China Test 2012

The Role of Testers Throughout the Project Lifecycle

Richard Bender
Bender RBT Inc.
17 Cardinale Lane
Queensbury, NY 12804
518-743-8755
rbender@BenderRBT.com
http://www.BenderRBT.com

Copy Right Notice

Goals

- Deliver **MORE** function,
- In **LESS** time,
- With **FEWER** resources,

and

- With **HIGHER QUALITY**.

Project Averages - 2010

- Cost overruns 46%
- Schedule overruns 71%

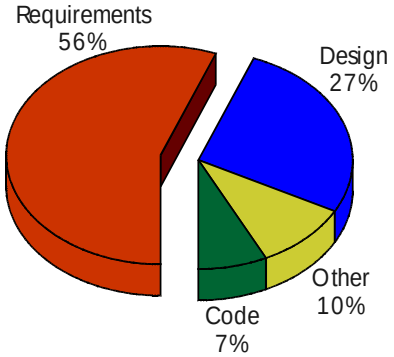
Standish Group – 2011 Chaos Manifesto

Relative Cost to Fix an Error

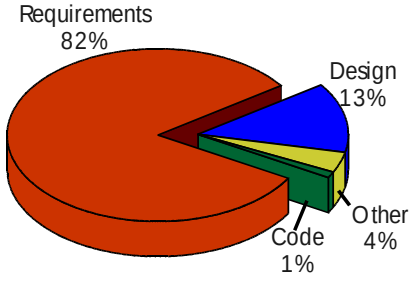
Phase in which found:	Cost Ratio:
Requirements	1
Design	3-6
Coding	10
Development Testing	15-40
Acceptance Testing	30-70
Operation	40-1000

(IBM, GTE, et.al.)

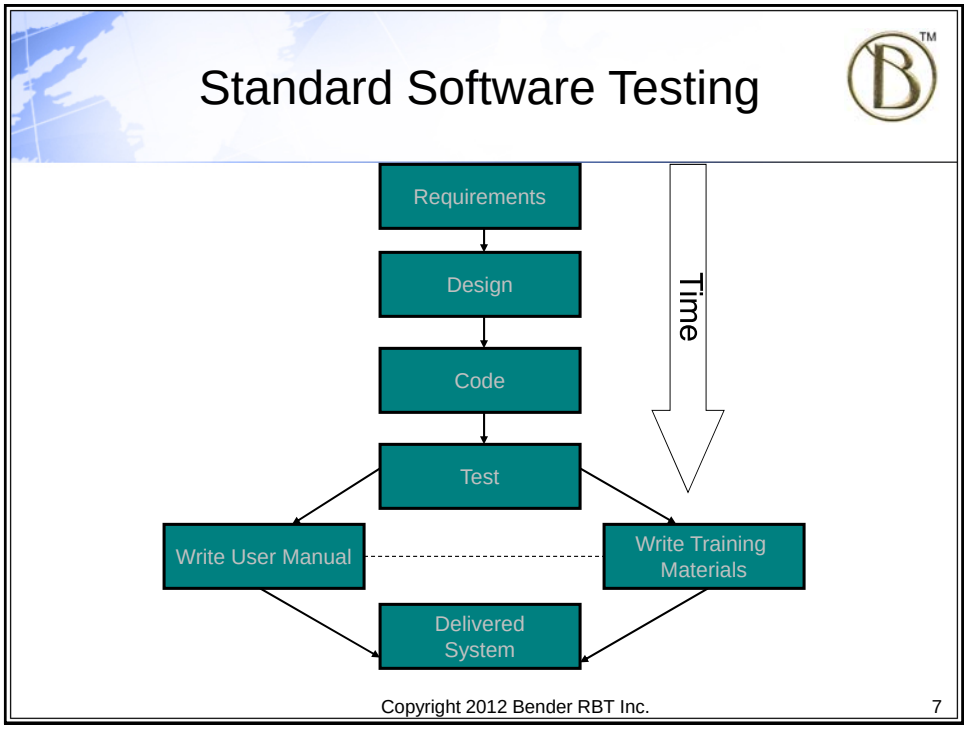
Distribution of Bugs

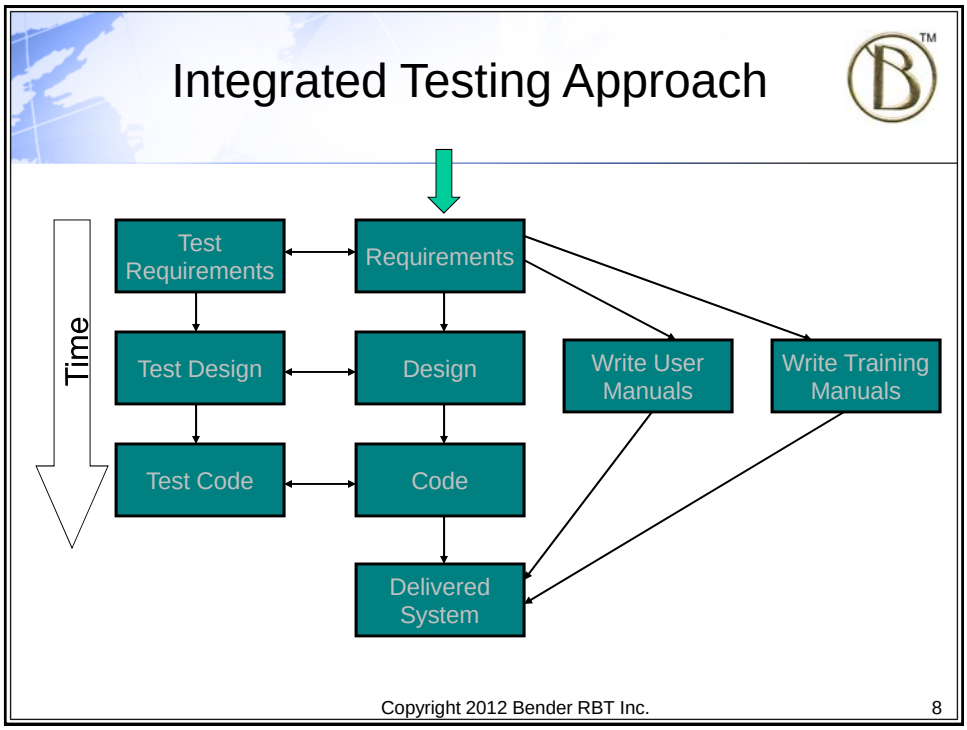


Distribution of Effort to Fix Bugs



(James Martin)


Standard Software Testing
Requirements
Design
Code
Test
Write User Manual
Write Training Materials
Delivered System
Time
Copyright 2012 Bender RBT Inc.
7


Integrated Testing Approach
Test Requirements
Requirements
Test Design
Design
Test Code
Code
Write User Manuals
Write Training Manuals
Delivered System
Time
Copyright 2012 Bender RBT Inc.
8

Test Then Build

- Timely:
  - Integrated throughout the life cycle
- Effective:
  - Rigorous test definition
- Efficient:
  - Strong automation
  - Minimum number of tests
- Manageable:
  - Measurable
  - Predictable

The Process

1. Validate requirements against objectives.
2. Apply scenarios against requirements.
3. Perform initial Ambiguity Review.
4. Perform domain expert reviews.
5. Create cause-effect graph.
6. Logical consistency check and test cases designed by RBT.
7. Review test cases with requirements author.
8. Validate test cases with users/domain experts.
9. Review test cases with developers.
10. Walk test cases through design.
11. Walk test cases through code.
12. Verify code against test cases designed from the requirements.
13. Supplement tests for design dependent features

Validate Requirements Against Objectives

Objectives define **WHY** the system is being created.

Fully Qualified Objectives:

- Identify goals/desired return on investment.
- Identify constraints - e.g., time, resources.

Example:

- Reduce operational expenses by 10% by year end 2013 with project expenditures not to exceed $2 million.

Validate Requirements Against Objectives

Objective:

- Comply with a regulatory requirement to supply the government with requested data within 5 days of the request.
- Delays are subject to fines.

Initial Solution:

- A real-time, on-line, database application costing hundreds of thousands of dollars.

Alternative Solution:

- Two part-time clerks.

Validate Requirements Against Objectives

- Ensures proper scope for high-level requirements.
- Ensures application rules stay focused.
- Critical for managing scope changes.
- This is an on-going activity against each iteration of the requirements and design.

Apply Scenarios Against Requirements

- Scenario - “what if” action by the user
  - A user may be a person, software, or even hardware
  - Scenarios are task oriented
- Goal:
  - Verify that the requirements are robust enough to handle all of the tasks initiated by the users.
  - Also important in verifying usability.

Apply Scenarios Against Requirements

Example - Transfer an Employee:

- Permanent versus temporary transfer
- Relocation involved or not
- Relocation in country or overseas
- Promotion, demotion, lateral transfer
- Spouse works for the company

Permanent transfer overseas, with a promotion, has a house, spouse works for the company, school aged children – what processes do you need?

Tour Based Scenarios

- Tours – testing the software via intent to do something specific
  - A refinement of Exploratory Testing
  - Strongly analogy based
  - Primarily used once the code has been written
- Goal:
  - Verify that the code/requirements are robust enough to handle all of the activities required of the system.

Apply Tours Against Requirements

- Guidebook Tour
  - Follow the user manual faithfully through each feature
  - Follow a competitors' user manual faithfully through your version of the feature
  - Follow blogs from third parties through each feature
- Crime Spree Tour
  - Enter bad data at each interface point

Apply Scenarios Against Requirements

- Testers are often the only ones with an end-to-end view of the system
- Testers are better at identifying exception cases (developers build things, testers break them)

Ambiguous Requirements

- Requirements are written primarily in natural language.
- All natural languages are inherently ambiguous.
- Formal specification languages are not a viable alternative.
- Using structured, natural language helps considerably.
- Most developers do not know how to write detailed unambiguous requirements.

Definition of Ambiguous

- If one person wrote it with one intent and another person read it differently, it is ambiguous.

Pitfalls of Natural Language

A difference between version I and version II exists only when mixed data types are used, and then only when operand lengths differ, and then only sometimes.

Ambiguity Review Checklist

- Dangling Else
- Ambiguity Of Reference
- Scope Of Action
- Omissions
  - Causes Without Effects
  - Missing Effects
  - Effects Without Causes
  - Complete Omissions
  - Missing Causes
- Ambiguous Logical Operators
  - Or, And, Nor, Nand
  - Implicit Connectors
  - Compound Operators
- Negation
  - Scope Of Negation
  - Unnecessary Negation
  - Double Negation
- Ambiguous Statements
  - Verbs, Adverbs, Adjectives
  - Variables, Unnecessary Aliases
- Random Organization
  - Mixed Causes And Effects
  - Random Case Sequence
- Built-In Assumptions
  - Functional/Environmental Knowledge
- Ambiguous Precedence Relationships
- Implicit Cases/Variables/Actions
- Etc.
- I.E. Versus E.G.
- Temporal Ambiguity
- Boundary Ambiguity

Dangling Else

MUST BE, WILL BE, IS ONE OF, SHOULD BE, COULD BE, CAN BE, SHALL, NORMALLY.

EXAMPLE:

"The Marriage Status must be either Married, Single, or Divorced."

Else? An error condition?

Ambiguity of Reference

IT, SUCH, THE ABOVE, THE PREVIOUS, THEM, THESE, THEY

EXAMPLE 1:

"Add field A to field B.

This number must be positive."

EXAMPLE 2:

"Transaction 1 displays the customer's name and address.

Transaction 2 displays the customer's account numbers.

Transaction 3 displays the customer's account balances.

Such transactions require the security code."

Additional Benefits from Ambiguity Reviews

- Timely feedback reduces issue resolution time.
- Explicit feedback leads to defect avoidance.
- Writing is accelerated.
- Critical to outsourcing.

Cultural Difference

- Western Culture – Transmitter Oriented
  - Responsibility of writer/speaker to be clear
- Eastern Culture – Receiver Oriented
  - Responsibility of the reader/listener to understand

Standish Group – 2009 Chaos Manifesto

Savings Via Ambiguity Reviews

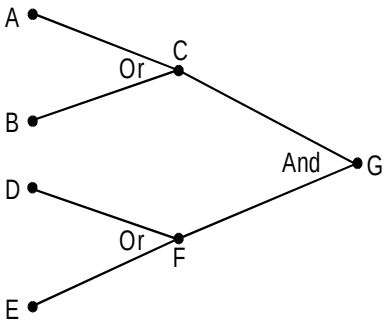
Ambiguity Reviews of Specifications
(source: Bender RBT Inc.)

- Costs per defects found
  - $25 per severity 1 or 2
- Costs if found in integration test/system test
  - $750 to $3,000 per defect (SEI)
- Cost if found in production
  - Many, many thousands of dollars

If something is ambiguous in the specs it will nearly always result in a defect(s) in the code

Cause-Effect Graphing

1. If A or B, then C.
2. If D or E, then F.
3. If C and F, then G.



- Resolve Aliases
- Clarify Precedence Rules
- Clarifies Implicit Information
- Begin Integration Test

Constraints

- Boundary Constraints
  - Limit which combinations of data can arrive at the function
    - **limited by pre-edited data**
    - **limited by the physical structure of the data**

 29

Constraints

Why are constraints important?

- Critical to designing physically possible tests.
- Used to check logical consistency of the rules.
- Part of reconciling the data model to the process model.

 30

Constraints

Boundary Constraints (Pre-conditions)

a. Exclusive (at most one is true)
b. One and Only One
c. Inclusive (at least one is true)
d. Requires
e. Attribute Mask
f. Anchor (object only has one state)

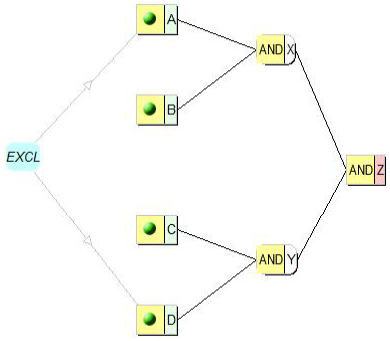
Constraints & Logical Inconsistencies

- Issue:
  - Preconditions may conflict with intra-process rules
- Critical testing issue
  - Multiple applications sharing common data
  - Complex domain and relational edits
  - Only the Cause-Effect Graphing technique addresses this

Constraints & Logical Inconsistencies

Example:

- X and Y can never be true at the same time, therefore Z is always false



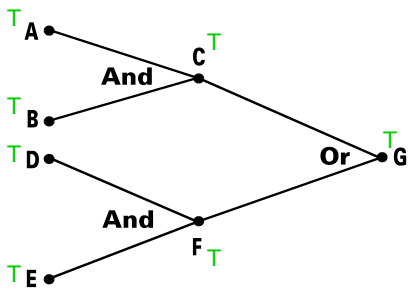
Test Design Algorithm Using the Cause-Effect Graph

- Based on hardware logic circuit testing via sensitizing paths.
- Logic defect rates in circuits are much better than six sigma
  Less than one defect per billions of gates!
- Rules are rules are rules whether in software, hardware, or firmware.

Cause-Effect Graphing

Observable Events and Path Sensitizing

- Assume C and F are not observable events.
- Assume A is stuck at FALSE.
- Enter as a test case A(T), B(T), D(T), E(T).
- Results should be C(T), F(T) and G(T).



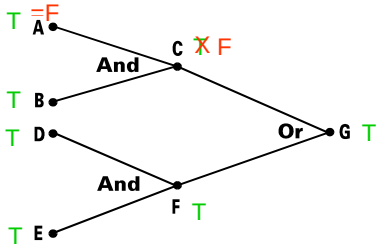
RBT24261

35

Cause-Effect Graphing

Observable Events and Path Sensitizing

- Results should be C(T), F(T) and G(T).
- A, stuck at FALSE, causes C to be (F).
- The error is not detected since G is still (T) due to F(T).
- Therefore, no test of C can be combined with tests of F which would result in F(T).



36

Test Path Sensitizing

Challenge:

Design a set of test cases factoring in:

- the relations between the nodes
- the constraints on the states
- the functional variations and
- node observability,

such that if any logical defect or any combination of defects are present, at least one test case will fail at an observable point.

Test Path Sensitizing

Diagnostic Probe Points

Some Functional Variations Identified As Untestable

Solution:

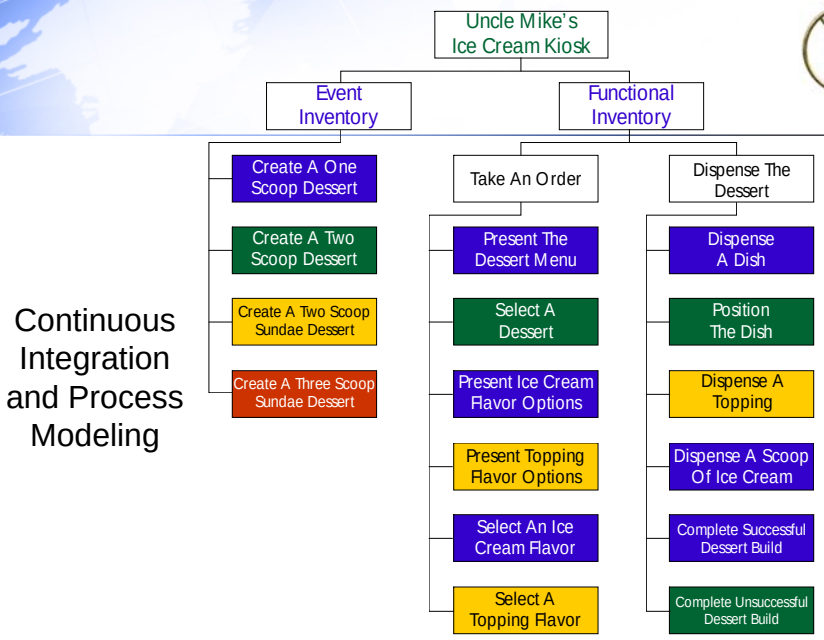
Force normally unobservable nodes to be observable.

The Testers are responsible for defining the testability requirements for the system

Additional Points of Integration

- Start continuous integration testing in requirements definition.
- Ambiguity Reviews influence requirements writing.
- Test Case Reviews identify additional errors in requirements.

Continuous Integration and Process Modeling



Continuous Integration and Process Modeling

- Perform Ambiguity Reviews of each process.
- Design Test Cases for each individual process.
- String the tests together into end-to-end test suites.
- Sequence the requirements writing to allow for early full thread test definitions.
- Sequence design and coding to continue to exploit early integration opportunities.

Continuous Integration and Process Modeling

- Identifies inconsistent interfaces.
- Identifies process gaps/redundancies.
- If you can not integrate the requirements, how will you integrate the code?

Using the Test Cases to Review the Requirements

Typical poorly written requirement

Dental Insurance Claims Payment Specification

Dentists with membership codes of 2, 3, or 9 are member dentists. For claims referencing a non-member dentist or for procedures not within the referenced dentist's record, a system table is used to calculate the amount paid. Otherwise, the amount submitted is paid. However, an override code of 1 or 9 allows the amount submitted to be paid for non-member dentists or for procedures not within the referenced dentist's record. When an override code is used an entry is made on the paid claims report.

Test Case Reviews
Dental Insurance Claims Payment

TEST 1

Cause States:
- The Dentist is a Member Dentist
- The procedure was not preauthorized
- An override code was entered

Effect States:
- Pay the full amount of the claim
- Make an entry on the paid claims report

TEST 2

Cause States:
- The Dentist is a Member Dentist
- The procedure was preauthorized

Effect States:
- Pay the full amount of the claim
- Do not make an entry on the paid claims report

Test Case Reviews

Tester Reviews Test Cases Produced by BenderRBT

- Verify tests match tester's understanding of the requirements.
- Tester identifies errors and omissions.

Tests Reviewed With Requirements Author

- Insures that tester and author are in sync.
- Author identifies missing cases and corrects the requirements.

Test Case Reviews

Tests Validated by Users/Domain Experts

- Users identify errors and omissions.

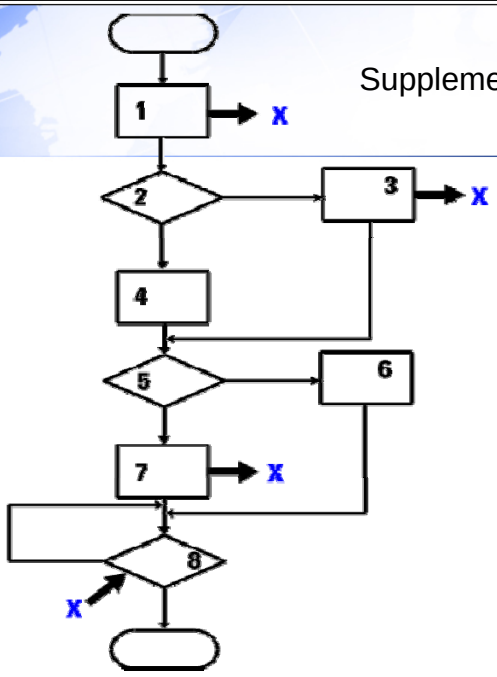
Tests Reviewed With Developers

- Clarifies developers' understanding of the requirements.
- Ensures design/code will match the requirements.

Note:

Sometimes the tests are the only version of the detailed rules available for review.

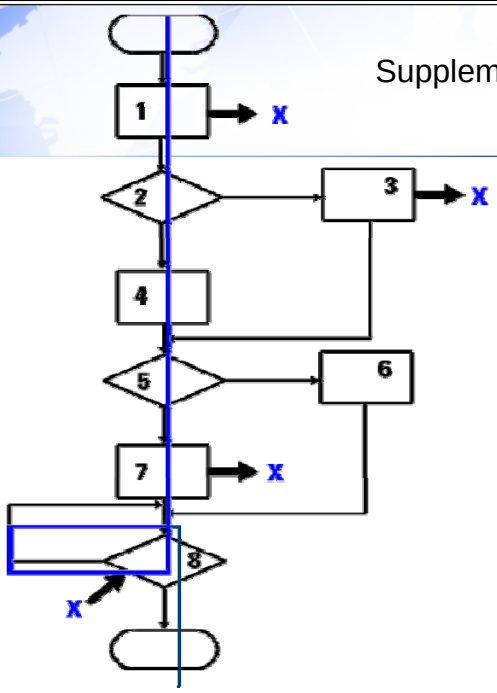
Supplement Tests



RBT26040

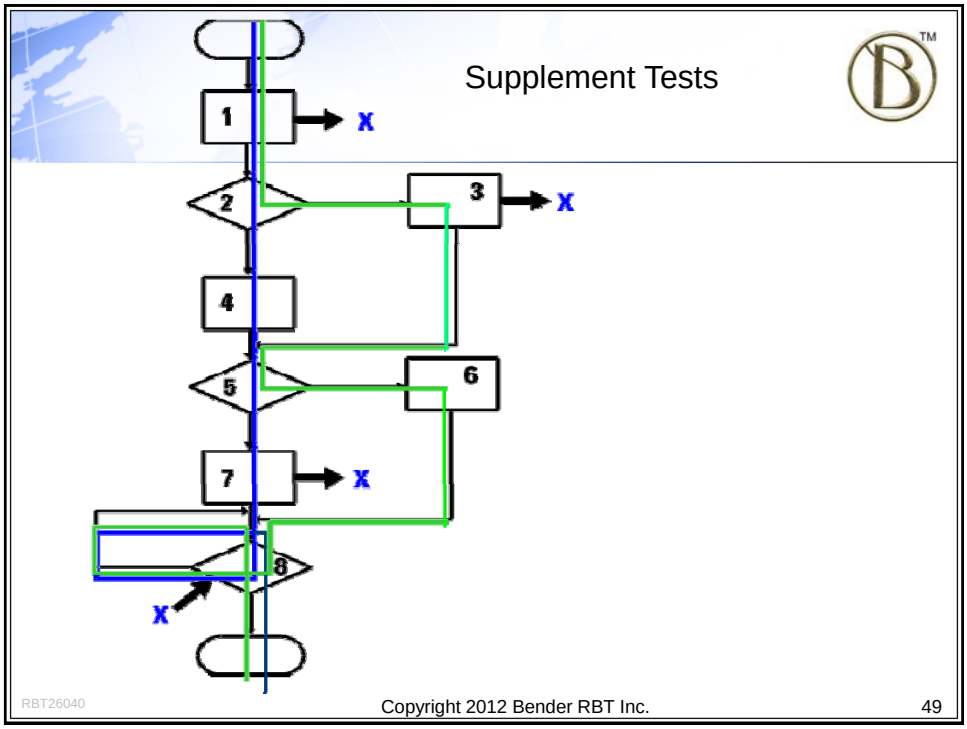
47

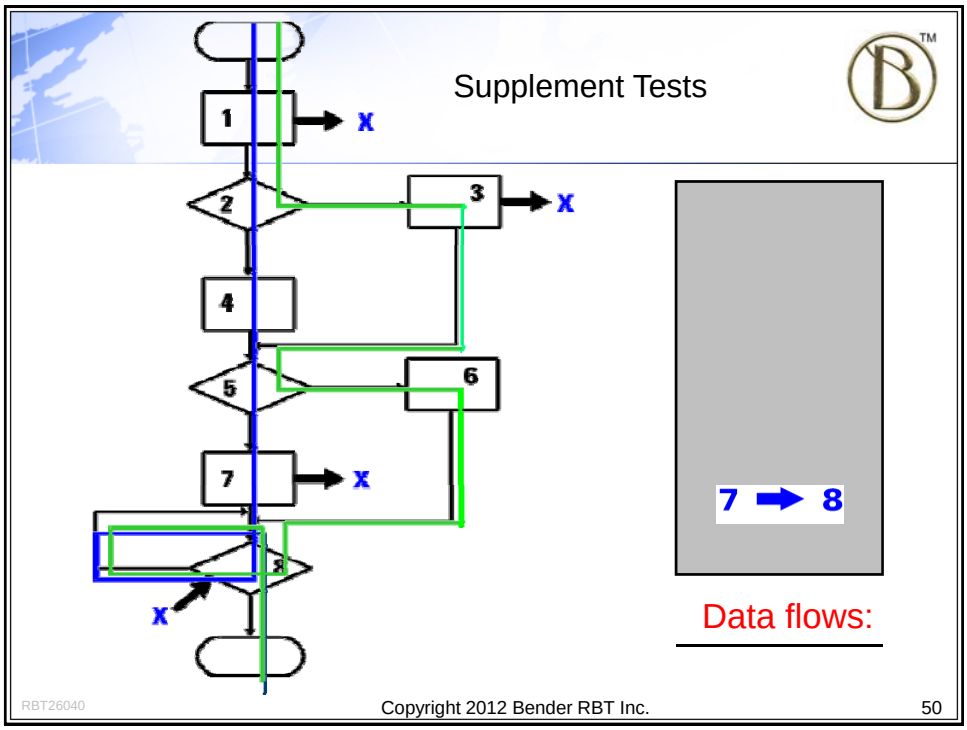
Supplement Tests

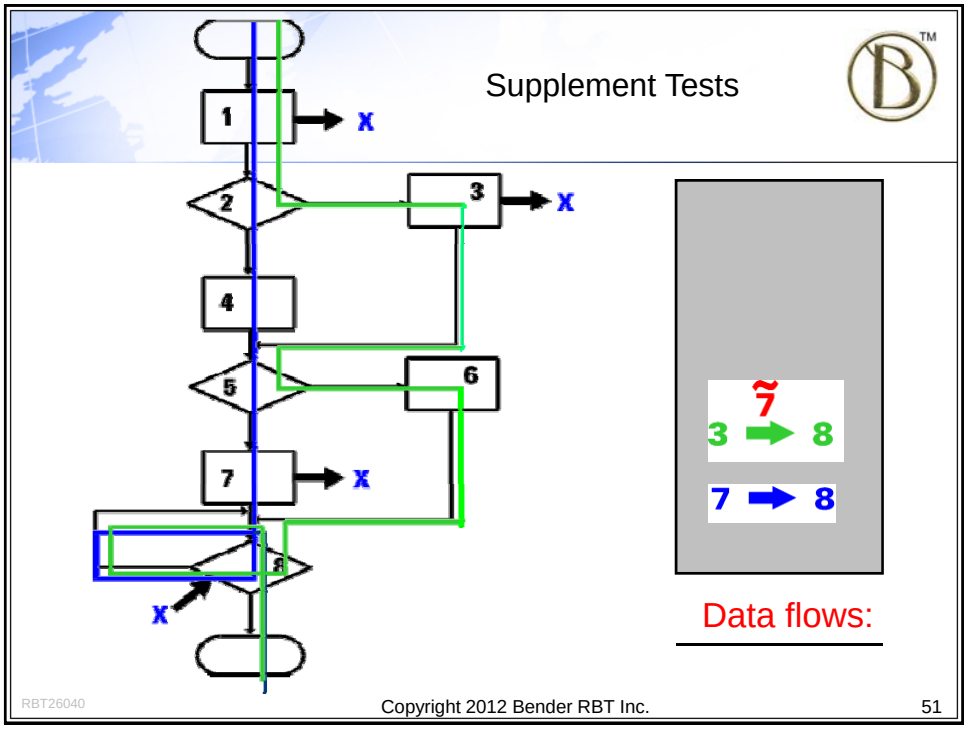


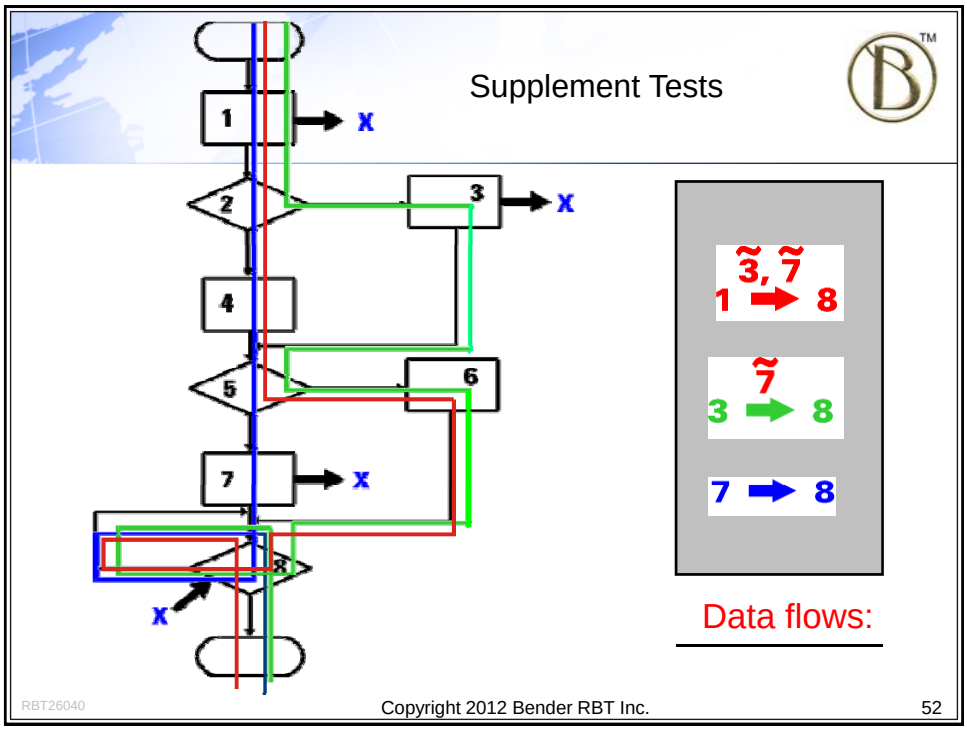
RBT26040

48


Supplement Tests
1
X
2
3
X
4
5
6
7
X
8
X
RBT26040
Copyright 2012 Bender RBT Inc.
49


Supplement Tests
1
X
2
3
X
4
5
6
7
X
8
X
7 → 8
Data flows:
RBT26040
Copyright 2012 Bender RBT Inc.
50


Supplement Tests
1
2
3
4
5
6
7
8
X
X
X
X
$\tilde{7}$
3 → 8
7 → 8
Data flows:
RBT26040
Copyright 2012 Bender RBT Inc.
51


Supplement Tests
1
2
3
4
5
6
7
8
X
X
X
X
$\tilde{3}, \tilde{7}$
1 → 8
$\tilde{7}$
3 → 8
7 → 8
Data flows:
RBT26040
Copyright 2012 Bender RBT Inc.
52

Supplement Tests

- Cannot achieve 100% code coverage with requirements-based tests.
- Can achieve 80% to 90% with RBT (have reached 95%).
- Build initial tests via RBT. Then supplement as needed with Code-Based Testing to achieve 100% code coverage.

Note: Most test cases can be designed before the code is written!!

Tracking The Test Effort

- Project = (Time, Resources, Function, Quality)
  - Time - Calendar
  - Resources - Spreadsheet
  - Function - ???
  - Quality - ???

Tracking The Test Effort - Status Reporting

- Quantifiers
  - # Functions
  - # Functional Variations
  - # Function Points
  - # Test Cases
  - # Modules
  - # Lines of Code
- Status
  - Not Executed
  - Failed Execution
  - Successful Execution
- Error Analysis
  - # Errors by Type, Severity
  - Mean Time Failure
  - Mean Time to Fix

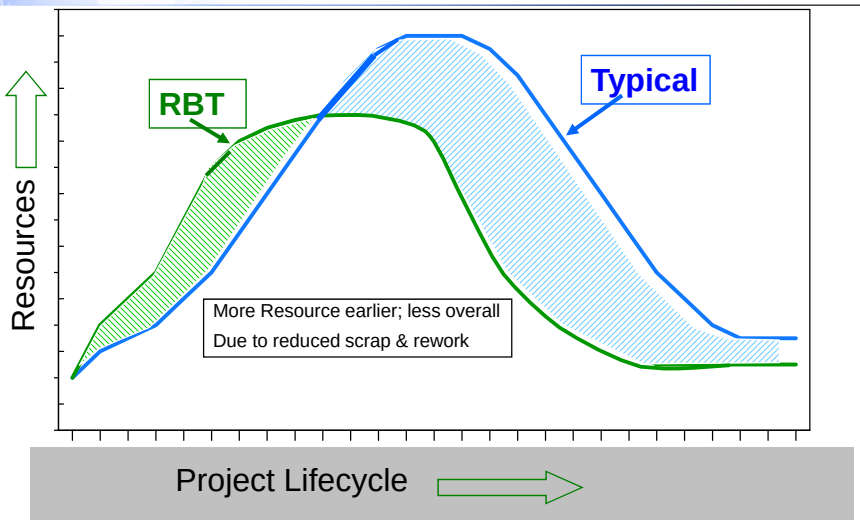
Tracking The Test Effort

- Management can make informed decisions, based on quantitative measures, as to whether or not to deploy the system.

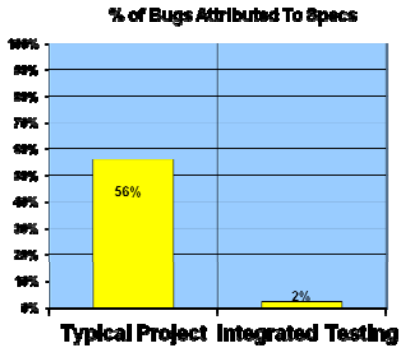
Impact on Staffing

- Staffing Curve Peaks Earlier
  - Requirements written in more detail.
  - Design concurrent with requirements.
  - Implementation preparation concurrent with design.
  - Testers involved from the beginning.
  - Technical writers involved earlier.
- Total Resources Reduced
  - Minimize scrap and rework.
  - Plans have better focus on scope and priorities.

Staffing Curve



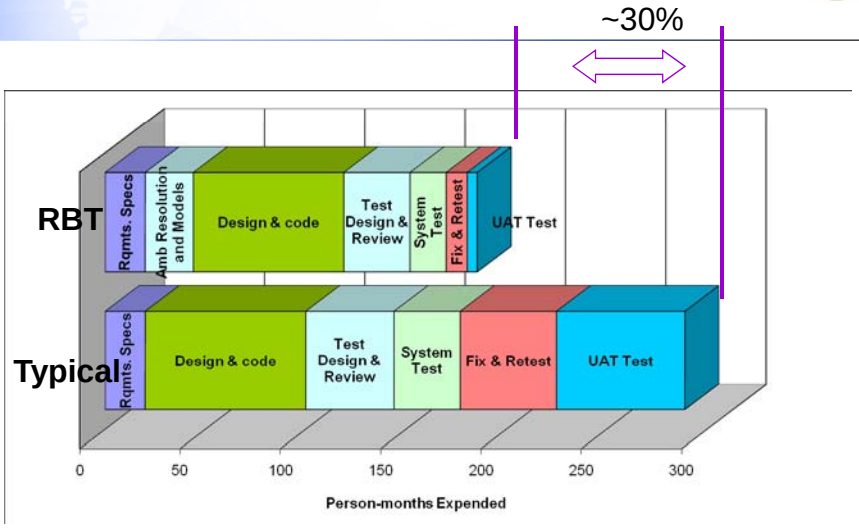
Early Defect Detection
Defect Avoidance



Results

- Shortened schedules due to **INCREASED PARALLEL EFFORTS**
- Reduced resources due to **MINIMIZING SCRAP & REWORK**
- Improved quality due to **REQUIREMENTS-BASED TESTING** and **INTEGRATED TESTING**

RBT vs. Typical Resources



What Integrated Testing Delivers

- Time to deliver
  - Reduced 20% to 30%
- Cost to deliver
  - Reduced 20% to 30%
- Residual defect rate
  - Reduce to zero or near zero