

Two Futures of Software Testing - OSQA February 24, 2004

Michael Bolton, DevelopSense mb@developsense.com (416) 656-5160 <http://www.developsense.com>

Prediction is very difficult, especially when it's about the future.

—Niels Bohr, or Woody Allen, or Yogi Berra¹

The Four Schools of Software Testing

Bret Pettichord has identified four schools of software testing:

- The Analytical or Mathematical School, which suggests that using the right formulas and flowcharts and state diagrams will save us.
- The Factory or Process School, which says that lots of planning, scripting, and other paperwork will save us.
- The Quality Police School, which says that testers are the gatekeepers, and that telling other people how to do their jobs (even when we don't do ours that way) will save us.
- The Context-Driven School, which says that our skills, our ability to think critically, and our ability to choose practices appropriate to serving the testing mission, will save us.

Bret Pettichord, The Four Schools of Software Testing. Presentation at the Workshop on Teaching Software Testing, Melbourne, FL, 2003. This was an important attempt to understand testing cultures and their principles.

http://www.testingeducation.org/conference/wtst_pettichord_FSofST2.pdf

Bret is also a very cogent thinker and writer on the subject of test automation. <http://www.pettichord.com>

It is currently unclear which of the Four Schools will prevail in the long run. I see at least two possible futures.

The Dark Future

Testing should be a rigorously planned and controlled process. We will prove that the systems work by strict verification testing. In order to ensure the quality of the testing process, we will test only according to documented requirements. **Tests must be scripted**, with each step specified in advance based on extensive study of functional specifications and use cases. Test scripts will be made simple enough that they can be understood by any person, and that person won't need any skills in testing, programming, or the business domain. Cost will be reduced, since **testing won't require skilled labour**.

To facilitate this, each program will be completely specified, and will be allocated two years in development. During development, **change will be politely refused**. Even though users or our business sponsor may want or need a particular feature, we'll recognize that **nothing is more important than following a process** to get the job done right. By sticking to this regimen, **we can eradicate risk** from the software development process.

Every test that we develop will be automated. **We do not leave the detection of defects to fallible humans**. We will spend large amounts of money on automated testing tools that capture and replay the work of senior testers. We will re-run all of those tests at the end of every release cycle. If the automation can't keep up with the pace of program development, **we'll just have to slow down development to let test automation catch up**. The data from the automated tests will not only assure product quality, but they'll also give confidence to senior management, since managers will know exactly how many test cases have been run. **Management can count on its testers to assure quality**.

Ad hoc or exploratory testing will be forbidden. In a well-defined testing process, **exploration is simply a luxury that we cannot afford**.

In order to ensure the objectivity of the test team, all testing will be done independently of development. Testers will work on the other side of a Chinese wall, and will be insulated thereby from the pernicious influence of the developers. The disengagement of development and testing will guarantee objectivity and be further reason for management to have confidence in the process. **Testers will be the gatekeepers**, bringing both quality and assurance into quality assurance.

This is the Dark Future. It's strongly rooted in the Factory and Quality Police Schools. The Dark Future's intellectual underpinnings (such as they are) are founded on several fantasies: that projects and requirements should not respond to the need for change, that people are interchangeable, that testing skill is not important, that we can eradicate risk. The worst thing about viewing the Dark Future from today's perspective is that **it looks so much like today**.

In the Dark Future, management will continue to misunderstand the role and the value of testing, at least in part because the testers themselves will do little to demonstrate their value. Testers will be granted the illusion of being quality police or gatekeepers, but will be overruled as soon as business realities trump the fantasy. In the Dark Future, testing remains a sinkhole in the career path, something for someone to do while they wait to become a developer, a marketer, or a pensioner. In the Dark Future, testing is not an intellectual process.

The Brighter Future

The Brighter Future puts critical thinking and tester skill at the centre of testing (not "the testing process"; testing is far more of a perspective than a process. At its core, testing is the ability to imagine and to pursue questions that begin with "What if...?" **Testing serves the project and the project community** by investigating the product and by credibly reporting on how it works.

In the brighter future, **testers do not work from scripts**. Just as machines aren't good at being people, people aren't good at being machines either. We do not merely report on deviations from prescriptive, outdated, and incomplete documents; good testers realize **that the map is not the territory, and the requirements documents are not the requirements**. (Note that in six years of consulting, I have never met anyone who said, "We are proud of our requirement documents and our functional specifications and would change nothing about them.") **Instead of building testing on such shaky foundations, testers consciously and conscientiously apply heuristics**—fallible rules of thumb—and are encouraged to investigate and explore the product rapidly and skillfully. We use heuristics to determine inconsistencies between the software and the project

¹ Attribution is very difficult, too.

community's ultimate vision of it. **Testing is based on systems thinking**, understanding the roles of input, control, and feedback.

Management recognizes testing's value because **testing provides service in the form of valuable information, *right now***, about the state of the program. Management knows that testers can answer questions about the program clear, rapidly, and credibly. Testers are expert communicators and acknowledged masters at providing valuable feedback at any moment.

In the Bright Future, we know the extent and limit of quantitative measurements. We realize that the **numbers don't mean anything without context**. A one-dimensional count is not a metric; the essence of a metric is to measure something in terms of something else. We recognize that data is multivariate, and **we continuously look for alternative explanations for numbers that appear to tell a simple story**. Among other qualitative measurements, we know the value of observing and gauging people's feelings, and how they can reflect the true state of the project better than any bug count.

In the Brighter Future, testing at some level can and should be done by everyone in the project community—we testers are specialists in the perspective, but **just as it's important for testers to know something about programming, it's important for programmers to know about testing**. We'll readily exchange skills—but our own skills must be entirely credible.

In the Bright Future, **change is anticipated and embraced**. Testers recognize the profoundly important reality that **we are not gatekeepers**, and that the business doesn't need our permission to make changes to its projects. Rather than preventing change, **testing assists us in making changes more confidently**. This is not to say that we accept everything that is asked of us. In particular, we gracefully decline from making decisions about when the product is to be shipped; that's management's prerogative, and we do not compromise management's position by usurping that role; nor would we accept it if it's offered.

Testers in the Bright Future know how to use automation effectively. Computers are brilliant at doing things quickly, but they don't do brilliant things. In particular, **computers don't think, observe, conjecture, suspect, anticipate, evaluate, or judge**. They don't feel, and they certainly don't empathize. They're very good at doing repetitive things very quickly, but repetition for its own sake is not usually valuable. No computer ever detected a single defect on its own, but computers can help us find certain kinds of defects quickly. So **we don't use automation to replace the testing effort, but to assist the testing effort**. In the Brighter Future, testers will have the training and the experience to recognize little, stupid things that computers can and should do, and that people can't and shouldn't do

Testers and developers collaborate on ways to **make programs more testable, providing visibility and controllability**. Testers apply at least modest skill with tools like quickie scripts and regular expressions. Each testing shop has ready collections of small useful tools. **Think Swiss Army Knives and Meccano sets**, rather than machine shops full of overwhelmingly large, powerful, and expensive tools. Some testers are expert programmers; all are expert in choosing and using tools appropriate to the task.

The brighter future will be brighter because, in James Bach's words, "As testers, we light the way." **Testers know, from moment to moment, how the product *really* works.**

Learning

To learn about **finding bugs**, read

- [Lessons Learned in Software Testing](#) by Cem Kaner, James Bach, and Bret Pettichord
- [Testing Computer Software](#) by Cem Kaner, Jack Falk, and Hung Quoc Nguyen
- [How to Break Software](#) and by Whittaker, and [How to Break Software Security](#) by Whittaker and Thompson
- [Testing Applications on the Web](#) by Hung Nguyen
- [Hacking Web Applications Exposed](#) by Joel Scambray and Mike Shema

To learn about **testing philosophy**, read

- [The Pleasure of Finding Things Out](#) by Richard Feynman. Read his Appendix to the Challenger Report.
- [Surely You're Joking, Dr. Feynman! Adventures of a Curious Character](#) by Richard Feynman
- [What Do You Care What Other People Think?](#) by Richard Feynman
- [Quality Software Management Vols. 1 – 4](#) by Jerry Weinberg
- [Anything](#) by Jerry Weinberg

To learn **other wonderful stuff that I believe is worth thinking about**, look at

- [Please Understand Me](#) by David Kiersey ♦ The Myers-Briggs Type Inventory, which provides insight into your own preferences and why *other people* seem to think so strangely.
- [The Visual Display of Quantitative Information](#), Edward Tufte ♦ How to present information in persuasive, compelling, and beautiful ways.
- [A Pattern Language](#), Christopher Alexander ♦ A book about architecture, even more interesting as a book about thinking and creating similar but unique things—like computer programs and tests for them.
- [Better Software](#), a most unfortunate name of a most wonderful magazine.
- [The Amplifying Your Effectiveness Conference](#), held every November in Phoenix, AZ. See <http://www.ayeconference.com> for details.

To find stuff **on the Web** about testing and other topics, see

- StickyMinds ♦ <http://www.StickyMinds.com>
- Risks Digest ♦ <http://catless.ncl.ac.uk/risks>
- Cem Kaner ♦ <http://www.kaner.com>
- James Bach ♦ <http://www.satisfice.com>
- Michael Bolton ♦ <http://www.developsense.com>
- The Florida Institute of Technology ♦ <http://www.testingeducation.org>

DevelopSense provides testing, training, and consulting services in plain English. info@developsense.com