

Four Schools of Software Testing

Bret Pettichord



bret@pettichord.com

www.pettichord.com

Workshop on Teaching Software Testing, Florida Tech, February 2003

Why Classify Testing Doctrines into Schools?

- ◆ Clarify why testing experts disagree
 - Not simply a matter of personality or experience
 - Many testers are unaware of the underlying reasons for disagreement
- ◆ Improve the basis for debate
- ◆ Explain how my school (context-driven testing) differs from the others

What is a School?

- ◆ A school is not a technique
- ◆ A school is not a paradigm
- ◆ A school is defined by
 - Standards of criticism
 - Exemplar techniques
 - Hierarchies of values
- ◆ Example: each school defines risk-based testing differently, based on its own values.
- ◆ Most teaching belongs to one of four schools

Analytical School: Core Beliefs

- ◆ Software is a logical artifact
- ◆ Testing is a branch of CS/Mathematics
 - Objective, rigorous, comprehensive
- ◆ Testing techniques must have a logico-mathematical form
 - “one right answer”
- ◆ Testing is technical
- ◆ Key Question:
Which techniques should we use?

Analytical School: Exemplar

Code Coverage

- aka “Structural” testing
- Dozens of code-coverage metrics have been designed and compared
- Provides an objective “measure” of testing

Analytical School

◆ Implications

- Precise and detailed specifications are a prerequisite for testing
- Testers verify that the software behavior conforms to its specification

◆ Most prevalent

- Academia
- High-reliability industry (e.g. Telecom)

◆ Authors

- Boris Beizer, Paul Jorgensen, Robert V. Binder, John Musa

Factory School: Core Beliefs

- ◆ Software development is a project
- ◆ Testing is a measure of progress
- ◆ Testing must be managed
 - Predictable, repeatable, planned
- ◆ Testing must be cost-effective
 - Low-skilled workers require direction
- ◆ Testing is following rules
- ◆ Key Question:
What metrics should we use?

Factory School: Exemplar

Requirements Traceability

- Make sure that every requirement has been tested

Factory School

◆ Implications

- Requires clear boundaries between testing and other activities (start/stop criteria)
- Resists changing plans (complicates progress tracking)
- Taylorism
- Accept management assumptions about testing
- Encourages industry testing standards, "best practices," and certification

◆ Most Prevalent

- IT projects
- Government projects

◆ Authors

- Rex Black, Dorothy Graham

Quality Assurance School: Core Beliefs

- ◆ Software quality requires discipline
- ◆ Testing determines whether development processes are being followed.
- ◆ Testers may need to police developers to follow the rules
- ◆ Testers have to protect users from bad software
- ◆ Key Question:
Are we following a good process?

Quality Assurance Exemplar

The Gatekeeper

- The software isn't ready until QA says it's ready

Quality Assurance

◆ Implications

- Prefer “Quality Assurance” over “Testing”
- Testing is a stepping stone to “process improvement”
- May alienate developers

◆ Most Prevalent

- Large bureaucracies
- Organizations under stress

◆ Authors

- Alka Jarvis

Context-Driven School: Core Beliefs

- ◆ Software is created by people. People set the context.
- ◆ Testing finds bugs. A bug is anything that could bug a stakeholder.
- ◆ Testing provides information to the project
- ◆ Testing is a skilled, mental activity
- ◆ Testing is multidisciplinary
- ◆ Key Question:
What tests would be most valuable right now?

Context-Driven School: Exemplar

Exploratory Testing

- Concurrent test design and test execution
- Rapid learning

Context-Driven School

◆ Implications

- Expect changes. Adapt testing plans based on test results
- Unchallenged assumptions are dangerous.
- Pragmatism
- Effectiveness of test strategies can only be determined with field research
- Focus on skill over practice

◆ Most Prominent

- Commercial, Market-driven Software

◆ Authors

- Cem Kaner, Brian Marick, James Bach

Four Views of Risk-Based Testing

◆ Analytical

- Use operational profiles
- Calculate reliability

◆ Factory

- Focus on management perception of risk
- Pseudo-math often used

◆ Quality Assurance

- Uncover *project* risks
- Prove that project is out of control

◆ Context-Driven

- Testing develops team understanding of risks
- Develop testers' ability to design tests for identified risks

Why I Like My School

- ◆ Consider the “triangle” problem. There is no correct answer!
- ◆ You can choose an approach based on personal values
 - Intellectual propriety
 - “Industry standards”
 - Placating management
- ◆ Or you can try something and then see how well it worked

Context-driven has testers learning as they test

Controversy #1

Usability Testing

FOR

- ◆ Context-Driven School
 - Definitely do it.
 - Usability bugs are bugs.
- ◆ Factory School
 - Do it if requested by management.
- ◆ Quality Assurance
 - Reluctant.
 - Hard to prove non-compliance.

AGAINST

- ◆ Analytical School
 - Not a form of testing.
 - Outside the testing skill set.
 - Let someone else do it.

Controversy #2

Testing Without Specs

FOR

- ◆ Context-Driven School
 - Do what you can to be useful
 - Ask questions if necessary
 - Dig up “hidden” specs

AGAINST

- ◆ Analytical School
 - Impossible
- ◆ Factory School
 - Some kind of spec is necessary
- ◆ Quality Assurance
 - Force developers to follow the process

Controversy #3

Tester Certification

FOR

- ◆ Factory School
 - Make testers easier to hire, train and manage
- ◆ Quality Assurance
 - Increase status

◆ Analytical School

- Little preference either way

AGAINST

- ◆ Context-Driven School
 - Existing certifications are based on practice, not skill