Test Management and Techniques

These slides are distributed under the Creative Commons License.

In brief summary, you may make and distribute copies of these slides so long as you give the original author credit and, if you alter, transform or build upon this work, you distribute the resulting work only under a license identical to this one.

For the rest of the details of the license, see http://creativecommons.org/licenses/by-sa/2.0/legalcode.
Management and Techniques Outline

1. Introduction
2. Test Management and Techniques
   - Testing Fundamentals in 1 hour!
   - Black Box and Glass Box testing
   - Test Planning and Management
   - Test Documentation
   - Test Execution and Techniques
   - Heuristic Risk-Based Testing
   - Progress Tracking and Metrics
3. ET Planning, Exec. and Documentation
4. ET Styles
5. ET Management
2. Test Management and Techniques

2.1 Testing Fundamentals in 1 hour!

2.2 Test Execution and Techniques

2.3 Heuristic Risk-Based Testing

2.4 Test Management and Metrics
Testing Fundamentals in 1 hour!

- Black-Box and Glass Box testing
- Test Strategy
- Test Planning
- Test Documentation and Structure

This is what you will learn about if you take a “Testing Fundamentals” class - anywhere else...
Black Box Testing

- The tester pretend NOT to know how the program works
  - Feed "interesting" input (expected do lead to "interesting" output
  - Observe "interesting" outcome

General information - not ET Specific!
Glass Box / White Box Testing

The programmer will test based on access to and understanding of source code

- Focused testing
  - Test the program in pieces
- Testing Coverage
  - Lines of code, branches, paths etc.
- Control Flow
  - State transitions
- Data Integrity
  - Data manipulation
- Internal boundaries
- Algorithm-specific testing
  - Numeric algorithms

General information - not ET Specific!

Kaner et. al. 1999, Beizer 1990, Beizer 1995
Test Strategy

“How we plan to cover the product so as to develop an adequate assessment of quality.”

A good test strategy is:
- Product-Specific
- Risk-focused
- Diversified
- Practical

From Rapid Software Testing, copyright © 1996-2002 James Bach
©2002 Amland Consulting
Test Plan

According to IEEE Std 829-1983:

- A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will be doing each task, and any risks requiring contingency planning.

Kaner et al 1999:

- A test plan is a valuable tool to the extent that it helps you manage your testing project and find bugs. Beyond that, it is a diversion of resources.
Test Plan IEEE Std 829-1998

IEEE Standard for Software Documentation

- Test Plan Identifier
- Introduction
- Test Items
- Features to be tested
- Features not to be tested
- Approach
- Item pass / fail criteria
- Suspension criteria and resumption criteria
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

General information - not ET Specific!

ET Workshop v. 1.20 - Management and Techniques

©2002 Amland Consulting
Test Documentation (IEEE 829 – 1998)

- Test Plan (see previous slide)
- Test Design Specification
  - Test approach and features to be tested
- Test Case Specifications
  - Input, Expected Output and Environmental needs
- Test Procedure Specification
  - Steps for executing a test case
- Test Item Transmittal Report
  - List items for testing
- Test Log
  - Chronological record of the execution of tests
- Test Incident Report
  - Document events that requires investigation
- Test Summary Report
  - To summarize the results and provide evaluation
General information - not ET Specific!

- Test plan
- Test item tree
- Test item spec.
- Test data
- Test design specific.
- Test Procedure Spec.
- Test log
- Incident Report (CR)
- Test summary report
- Error / change log
- Application based groups
- Function based groups

STATIC TEST PLANNING

DYNAMIC TEST PLANNING

TEST EXECUTION

LOG KEEPING

Objectives, policy and strategy for the test project

Test Item Transmittal Report
2. Test Management and Techniques

2.1 Testing Fundamentals in 1 hour!
2.2 Test Execution and Techniques
2.3 Heuristic Risk-Based Testing
2.4 Test Management and Metrics
Test Execution and Techniques

- The Problem of testing
- Test Models, Coverage and Evaluation
- Overview Test Techniques
- Boundaries and Equivalence Classes
- Bug Reporting
- Test Automation
The Universal Test Procedure

“Try it and see if it works.”

- Learn about it
- Model it
- Speculate about it
- Configure it
- Operate it

- Know what to look for
- See what’s there

- Understand the requirements
- Identify problems
- Distinguish bad problems from not-so-bad problems

Models Coverage Evaluation

From Rapid Software Testing, copyright © 1996-2002 James Bach
©2002 Amland Consulting
All Product Testing is Something Like This (The Satisfice Testing Model)

- Project Environment
- Test Techniques
- Quality Criteria
- Product Elements

Perceived Quality

From Rapid Software Testing, copyright © 1996-2002 James Bach
The Four Problems of Testing

Logistics

Project Environment

Coverage Problem

Test Techniques

Evaluation Problem

Quality Criteria

Reporting Problem

Perceived Quality

Product Elements

From Rapid Software Testing, copyright © 1996-2002 James Bach
Models

A Model is...
- A map of a territory
- A simplified perspective
- A relationship of ideas
- An incomplete representation of reality
- A diagram, list, outline, matrix...

No good test design has ever been done without models.

The trick is to become aware of how you model the product, and learn different ways of modeling.
Coverage

Product coverage is the proportion of the product that has been tested.

There are as many kinds of coverage as there are ways to model the product.
- Structural
- Functional
- Data
- Platform
- Operations
An evaluation strategy is how you know the product works.

“...it works.”

really means

“...it appeared to meet some requirement to some degree.”

Which requirement? Appeared how?
To what degree? Perfectly? Just barely?
Evaluation: “HICCUPP”

- Consistent with History: Present function behavior is consistent with past behavior.
- Consistent with our Image: Function behavior is consistent with an image that the organization wants to project.
- Consistent with Comparable Products: Function behavior is consistent with that of similar functions in comparable products.
- Consistent with Claims: Function behavior is consistent with what people say it’s supposed to be.
- Consistent with User’s Expectations: Function behavior is consistent with what we think users want.
- Consistent within Product: Function behavior is consistent with behavior of comparable functions or functional patterns within the product.
- Consistent with Purpose: Function behavior is consistent with apparent purpose.
A test technique is a recipe for performing these tasks in order to reveal something worth reporting:

- Analyze the situation
- Model the test space
- Select what to cover
- Determine test oracles
- Configure the test system
- Observe the test system
- Evaluate the test result
General Test Techniques (1)

- Functional testing
- Domain testing
- Stress testing
- Flow testing
- User testing
- Risk testing
- Claims testing
- Random testing
- Regression testing

For any one of these techniques there’s somebody, somewhere, who believes that is the only way to test.
General Test Techniques (2)

- Flow graphs and Path Testing
- Transaction-Flow Testing
- Data-Flow Testing
- Domain Testing
  - Boundaries and Equivalence Classes
- Syntax Testing
- Logic-Based Testing
- States, State Graphs, and Transition Testing
Five-fold Testing System

Five dimensions of testing:

- Testers
  - Who does the testing.
- Coverage
  - What get tested.
- Potential Problems
  - Why you’re testing (what risk you’re testing for)
- Activities
  - How you test. For example: exploratory testing
- Evaluation
  - How to tell whether the test passed or failed.

Kaner et al, 2001b
People based techniques: Who

- User Testing
- Alpha testing
- Beta testing
- Bug bashes
- Subject-matter expert testing
- Paired testing
- Eat your own dog food

Kaner et al, 2001b
Coverage-based techniques: What

- Function testing
- Feature or function integration testing
- Menu tour
- Domain testing
- Equivalence class analysis
- Boundary testing
- Best representative testing
- Input field test catalogs or matrices
- Map and test all the ways to edit a field

- Logic testing
- State-based testing
- Path testing
- State and branch coverage
- Configuration coverage
- Specification-based testing
- Requirements-based testing
- Combination testing

Kaner et al, 2001b
Problem-based techniques: Why

Risk-Based Testing
- Risk-Based Test Management (Amland 1999)
- Doing risk analysis for the purpose of finding errors (Bach 1999a)

Constraint violations
- Input constraints
- Output constraints
- Computation constraints
- Storage (or data) constraints
Activity Based Techniques: How

- Regression testing
- Scripted testing
- Smoke testing
- Exploratory testing
- Guerilla testing
- Scenario testing
- Use Case Flow tests
- Installation testing
- Load testing
- Long Sequence testing
- Performance testing

Kaner et al, 2001b
Evaluation-based techniques

- Self-verifying data
- Comparison with saved results
- Comparison with a specification or other authoritative document
- Heuristic consistency
  - H ICCUPP (see earlier slide)
- Oracle-based testing
Boundaries and Equivalence Classes
(Example Test Series)

Here is the program’s specification:

- This program is designed to add two numbers, which you will enter
- Each number should be one or two digits
- The program will print the sum. Press Enter after each number
- To start the program, type ADDER

Before you start testing, do you have any questions about the spec?
Boundaries and Equivalence Classes
(Example Test Series)

Example continued: The First Cycles of Testing

Basic strategy for dealing with new code:

1. Start with mainstream-level test. Test the program with easy-to-pass values that will be taken as serious issues if the program fails.

2. Test broadly, rather than deeply. Check out all parts of the program quickly before focusing.

3. Look for more powerful tests. Once the program can survive the easy test, put on your thinking cap and look systematically for challenges.

4. Pick boundary conditions. There will be too many good tests. You need a strategy for picking and choosing.

5. Do some exploratory testing. Run new tests every week, from the first week to the last week of the project.
Equivalence Classes & Boundary Analysis

There are too many tests!
There are $199 \times 199 = 39,601$ test cases for valid values:
- Definitely valid: 0 to 99
- Might be valid: -99 to -1

There are infinitely many invalid cases:
- 100 and above
- -100 and below
- Anything non-numeric

Some people want to automate these tests.
- Can you run all the tests?
- How will you tell whether the program passed or failed?

We cannot afford to run every possible test. We need a method for choosing a few tests that will represent the rest. Equivalence analysis is the most widely used approach.
To avoid unnecessary testing, partition (divide) the range of inputs into groups of equivalent tests.

Then treat an input value from the equivalence class as representative of the full group.

Two tests are equivalent if we would expect that an error revealed by one would be revealed by the other.

Boundaries mark the point or zone of transition from one equivalence class to another. The program is more likely to fail at a boundary, so these are good members of equivalence classes to use.

Myers p.45 and Black Box Software Testing, Kaner
Traditional Presentation (Myers)

One input or output field

- The “valid” values for the field fit within one (1) easily specified range.
- Valid values: -99 to 99
- Invalid values
  - < -99
  - > 99
Building the Table (in practice)

Relatively few programs will come to you with all fields fully specified. Therefore, you should expect to learn what variables exist and their definitions over time.

To build an equivalence class analysis over time, put the information into a spreadsheet. Start by listing variables. Add information about them as you obtain it.

The table should eventually contain all variables. This means, all input variables, all output variables, and any intermediate variables that you can observe.

In practice, most tables that I’ve seen are incomplete. The best ones that I’ve seen list all the variables and add detail for critical variables.
### Equivalence Classes & Boundary Analysis:

#### Myers’ Boundary Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Valid Case Equivalence Classes</th>
<th>Invalid Case Equivalence Classes</th>
<th>Boundaries and Special Cases</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Number</strong></td>
<td>-99 to 99</td>
<td>&gt;99</td>
<td>99, 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;-99 non-number expressions</td>
<td>-99, -100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>null entry</td>
<td></td>
</tr>
<tr>
<td><strong>Second Number</strong></td>
<td>same as first</td>
<td>same as first</td>
<td>same</td>
<td></td>
</tr>
<tr>
<td><strong>SUM</strong></td>
<td>-198 to 198</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Are there other sources of data for this variable? Ways to feed it bad data?</td>
<td></td>
</tr>
</tbody>
</table>
Bug Reporting / Advocacy

You are what you write

- Bug reports are the primary work of testers
- Programmers rely on your reports for vital information

Your advocacy drives the repair of the bugs you report

- Any bug report that you write is an advocacy document that calls for the repair of the bug

Make your bug report an effective sales tool
Exercise: How do you advocate this bug?

Searching for a book by title, author or ISBN:
Test Automation

Kaner, Bach, Pettichord 2001b.
- Use automation when it advances the mission of testing.
- Select your automation strategy based on your context.
- A test tool is not a strategy.
- Test automation is a significant investment.
- Test automation is design and programming.

Fewster & Graham 1999
- “Automating chaos just gives faster chaos”
2. Test Management and Techniques

| 2.1 Testing Fundamentals in 1 hour! |
| 2.2 Test Execution and Techniques |
| 2.3 Heuristic Risk-Based Testing |
| 2.4 Test Management and Metrics |
Heuristic Risk-Based Testing

"This kind of analysis is always available to you, no calculator required."

"A heuristic method: a useful method that doesn’t always work."

Checklist of open-ended questions, suggestions or guidewords.

Two Approaches:
- Inside-Out:
  - "What can go wrong here?"
- Outside-In (predefined lists of risk):
  - Quality Criteria Categories
  - Generic Risk Lists
  - Risk Catalogs

James Bach, 1999a
Heuristic Risk-Based Testing

Inside-out:

- “What can go wrong here?”
- “What risks are associated with this thing?”
- Require technical insight

- Vulnerabilities
  - What weaknesses or possible failures?

- Threats
  - What situation might exploit a vulnerability and trigger a failure?

- Victims
  - Who or what would be impacted by potential failures - and how bad?
Possible Inside-Out-process

1. Sit down with one of the programmers / developers
   - Create a non-threatening, friendly atmosphere (Coffee? Tee? Beer?)
   - Start asking about what he is working on (you need to understand the technology well enough to ask questions!):
     - Can he explain what the functions are supposed to do?
     - Is he aware of how his function fits into the "big picture"? I.e. do he know all (the main) interfaces? And data flows?
     - Do he know where to find the information he needs?
     - Can he draw a (high level) diagram of process / flow?
     - Is he aware of data being read (used), modified, deleted etc.
     - What kind of error / exception handling has been implemented?
     - What are the complex and / or complicated part(s) of his functions? Why?
     - What kind of Unit Testing has been / will be performed? How?
   - Look for Uncertainty, Contradictions, Inconsistency etc. They are all indicators of potential problems; misunderstandings, incorrect implementation and possibly poor unit testing
   - Go and do some testing in these "problem" areas!
Heuristic Risk-Based Testing

Outside-in

- “What things are associated with this kind of risk?”
- Predefined set of potential risks

- Quality Criteria Categories
- Generic Risk Lists
- Risk Catalogs

James Bach, 1999a
Heuristic Risk-Based Testing

Quality Criteria Categories
- Capability
- Reliability
- Usability
- Performance
- Installability
- Compatibility
- Supportability
- Testability
- Maintainability
- Portability
- Localizability

Generic Risk Lists
- Complex
- New
- Changed
- Upstream Dependency
- Downstream Dependency
- Critical
- Precise
- Popular
- Strategic
- Third-party
- Distributed
- Buggy
- Recent Failure

Risk Catalogues
- See Testing Computer Software by Kaner, Falk and Nguyen

James Bach, 1999a
Communicating Heuristic RBT

- Risk Watch List
- Risk / Task Matrix
- Component Risk Matrix

<table>
<thead>
<tr>
<th>Components</th>
<th>Risk</th>
<th>Risk Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printing</td>
<td>Normal</td>
<td>Distributed, popular</td>
</tr>
<tr>
<td>Report Generation</td>
<td>Higher</td>
<td>New, strategic, third-party, complex, critical</td>
</tr>
<tr>
<td>Installation</td>
<td>Lower</td>
<td>Popular, usability, changed</td>
</tr>
<tr>
<td>Clipart Library</td>
<td>Lower</td>
<td>Complex</td>
</tr>
</tbody>
</table>
Other Risk Catalogues

- www.bugnet.com
- www.cnet.com (news)
- Testing Computer Software
  - Includes Kaner, Falk and Nguyen’s list of bugs
- Software Testing Techniques
  - Includes Boris Beizer’s taxonomy of bugs
- Computer-Related Risks
  - Peter G. Neumann
- Managing the Testing Process
  - Includes Rex Black’s list of common software failures
WEB E-business Risks

No second chance – risks outside control

- Unlimited potential users - anyone can visit
- Many Potential Points of Failure
- No control over client platform
- No control over client configuration
- Which browser to support?
- Required plug-ins not available
- The context of web transactions is a risk area
- Cookies
- Network connections
- Firewalls
- Usability
- Internationalisation
- Etc....
2. Test Management and Techniques

2.1 Testing Fundamentals in 1 hour!
2.2 Test Execution and Techniques
2.3 Heuristic Risk-Based Testing
2.4 Test Management and Metrics
Test Management

- Establish a strong and supportive testing role.
- Re-visit your strategy and mission every day.
- Advocate for bugs and testability.
- Maintain three lists: risks, issues, coverage.
- Test with your team.
- Continuously report the status and dynamics of testing.
Metrics - Overview

- Defect Analysis
  - Defect Density
  - Defect Aging
  - Defect Trends

- Planning and Progress tracking
  - Planned vs. Actual
  - Progress indicators
  - Number of hours (days) remaining? Estimated to Complete (EtC)

- Iterative Projects - Trend Analysis

- Coverage
  - Identifies how “complete” the testing is (will be)
  - Defect Detection Percentage - DDP (Overall test efficiency)

- Further reading: Fenton (1997)
Planning and Progress Tracking

On-line Test Cases Completed

- Planned
- Executed
- QAed

Date

Number of Test Cases

Introduction
ET Management and Techniques
ET Planning and Documentation
ET Styles
ET Management

ET Workshop v. 1.20 - Management and Techniques
©2002 Amland Consulting
“To be fixed” vs. “Actually fixed”

“To be Re-tested” vs. “Actually Re-tested”
Progress Indicators - Hours

Hours Indicators

- Number of hours for finding one fault and for fixing one

Hours per Fault for Test and Fix

Date

Hours per Fault

Test

Fix
“Estimated to Complete”

ETC for system test based on:
- Number of hours testing per fault found
- Number of hours fixing per fault
- Number of faults found per function
- Number of fixes being rejected
- Number of remaining tests (functions to be tested)

Calculated ETC and Actual Hours

Date

Estimated to Complete at Time t

Actual to Complete at Time t
Iteration Trend Analysis

Iteration 1

Verification Points

Days

Approx. 7200

Approx. 6000

Iteration 2

Verification Points

Days

Approx. 6200

Approx. 4200

Iteration 3

Verification Points

Days

Approx. 6800

Approx. 5700
Test planning and execution coverage

- Identify units to be tested:
  - Use Cases
  - Use Case flows
  - Requirements
  - Etc.

- Build and prioritise test Cases (based on risk)

- Track:
  - Test preparation/planning: number of test cases built for each priority
  - Test Execution: number of tests executed for each priority, with status
Defect Detection Percentage

\[
\frac{Total\ Number\ Of\ Bugs\ In\ Test + Total\ Number\ Of\ Bugs\ In\ Production}{Total\ Number\ Of\ Bugs\ In\ Test}
\]
Example: Requirements Coverage

Diagram generated from TestDirector, Mercury Interactive
Risk - ordinal scale!

Ordinal Scale:
- The numbers are ordered but no arithmetic can be applied.
- A Risk Exposure of 3 IS bigger then 2
- We do NOT know HOW much bigger!
Ordinal vs. Linear Scales

Average OK?
Gilb’s Law

"Anything can be made measurable in some way, which is superior to not measuring it at all."
Einstein...

"Not everything that counts can be counted. Not everything that can be counted, counts."
Management and Techniques Summary

Introduction

Test Management and Techniques

ET Planning, Exec. and Documentation

ET Styles

ET Management

- Testing Fundamentals in 1 hour!
- Black Box and Glass Box testing
- Test Planning and Management
- Test Documentation
- Test Execution and Techniques
- Heuristic Risk-Based Testing
- Progress Tracking and Metrics