Module 2
Core Concepts of Software Testing

Principles of Software Testing for Testers
Module 2: Core Concepts of Software Testing

Topics

Objectives

Objectives

- ◆ Introduce foundation topics of functional testing
- ◆ Provide stakeholder-centric visions of quality and defect
- ◆ Explain test ideas
- ◆ Introduce test matrices

2

Rational
the software development company

Definitions

Module 2 Content Outline

➔ **Definitions**
  ▪ Defining functional testing
  ▪ Definitions of quality
  ▪ A pragmatic definition of defect
  ▪ Dimensions of quality
  ◆ Test ideas
  ◆ Test idea catalogs
  ◆ Test matrices

3

Rational
the software development company

Functional Testing

Functional Testing

- ◆ In this course, we adopt a common, broad current meaning for functional testing. It is
  - ▪ Black box
  - ▪ Interested in any externally visible or measurable attributes of the software other than performance.
- ◆ In functional testing, we think of the program as a collection of functions
  - ▪ We test it in terms of its inputs and outputs.

Rational
the software development company

Functional testing started with the proposal that we treat a program as a function. To test it, we would feed it inputs and check its outputs.

In functional testing, knowledge of the inner workings of the "function" is less important than knowing what the function is *supposed to* do. For that reason, functional testing is often called

- *Black box testing* (testing the program as a "black box" without knowledge of the internals) or

- *Behavioral testing* (focusing on the visible behavior of the program in response to, which may or may not be predicted from, analysis of the source code).

Functional testing was sometimes distinguished from "non-functional" testing, which looked at "qualities of service" characteristics of the program that spanned many functions, such as performance or usability.

Discussion Exercise 2.1: Define Quality

Discussion Exercise 2.1: Define *Quality*

- ◆ Form pairs
- ◆ Define *quality*
  - ▪ Write the definition down
  - ▪ Is this your company's definition?
  - ▪ Is this your partner's company's definition?
  - ▪ Does your partner agree with his/her corporate definition of quality?

5

Rati**o**nal®
the software development company

Exercise

What is your definition of "quality"?

1. Take ten minutes.

2. Break into pairs.

3. Agree with your teammate on a definition of quality.

4. Write it down so that you can share it with the class.

How Some Experts Have Defined Quality

<div style="border:1px solid">

How Some Experts Have Defined Quality

- Fitness for use (Dr. Joseph M. Juran)
- The totality of features and characteristics of a product that bear on its ability to satisfy a given need (American Society for Quality)
- Conformance with requirements (Philip Crosby)
- The total composite product and service characteristics of marketing, engineering, manufacturing and maintenance through which the product and service in use will meet expectations of the customer (Armand V. Feigenbaum)
- Note absence of "conforms to specifications."

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved   6

Rational
the software development company

</div>

Many software developers and QA staff define defects in terms of a failure to conform to a specification. Unfortunately, the product can be defective even if it conforms perfectly to a (defective) specification. The spec is a partial description of the intended product.

Quality is not defined in terms of match to a spec. It is defined in terms of match to the needs of the stakeholders.

References for these definitions:

J. M. Juran & Frank Gryna, *Quality Planning & Analysis: From Product Development Through Use, 2nd Edition,* 1980.

Jack Campanella (Ed.), *Principles of Quality Costs: Principles, Implementation and Use, 2nd Edition,* 1990.

Philip R. Crosby, *Quality Without Tears*, 1984.

Armand Feigenbaum, *Total Quality Control,Revised (Fortieth Anniversary Edition),* 1991.

Quality As Satisfiers and Dissatisfiers

Quality As Satisfiers and Dissatisfiers

- ◆ Joseph Juran distinguishes between Customer *Satisfiers* and *Dissatisfiers* as key dimensions of quality:
  - ▪ Customer Satisfiers
    - • the right features
    - • adequate instruction
  - ▪ Dissatisfiers
    - • unreliable
    - • hard to use
    - • too slow
    - • incompatible with the customer's equipment

Rational®
the software development **company**

Prof. Kaner provides these examples:

- Satisfiers are the aspects of the program that make you want to buy it and want to keep using it. Feature comparison lists are comparisons of satisfiers. Consumer Reports product reviews primarily focus on satisfiers.

- Dissatisfiers are the aspects of the program that make you complain about it or decide not to keep using it.
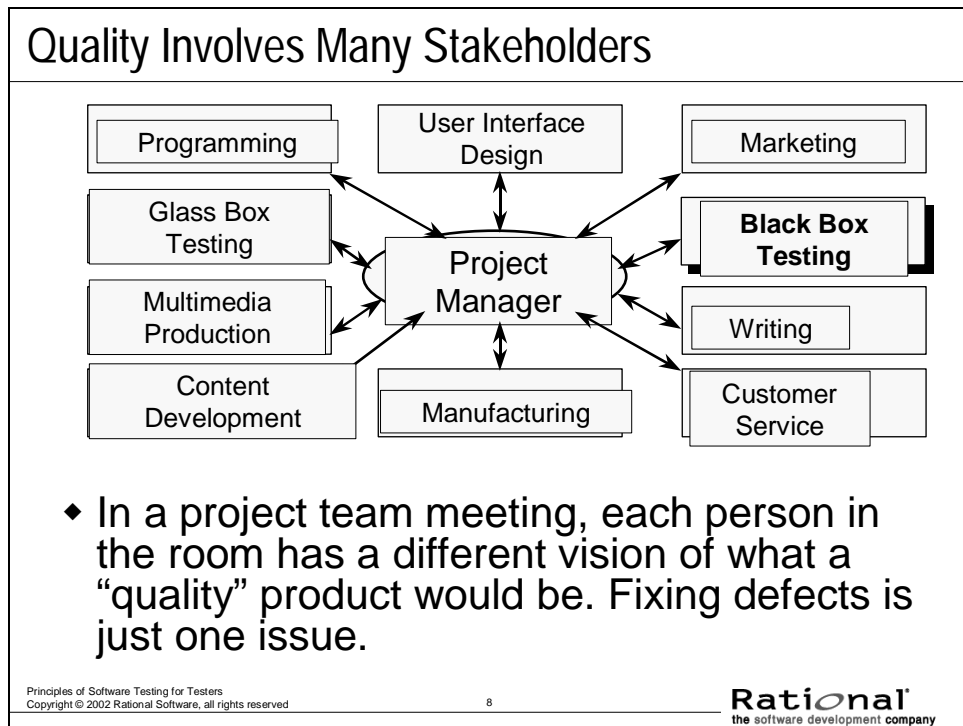
A good illustration of the distinction between a focus on satisfiers and a focus on dissatisfiers is the often repeated debates between marketers and testers:

- One group will shout about the need for features

- The other will answer, "First fix what you have…"

For Juran's discussion of these terms, see:

J.M. *Juran, Juran on Planning for Quality*, 1988

Quality Involves Many Stakeholders

Quality Involves Many Stakeholders

Programming | User Interface Design | Marketing
Glass Box Testing | Project Manager | **Black Box Testing**
Multimedia Production | | Writing
Content Development | Manufacturing | Customer Service

- In a project team meeting, each person in the room has a different vision of what a "quality" product would be. Fixing defects is just one issue.

Rational
the software development company

In commercial software products, there are many stakeholders and many points of view. Different people, different visions:

For example:

Localization Manager: *A good product is easy to translate and to modify to make it suitable for another country and culture. Few experienced localization managers would consider acceptable a product that must be recompiled or relinked to be localized.*

Tech Writers: *A high quality program is easily explainable. Aspects of the design that are confusing, unnecessarily inconsistent, or hard to describe are marks of bad quality.*

Marketing: *Customer satisfiers are the things that drive people to buy the product and tell their friends about it. A Marketing Manager who is trying to add new features to the product generally believes in these attempts to improve the product.*

Customer Service: *Good products are supportable. They have been designed to help people solve their own problems or to get help quickly.*

Programmers: *Great code is maintainable, well documented, easy to understand, well organized, fast and compact.*

Exercise 2.2: Quality Has Many Stakeholders (1/2)

- A programming group agrees on variable naming conventions and follows them. Then they hire a new senior programmer, who won't follow the conventions. Programming the way he did in the 1970's, he gives variables names like Mabel and Al. Some of the other programmers are unhappy about this and so they want to enter a bug report into the bug tracking system, saying that "Mabel is not a proper variable name."

- **Question**: Does this report belong in the change request system? Why or why not?

9

Rational
the software development company

Exercise 2.2: Quality Has Many Stakeholders (2/2)

Exercise 2.2: Quality Has Many Stakeholders (2/2)

- A company is developing a product that they will release in English first. The company expects to create other language versions by changing strings and other resource variables from the English version. They plan to localize versions without recompiling or relinking the code. However, the English code has a hyphenation error that doesn't affect English words but will mishandle some German words.

- The localization manager wants this filed as a bug against the English base code. Fixing the problem in the German version would require a code change and recompilation. The American project manager says that it is not a bug because the English version works OK. Who is right? Should the change request go with the English product, the German product, or both?

10

Rational®
the software development company

A Working Definition of Quality

A Working Definition of Quality

Quality is value to some person.

---- Gerald M. Weinberg

11

Rational®
the software development company

From Gerald M. Weinberg, *Quality Software Management: Volume 1, Systems Thinking*, 1997.

Change Requests and Quality
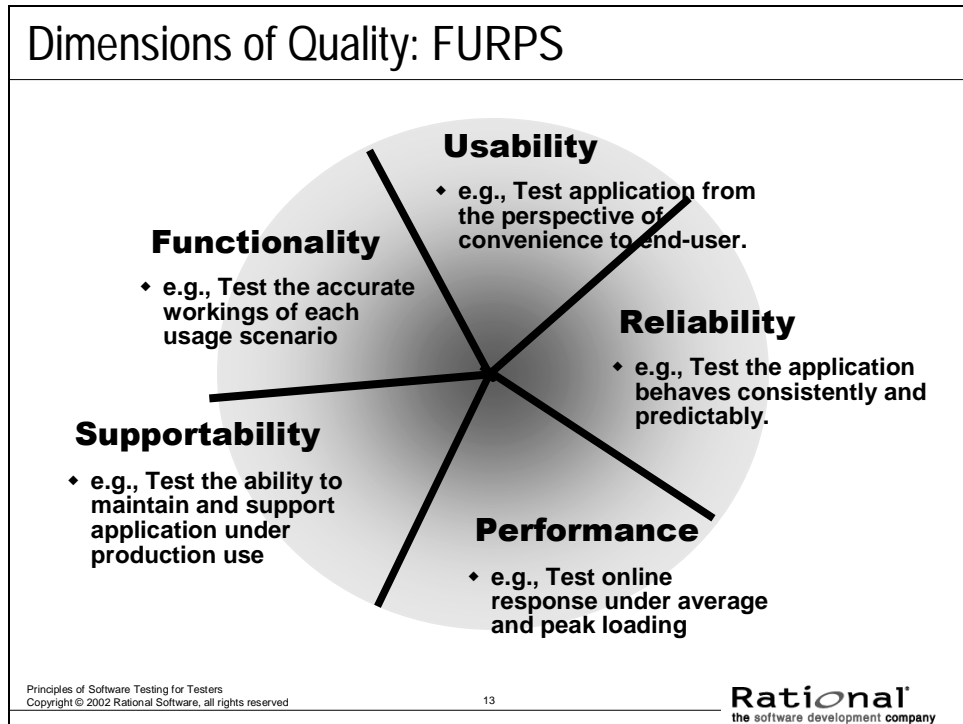
> ### Change Requests and Quality
>
> - A "defect" – in the eyes of a project stakeholder– can include anything about the program that causes the program to have lower value.
>
> - It's appropriate to report any aspect of the software that, in your opinion (or in the opinion of a stakeholder whose interests you advocate) causes the program to have lower value.
>

Early in development, report any defect, even if you believe it will be triaged out.

Later in development, testers should exercise more judgment. It costs time to process a change request—up to 8 hours (all people's time included) in some companies, and problems are less likely to be fixed. Low priority change requests may be a distraction at the end of the project. Processing them takes time away from fixing other problems, and they are often *seen* as a distraction.

Some companies set up a second change request database. Testers report late-found low-priority defects into this database. Other companies tag defects for future releases. The project manager or a senior programmer skims these reports for issues that raise red flags. These requests are re-opened and fully evaluated at the start of development of the next major release.

Dimensions of Quality: FURPS

Dimensions of Quality: FURPS

Usability
- e.g., Test application from the perspective of convenience to end-user.

Functionality
- e.g., Test the accurate workings of each usage scenario

Reliability
- e.g., Test the application behaves consistently and predictably.

Supportability
- e.g., Test the ability to maintain and support application under production use

Performance
- e.g., Test online response under average and peak loading

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved   13

Rational
the software development company

The traditional view of FURPS is:

Functional testing verifies that a system executes the required use-case scenarios as intended. Functional tests may include the testing of features, usage scenarios and security.

Usability testing evaluates the application from the user's perspective. Usability tests focus on human factors, aesthetics, consistency in the user interface, online and context-sensitive help, wizards and agents, user documentation, and training materials.

Reliability testing verifies that the application performs reliably and is not prone to failures during execution (crashes, hangs, memory leaks). Effective reliability testing requires specialized tools. Reliability tests include integrity, structure, stress, contention and volume tests.

Performance testing checks that the target system works functionally and reliably under production load. Performance tests include benchmark tests, load tests, and performance profile tests.

Supportability testing verifies that the application can be deployed as intended. Supportability tests include installation and configuration tests.

A Broader Definition of Dimensions of Quality

<div style="border:1px solid">

A Broader Definition of Dimensions of Quality

- Accessibility
- Capability
- Compatibility
- Concurrency
- Conformance to standards
- Efficiency
- Installability and uninstallability
- Localizability

- Maintainability
- Performance
- Portability
- Reliability
- Scalability
- Security
- Supportability
- Testability
- Usability

Collectively, these are often called *Qualities of Service*, *Nonfunctional Requirements, Attributes*, or simply *the -ilities*

Rational
the software development company

</div>

FURPS is often seen as a conceptually complete system, and it may be. But where would you list "accessibility" in FURPS? (Answer – probably in Usability). Or printer compatibility? (Answer – probably in supportability). Even though many different dimensions fit within the FURPS five, test designers often find it useful to work from a longer list of qualities of service, perhaps generating several test cases of a given feature from each dimension.

Note that you cannot test every area of the program fully against a list of quality dimensions. There are (as always with testing) too many possible tests. Somehow, you will have to find a way to cull the "best ideas out" and test using those. Projects vary in risk and objective, so the "best ideas" list will be different for each program.

Test Ideas

Module 2 Content Outline

- ◆ Definitions
  - ▪ Defining functional testing
  - ▪ Definitions of quality
  - ▪ A pragmatic definition of defect
  - ▪ Dimensions of quality
- ➔ **Test ideas**
- ◆ Test idea catalogs
- ◆ Test matrices

15

Rational
the software development company

Test Ideas

Test Ideas

- ◆ A test idea is a brief statement that identifies a test that might be useful.
- ◆ A test idea differs from a test case, in that the test idea contains no specification of the test workings, only the essence of the idea behind the test.
- ◆ Test ideas are generators for test cases: potential test cases are derived from a test ideas list.
- ◆ A key question for the tester or test analyst is which ones are the ones worth trying.

Rational
the software development company

Exercise 2.3: Brainstorm Test Ideas (1/2)

Facilitating and Recording Suggestions:

- Exercise patience: Goal is to get lots of ideas.

- Encourage non-speakers to speak.

- Use multiple colors when recording

- Echo the speaker's words.

- Record the speaker's words

- The rule of three 10's—don't cut off the brainstorm until there have been three 10 second (or longer) silent periods. Silent times during a brainstorm are useful— people are thinking.

- Silence is OK.

- Switch levels of analysis.

- Some references:

  - S. Kaner, Lind, Toldi, Fisk & Berger, *Facilitator's Guide to Participatory Decision-Making*

  - Freedman & Weinberg, *Handbook of Walkthroughs, Inspections & Technical Reviews*

  - Doyle & Straus, *How to Make Meetings Work*.

Exercise 2.3: Brainstorm Test Ideas (2/2)

Exercise 2.3: Brainstorm Test Ideas (2/2)

- ◆ A field can accept integer values between 20 and 50.
- ◆ What tests should you try?

18

Rati⊘nal
the software development company

(Intentionally blank slide)

19

Rati○nal
the software development company

A Test Ideas List for Integer-Input Tests

A Test Ideas List for Integer-Input Tests

♦ Common answers to the exercise would include:

Test	Why it's interesting	Expected result
20	Smallest valid value	Accepts it
19	Smallest -1	Reject, error msg
0	0 is always interesting	Reject, error msg
Blank	Empty field, what's it do?	Reject? Ignore?
49	Valid value	Accepts it
50	Largest valid value	Accepts it
51	Largest +1	Reject, error msg
-1	Negative number	Reject, error msg
4294967296	2^32, overflow integer?	Reject, error msg

20

Rational
the software development company

Discussion 2.4: Where Do Test Ideas Come From?

For each item that you target some testing against, it is useful to create a list of test ideas to be considered against that item. It's easier to maintain a "task list" of ideas for each item to be tested, rather than maintaining detailed documentation about each specific test.

What are some other good sources for test ideas lists?

- Bug lists (for example, *Testing Computer Software's* appendix, and *www.bugnet.com*)

- Business domain. For example, walk through the auction web site, the shopping cart, customer service app, and for each one, list a series of related ideas for testing.

- Technology framework: COM, .NET J2EE, …

- Fault models

- Representative exemplars (such as the "best" examples of devices to use for compatibility and configuration testing. *Testing Computer Software* illustrates this in its chapter on printer testing.)

A best example might not be the most popular or the most reliable. It is "best" representative of a class if testing it is *likely to yield more information than testing other members of the class*. So if a group of printers are allegedly compatible, but one has slightly weaker error handling, you might test with the weaker printer. If the program can pass testing with that one, it can pass with the rest.

A good example of this process in a business domain is the paper by Giri Vijayaraghavan & Cem Kaner, "Bugs in your shopping cart: A Taxonomy", in *Proceedings of 15th International Software Quality Week*, 2002.

Test Idea Catalogs

Module 2 Content Outline

- ◆ Definitions
  - ▪ Defining functional testing
  - ▪ Definitions of quality
  - ▪ A pragmatic definition of defect
  - ▪ Dimensions of quality
- ◆ Test ideas
- ➔ **Test idea catalogs**
  - ▪ **Test matrices**

Rational
the software development company

Now we will discuss how to apply the test ideas more generally.

Identify a Generic List of Test Ideas

Identify a Generic List of Test Ideas

- ◆ Think about the categories of values you'd consider generally applicable in an integer input field with Lower Bound (LB) and Upper Bound (UB).

Test	Why it's interesting	Expected result

"Why it is interesting" often means "what risk are we managing" or "what error we are looking for" or "what category or group of tests this test is an example of."

A Catalog of Test Ideas for Integer-Input tests

A Catalog of Test Ideas for Integer-Input tests

- Nothing
- Valid value
- At LB of value
- At UB of value
- At LB of value - 1
- At UB of value + 1
- Outside of LB of value
- Outside of UB of value
- 0
- Negative
- At LB number of digits or chars
- At UB number of digits or chars
- Empty field (clear the default value)
- Outside of UB number of digits or chars

- Non-digits
- Wrong data type (e.g. decimal into integer)
- Expressions
- Space
- Non-printing char (e.g., Ctrl+char)
- DOS filename reserved chars (e.g., "\ * . :")
- Upper ASCII (128-254)
- Upper case chars
- Lower case chars
- Modifiers (e.g., Ctrl, Alt, Shift-Ctrl, etc.)
- Function key (F2, F3, F4, etc.)

Rational®
the software development **company**

Would anyone really conduct tests for all of these ideas?

It depends on the perceived importance and risk of the feature, but in general – No. You can't run all of the interesting tests against all of the variables you would like to. There just isn't enough time.

A generic list of test ideas gives you a collection of good ideas that you can sample from: this is referred to as a test-ideas catalog. A good catalog helps you manage the infinite number of possible tests.

For example, you might not test every integer variable with every member of this ideas catalog, but you might make sure that you test every variable, and that any specific member of the catalog is tested against at least a few variables. If you find an error, you might base more tests on the idea that led you to the error.

The Test-Ideas Catalog

A test-idea catalog is a collection of test ideas useful in addressing a particular test objective (e.g. boundary test cases for numeric fields). A test project can make use of many catalogs based on test scope.

Test-idea catalogs are good reminders for experienced staff, because they capture thinking that doesn't need to be redone time and again.

They are especially handy when adding a new tester near the end of the project. It's best to hire (or contract with) experienced testers for end-of-project work, but even if they are experienced in general, they still won't know your product or how best to test it. In either case, the challenge of late additions to staff is that you don't have time to train them and you need their productivity immediately.

Test-idea catalogs can help a new person gain productivity quickly. They act as "training wheels".

It's worth repeating:

- Test ideas are not test cases: they are the ideas from which you derive test cases

- Test ideas are not test techniques: they provide ideas from which you decide which techniques to apply

- Although you might consider all the test ideas in the catalog each time you use it, you usually won't conduct a test for every idea in the catalog each time you test.

Apply a Test Ideas Catalog Using a Test Matrix

	Lower Bound	LB-1	Upper Bound	UB+1	Zero	Spaces	Nothing	Negative
Field name								
Field name								
Field name								

A test matrix is a useful way of working with the ideas from a catalog.

Imagine looking through the test ideas in the catalog, selecting a subset that you think will be useful in the context of your current work. Next, imagine walking through all of the dialog boxes in your application and filling in the field names of all the numeric input fields. Now test each field in turn, checking each test idea in the matrix off as you conduct an associated test. Perhaps you would highlight cases that pass by coloring the cells green (use a green highlighter if you're doing this on a printout of the matrix) and highlight the failing cases pink.

A matrix like this is easy to delegate to a new member of the testing team.

Many groups bring experienced testers into the project late in development to cope with schedule slippage. The new testers know how to test, but they don't know what to do on *this* project. They can figure out the *how* (what steps to take to conduct a test) but don't yet know which tests to run or why. A matrix or catalog can be very useful for these testers, helping them to get up to speed quickly.

Many groups find matrices like these much more useful, *for dealing with tests that are routine and well understood* than full test case descriptions.

Many test managers find matrices like this a good assessment tool for evaluating the test work being performed.

Exercise 2.5: Your Own Test Ideas Lists

Exercise 2.5: Your Own Test Ideas Lists

- ◆ Now (in class)
  - ▪ Pick a topic of interest for a test ideas list
- ◆ Homework
  - ▪ Expand into test ideas list
  - ▪ Tomorrow we will discuss this
- ◆ Homework follow-up in class
  - ▪ Develop a matrix from these ideas

Rational®
the software development company

Optional Exercise:

Think about candidates in your organization for a test ideas list. What makes the test ideas list good is that there are many generalizations.

As homework, write out the ideas you have.

Depending on time tomorrow, it may be worth collecting and consolidating these ideas lists, then brainstorming more. After the brainstorm, a second level of homework would be to turn the lists into matrices.

Review: Core Concepts of Software Testing

Review: Core Concepts of Software Testing

- ◆ What is Quality?
- ◆ Who are the Stakeholders?
- ◆ What is a Defect?
- ◆ What are Dimensions of Quality?
- ◆ What are Test Ideas?
- ◆ Where are Test Ideas useful?
- ◆ Give some examples of a Test Ideas.
- ◆ Explain how a catalog of Test Ideas could be applied to a Test Matrix.

28

Rational
the software development company