means checking that it fulfills its ultimate requirements (that it is "fit for purpose").	The immediate purpose of testing is to understand the truth about the product. This in turn is done for other purposes. Usually the broader purpose is to find bugs. That means informing our clients about what they would consider to be anything about the product that threatens or unduly limits its value. In that case, a tester acts as an agent for people who have the power to decide what the product should be. Testing is sometimes done for other broad purposes, too, such as evaluating another testing process, training testers, or helping customer service prepare to support the product.
Central Questions of Testing	Does the product pass all the tests? Are there formal tests for each defined requirement?	Whom do we serve? What matters to them? Are we confident we know all the important problems in the product (regardless of defined requirements)? Without wasting our time or resources, is the testing adequate to detect every important problem that could reasonably be found (regardless of the formality of the testing)?
Unit of Work	The unit of testing work is usually a "test case," which may be a detailed set of instructions or a set of data for feeding to a formal (and perhaps automated) fact-checking mechanism. Often used interchangeably with the term "test."	There are no fixed "units of work" as such in RST. Testing is conceived as a deep intellectual process rather than an algorithmic mechanism. However, the central unit of concern in RST is the "test." A test is an experiment performed by a tester for the purposes of evaluating a product. Tests are usually embedded in a broader entity called a "test activity." Test activities may be structured in sessions (uninterrupted blocks of time), threads, or phases. Test sessions may be amenable to counting, and in Session-based Test Management they form a reasonably comparable unit of work that can be made visible to outsiders.

Method of Control	*Artifact-based and procedure-based management:* The process is intended to be manageable, ultimately, by non-testers or testers-not-present, using detailed instructions communicated explicitly via formal documents. These instructions (which may be loosely called "scripts") are followed by testers who are not expected to manage the value of their own time (but are expected to faithfully follow the instructions). Documents generally include: test plan, test case specifications (probably including test procedures as well), test results, occasionally a traceability matrix, too.	*People-based and activity-based management:* A tester managing a test process is called the "responsible tester" for that work. In RST it is important to trace who is responsible for each aspect of testing, and for that tester to be appropriately equipped and skilled in order to fulfill that role. Skilled testers manage their own processes (which may include personal supervision of supporting testers) via a negotiated mission, formal and informal heuristics, and ongoing evaluation and communication of the emerging "testing story" comprised mainly of a description of test activities. Testing proceeds in a generally exploratory fashion, even though it may be formalized ("scripted") to some degree at the discretion of the responsible tester. Also, at the tester's discretion, many different forms of documents may be used to help manage the process. Documents are as concise as possible to minimize maintenance cost and maximize testing value. Typical documents include: risk outline, product coverage outline, test activity outline. These are often manifested as mindmaps or post-it notes. If high accountability and frequent formal reporting is needed, consider using thread-based or session-based test management to package and monitor test activities. Do not use metrics for any purpose of controlling people; use metrics only for purposes of casual inquiry, so as to provoke useful conversations.
Approach to Estimating Work	Estimate required test cases based on review of specifications, if possible. Otherwise estimate by analogy to comparable projects.	Estimate work incrementally as you test or prepare to test, using the test estimation poster heuristic. Use all available information to identify necessary activities as well as any obstacles to the test process. Do not pretend to be able to predict how many bugs or builds or changes you will have to deal with—all of which may strongly impact testing. Avoid estimating if possible, but if needed, estimate test effort for an ideal (i.e. bug-free and instantly available) individual test cycle (i.e. testing needed for a single build) based on a requisite variety of test activities mapped to the full breadth of coverage areas. Express the estimation as a set of test session counts.

Approach to Test Design	Apply any of a list of named test techniques such as "equivalence class partitioning," or "boundary testing." Another approach is to itemize the parts of the specifications and write instructions that "try" or "tour" each of them. Whatever test design method is used, it should be standardized across the organization.	Test design is identical to experiment design as practiced in the sciences. All the methods and skills used in science are potentially relevant to software testing. We proceed by conjecture and refutation. Test design skill is gained through training and practice. Test techniques are heuristics that require skill, otherwise they are hollow. The basic method of test design is to model the product in a requisite variety of business-relevant ways, then determine ways to operate and interact with the product that "cover" the product with respect to those models while applying a requisite variety of business-relevant oracles to detect problems.
Ideal Sequence of Events	1. Receive correct specification. 2. Create test cases based on specification. 3. Set up the test lab and facilities. 4. (if possible) Automate test cases. 5. Receive product to test. 6. Run (or re-run) tests. 7. Report problems. 8. Receive new product with bug fixes. 9. Re-run tests and verify fixes.	There is no ideal sequence. Sequence and phasing of work is entirely contingent on the testing context. We ask ourselves "what's a good thing to do now?" Here are some general things that happen, which may happen in any order or portion, simultaneously or incrementally. • Learn about and model the product. • Analyze product risk. • Conceive of worthwhile test activities. • Test the product and report problems. • Explain and justify the testing. • Formalize the testing to improve test integrity. • Deformalize the testing to expand test coverage. • Set up the test lab and facilities. • Modify, redirect, and improve the work as conditions change. • Stop doing things that aren't helping enough. • Develop and improve relationships with the team. • Discover and experiment with new tools and methods.
Attitude Toward Change	V-Model or Waterfall. Prevent disruptive changes by proper planning in advance. Document the plan in detail, and prepare test plan and test cases in advance. Disciplined planning and communication eliminate surprises, later on.	We use an agile approach. We focus on preparation rather than planning. We focus on lowering the cost of exploratory cycles rather than deciding things up front. Whatever plans are made are going to change, so let's adapt to that change quickly.
Method of Assessing Testing	Review test artifacts. Review traceability of test cases to requirements. Capture and monitor metrics based on test case counts. Check whether documents conform to all process requirements. Possibly monitor code coverage using appropriate tools. Count bugs that escape the test process.	Discuss the testing with responsible tester. Personally observe testing (or demonstrations thereof). Observe and discuss the application of relevant heuristics. Evaluate the test strategy and test results relative to the needs of the business (we call that "test framing"). Don't count bugs that escape the test process (the count doesn't mean anything), instead investigate and learn from each one.

Definition of Done	All tests performed. Planned testing is complete. At this point the product is considered validated. The test results or report is then "signed off" by management.	All important questions about the status of the product have been answered. Clients are able to make well-informed decisions about it. Complete testing is impossible and there is no test for "always works." Instead we are obliged to stop when we conclude that further testing does not seem justified. Since we may be wrong about this, we evaluate the testing partly by the performance of the product in the field after the product is released. Since understanding of risk changes over the course of testing (testing is, in fact, an empirical form of risk analysis), we cannot rely on specific pre-specified "exit criteria" to decide when to stop. Furthermore, development activity constantly changes our baseline of understanding.
Role of Humans	Humans may play any of four roles in Factory School testing: designing methodologies, designing test procedures, automating test procedures, or following test procedures. Methodologists are rarely necessary. Instead, following perceived consensus standards and "best practices" is preferred. Test designers are not necessarily the same people who follow the test procedures that the designers create, but might be. Test design and execution are almost always two separate processes, however, regardless of whether they are done by the same people or different people. Automation is important, because tools are seen as a way to make test execution cheap and reliable.	The role of humans is central. There are three basic roles: test lead, responsible tester, and supporting tester. A responsible tester is a tester in charge of testing some part of a product, and is able to control his own methodology, procedures, tools, and activities. A test lead is a tester with three additional responsibilities: creating the conditions necessary for testing to succeed, coordinating the activities of other testers and helpers, and training testers. A supporting tester is someone who does testing activities under the supervision of a tester or lead, but is not responsible for the value of his own time. A supporting tester may be a senior person, such as an experienced developer, who is temporarily assisting the test process, or perhaps a novice tester not yet ready to take full responsibility.
Core Required Skill	The core skill is procedural discipline (in other words, the ability to follow instructions). Strongly relates to the ability to write instructions.	The core skill is ability to learn. This relates to curiosity, play, puzzle-solving, and tolerance for confusion.
Tester Diversity	Testers should be interchangeable. The test process benefits from standardization and formalization on all levels. Industry-wide certification makes it easier to find and foster appropriately qualified testers.	Each tester is unique, just as each lawyer, writer, or doctor is unique. Two testers may both be qualified to serve the same project, but we do not expect them to use the same methods or strategies, or perform the same tests in the same ways. For maximum effectiveness, a tester should work in a way that best exploits his own talents and temperament. The appropriate unit of analysis is the team, not the tester. Testing is served best by a diversified team—because that minimizes the probability of missing an important problem.
Role of Tacit Knowledge	There is no official role for tacit knowledge, although some techniques are defined as "experience-based" and job descriptions sometimes call for a certain number of years of experience, presumably because that may be correlated with higher competence of some unidentified kind.	Tacit knowledge is extremely important. The RST methodology is based on the premise that much of competence is tacit (unspoken) and is conveyed not through listening or reading to explicit instructions, but rather through observation of natural work, deliberative practical problem-solving, and live coaching by a supervisor. RST makes extensive and systematic use of heuristics that activate and direct tacit knowledge and skill.

Shifting Work to a New Tester	The role of humans should be minimized. In a well-run factory-style test process, it shouldn't matter who is doing the testing. The testing artifacts define the testing so that anyone can read them. Any new tester reads the documentation and follows the procedures. Automation should be used wherever possible to make this a moot point.	The role of humans is primary. Every tester is different. No one is interchangeable, even though all competent testers are potentially interoperable. Any skilled tester is capable of testing any product from scratch, to a reasonable degree, given reasonable time to prepare. Any unskilled tester will be working under supervision. If there are no skilled testers, then good testing will be impossible no matter what methodology you try. In any situation where testing is or should be formalized, records of some kind are typically produced. Concise notes, tables, or other artifacts—up to and including extremely detailed and rigorous test procedure documentation—may be created. A tester may use such material to take over testing from another tester. However, the receiving tester must be able to take full responsibility for the contents of what he inherits. Any mysterious document or tool must be discarded or recreated. Mysterious instructions are a potential hazard to the project. Testers may also pass work to each other through paired work, or through talking or live demonstration.
Role of Tools	Tools should be used to store and track testing documents and artifacts, as well as to automate test execution as much as possible. Tool use should be standardized across the organization.	In RST, we say testing cannot be automated, because any testing-like activity done exclusively by an algorithmic process is called “checking.” We do this for the same reason that programmers call automated programming “compiling.” It is important to distinguish between the capability and responsibility of humans vs. that of machines. However, testing may be supported and expanded by the use of tools. Testers and test teams are strongly encouraged to innovate and experiment with tools. Testers should develop or acquire any tools that might help make their testing more powerful or reliable, as long as these don't cost too much or create an unhelpful bias in test coverage. While it is not required or even desirable for every tester to be a programmer, a high functioning test team will have the ability to put tools in place quickly and inexpensively as the needs arise.