Approaches to Software Testing: An Introduction

Customized and presented to:


Scott Barber
Chief Technologist
PerfTestPlus, Inc.
Who am I?

- Chief Technologist of PerfTestPlus
- Executive Director of the Association of Software Testing
- Co-Founder of the Workshop on Performance and Reliability
- An international speaker and contributer to various publications

- My specialties include
  - Testing and analyzing performance for complex systems
  - Developing customized testing methodologies
  - Embedded systems testing
  - Testing biometric identification and security systems

- I am a member of
  - IEEE
  - American MENSA
  - the Context-Driven School of Software Testing
- and a signatory to the Manifesto for Agile Software Development
Who am I?

My name is Scott and I'm a...

Test-o-holic.
Welcome & Overview

Logistics/Facilities?

Who are each of you?

- Name
- Business Card Title
- What you actually do
- What you hope to take away from this seminar
- What is distracting you
Welcome & Overview

What are we going to be talking about?
- How and where testing fits in
- The purpose of testing
- The value of testing

How will this help me do my job better?
- Who is responsible for what & when
- How to work together to improve the quality of the product
- Avoiding “turf-wars”
- Improving the fit between process and purpose
Seminar Objectives

At the conclusion of this seminar, attendees will be familiar with and have a basic understanding of the following items, the contexts where they are generally practiced, their strengths and weaknesses in context and where to go for more information:

- Popular system test life-cycles and their relationship to popular software development life-cycles
- Industry common system testing approaches
- Industry common system testing practices
Agenda

Self-Categorization
- Testing “School”
- Testing Life Cycle
- Testing Techniques/Practices

Break
Approaches/Schools
Life Cycles
Break
Techniques/Practices
Putting it all Together
Wrap-up
Self-Categorization

On the walls of the room are several flip-chart sheets with descriptions of certain common classifications related to software testing.

I have provided each of you with “voting stickers” that you will use to indicate how well your project fits into the described category.

On each chart is a horizontal scale. You can place one sticker per sheet anywhere along the scale indicating if your project fits each description:

- Extremely well
- Somewhat well
- Not at all well

If you are completely uncertain where your project fits, place your sticker in the box labeled
Ready... Set...

Go!
The Four “Schools” of Software Testing

- Analytical/Mathematical
- Factory/Process
- Quality Control
- Context-Driven
Software Testing Life Cycles

- Waterfall/Big Bang
- Iterative/Spiral
- Test First/TDD
- Agile
Software Testing Techniques/Practices

- Scripted Manual/Automated
- Black Box/UAT
- Exploratory
- Unit/Developer/White Box
Questions
10 Min Break
Testing “Schools”

What are they?

Why make the distinction?

How do they relate to your project?

*Note* The slides in this segment heavily reference the original Four Schools of Software Testing as coined and presented by Brett Pettichord which can be found at www.pettichord.com.
Four Views of Software Testing

**Analytic School** - sees testing as rigorous and technical with many proponents in academia

**Factory School** - sees testing as a way to measure progress with emphasis on cost and repeatable standards

**Quality School** - emphasizes process, policing developers and acting as the gatekeeper

**Context-Driven School** - emphasizes people, setting out to find the bugs that will be most important to stakeholders
Why Classify Testing into Schools?

Understand why testing experts disagree
- Not simply a matter of personality or experience
- There are often underlying reasons for disagreement

Improve the basis for debate
- Differences in values may explain why we favor different policies
- Explain how my school differs from the others

But it can lead to oversimplification
What is a School?

A school is not a technique.

A school is defined by:
- Standards of criticism
- Exemplar techniques
- Hierarchies of values
Analytic School: Core Beliefs

Software is a logical artifact

Testing is a branch of CS/Mathematics (i.e. Objective, rigorous and comprehensive)

Testing techniques must have a logico-mathematical form (i.e. “one right answer”)

Testing is technical

Key Question: Which techniques should we use?
Analytic School Exemplar

Code Coverage

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

Implications
- Require precise and detailed specifications
- Testers verify whether the software conforms to its specification
- Anything else isn’t testing

Most prevalent
- Academia
- Telecom
- Safety-Critical
Factory School: Core Beliefs

Testing must be managed (i.e. Predictable, repeatable, planned)

Testing must be cost-effective (i.e. Low-skilled workers require direction)

Testing validates the product

Testing measures development progress

Key Questions:
- How can we measure whether we’re making progress?
- When will we be done?
Traceability Matrix

- Make sure that every requirement has been tested
Factory School

Implications
- Require clear boundaries between testing and other activities (start/stop criteria)
- Resist changing plans (complicates progress tracking)
- Software testing assembly line (V-model)
- Accept management assumptions about testing
- Encourage standards, “best practices,” and certification

Most prevalent
- Enterprise IT
- Government
Quality 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 School Exemplar

The Gatekeeper

- The software isn’t ready until QA says it’s ready
Quality School

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

Most prevalent
- Large bureaucracies
- Organizations under stress
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 testing would be most valuable right now?
Exploratory Testing

- Concurrent test design and test execution
- Rapid learning
Context-Driven School

Implications
- Expect changes. Adapt testing plans based on test results.
- Effectiveness of test strategies can only be determined with field research
- Testing research requires empirical and psychological study
- Focus on skill over practice

Most prevalent
- Commercial, Market-driven Software
What is Testing?

Analytic School says
- A branch of computer science and mathematics

Factory School says
- A managed process

Quality School says
- A branch of software quality assurance

Context-Driven School says
- A branch of development
What school is right for you?

Analytic?
Factory?
Quality?
Context-Driven?
Questions
Software Life Cycles

**Waterfall** – Sequential, minimal feedback loops

**V-Model** – Parallel, decoupled, minimal feedback loops

**Iterative** – Similar to lots of little waterfalls

**Spiral** – Like iterative, but presumes iterations get smaller

**TDD/TFD** – Test first, developers do much testing

**Agile/XP** – Adaptive, incremental, just-in-time, just enough
Iterative

Core Process Workflows
- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Testing
- Deployment
- Operations & Support

Core Supporting Workflows
- Configuration & Change Mgmt
- Project Management
- Environment
- Infrastructure Management

Phases
- Inception
- Elaboration
- Construction
- Transition
- Production

Iterations
- Preliminary Iteration(s)
- Iter.#1
- Iter.#2
- Iter.#tk
- Iter.#tk
- Iter.#tk
- Iter.#tk
- Iter.#tk
Test Driven/Test First Development

- **SOURCE CODE** (written in C or Assembly)
- **TEST CODE** (written in C or Assembly)
- **COMPILER/ASSEMBLER**
- **LINKER**
- **SIMULATOR**
- **EMULATOR**
- **MONITOR**
- **DEPLOY**

Diagram:

1. Add a test
2. Run the tests
   - [Pass]
   - [Fail]
3. Make a little change
4. Run the tests
   - [Development continues]
   - [Fail]
5. [Development stops]

Copyright 2003 Scott W. Ambler
Life Cycle Summary

Which Life Cycle does your project use?

What are the pros/cons of that Life Cycle Model?

Which Testing Schools seem to fit with which Life Cycles?

How about Hybrids?

Are there others?
10 Min Break
Testing Techniques/Practices

**Scripted** – Follow the well-defined predetermined steps

**Regression** – Find changes from previous release

**Ad Hoc** – One time test to answer a specific question

**Automated** – Computer conducts tests and reports results

**Exploratory** – Simultaneous learning, test design & test execution to detect defects of interest to a stakeholder

**Unit/Developer/White Box** – Tests not using the UI

**Black Box/UAT** – Expected usage and error modes

**Shotgun** – “If you use a shotgun, you don't have to aim that carefully”
Testing Techniques/Practices

What testing techniques have you seen work...
- Particularly well
- Particularly poorly
- Well in specific circumstances

What other techniques/practices...
- Do you know of
- Would you recommend
- Would you like to know more about or try
Questions
Examples of what works well together

Waterfall + Unit Tests + Scripted + UAT = Factory

V-Model + Unit Tests + Scripted = Analytical

Iterative + Regression + Scripted + UAT = Quality

TDD/TFP + Unit Tests + Exploratory = Context-Driven
Examples of what doesn't work together

- Waterfall + Regression
- Agile + Scripted
- Analytical + Exploratory
- Context-Driven + Anything done without adding value
Moving Forward

Compared to where you placed your stickers earlier, where would you like those stickers to be in the future and why?
Questions