

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

***See
accompanying
Word Doc for
detailed
instructor notes.***



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

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

The highlighted terms are the key terms and concepts introduced in this module.

Objectives

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

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

2

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

⌚ Transition Slide. Don't spend any time here.

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

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

3

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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.

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

4

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Use this exercise to discover the students' working definition of quality.

Exercise guidelines

- Count off to form pairs
- Ask every group to do the exercise
- Give them 10 minutes or so
- Collect answers and write on the flipchart

• Then transition to next slide by saying,
"Now we'll look at how some experts have defined 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?

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

5

Rational
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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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

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

7

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..."

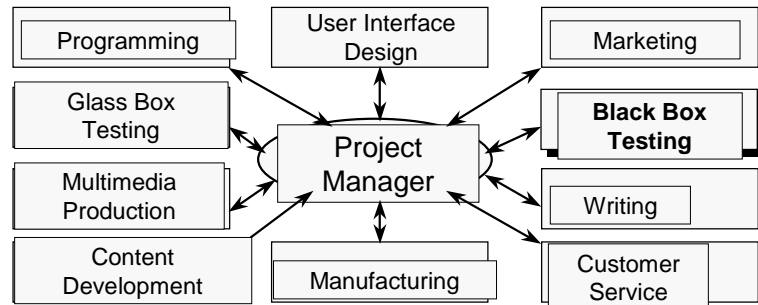
For Juran's discussion of these terms, see:

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

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Quality Involves Many Stakeholders



- ♦ 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.

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

8

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

It's important to keep this debate going long enough that key points get made on both sides.

Testers who argue **in favor** of allowing this report in the system might argue that:

- any stakeholder should be able to file a report, including the programmer.
- even though this problem is not customer-visible today, it will affect maintainability of the product and therefore may lead to errors in the future. The product has lower quality (lower maintainability) because of poor naming and therefore is appropriate to report as defective.

- this is a deviation from a corporate standard and is therefore a bug.

Testers who argue **against** allowing this report might argue that:

- this report is based on a personal vendetta and doesn't belong in the database. (Actually, this report is NOT based on a personal vendetta. I suggest correcting the student who raises this.)
- only the testers should file bug reports.
- the programmers should settle this among themselves without using the formal bug tracking system.
- the problem is not visible to the customer and therefore doesn't belong in the bug tracking system.

Our recommended answer is that the problem belongs in the bug tracking system because a stakeholder wants it there. The tester's responsibility is to help that stakeholder make the report effective (accurate, sufficient detail, persuasive).

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?

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved9**Rational**
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Few companies that release software in multiple languages test it equally well in all languages. For example, suppose that the English version is the "base code" (the version that will be translated) -- the company might spend several months testing this, but only several days or a couple of weeks testing the German (French, Italian, Spanish, etc.) version. If the company changes the code of a translated version, then relinks and recompiles the software, error can come in at the programmer level (wrong code) or as a result of differences in the German compiler or other German tools.

In practice, many companies have released an English version, translated the software by changing the code, and then released it with relatively brief testing. The result has commonly been a translated version that is much less reliable than the English version.

The norm now is to create a base code version that will not have to be recompiled or relinked. Instead, the localizer (person who translates the software and adapts it to another culture) works with a resource file--a data file that contains every variable that must be changed during localization. The program reads the values in the file at run-time. If the base code is well-tested and reasonably reliable, the localized version can be released with much less testing than was needed for the base code.

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?

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved10**Rational**
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

A Working Definition of Quality

Quality is value to some person.

---- Gerald M. Weinberg

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

11

Rational
the software development company

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

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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.

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

12

Rational
the software development company

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.

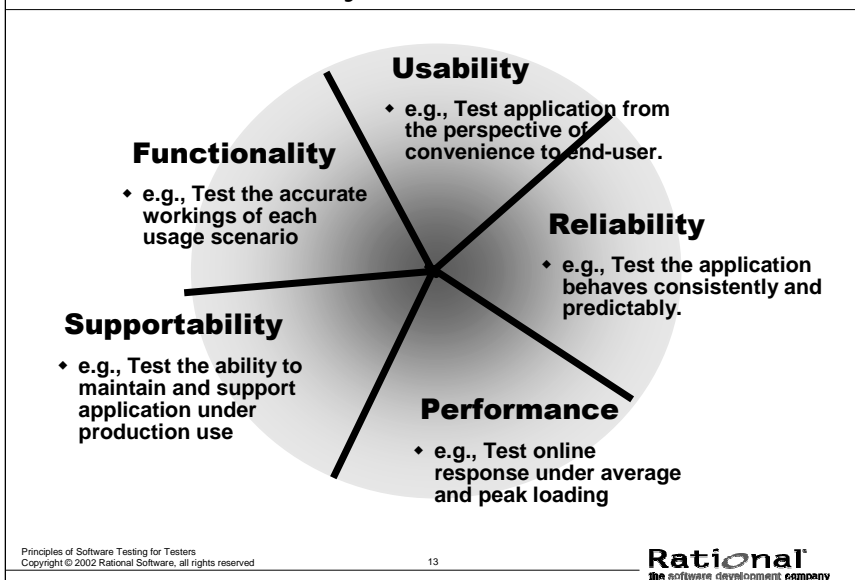
Some companies set up a second change request database. Testers report late-found low-priority defects into this database. Other companies tag defects for future releases. The project manager or a senior programmer skims these reports for issues that raise red flags. These requests are re-opened and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

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

- Consider the FURPS model a generator of ideas and summary of categories. There are really many more dimensions of the problem to consider as we see on the next slide.

Dimensions of Quality: FURPS



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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Note these are dimensions can be thought of as qualities of service you wish your application to provide. They are **not** test techniques. Test techniques tend to evaluate several of these at a time.

The value of these dimensions is to help you ask the question: Do I need to check the quality of service of application X along this dimension?

Soon we'll talk about the Test Ideas List as a way of capturing the drill down questions for the dimensions.

=====

Notes on some of the dimensions:

Accessibility refers to the usability of the program for someone with a handicap. For example, a program could be accessible to the colorblind or to the deaf or to someone who cannot control a mouse.

Supportability refers to the extent to which the program can be supported, e.g. by a tech support representative. A more supportable program might have diagnostics for troubleshooting, or informative error messages.

Maintainability refers to the extent to which a program can be safely and easily modified.

A Broader Definition of Dimensions of Quality

- | | |
|---------------------------------------|-------------------|
| ♦ Accessibility | ♦ Maintainability |
| ♦ Capability | ♦ Performance |
| ♦ Compatibility | ♦ Portability |
| ♦ Concurrency | ♦ Reliability |
| ♦ Conformance to standards | ♦ Scalability |
| ♦ Efficiency | ♦ Security |
| ♦ Installability and uninstallability | ♦ Supportability |
| ♦ Localizability | ♦ Testability |
| | ♦ Usability |

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

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

14

Rational
the software development company

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

⌚ Transition Slide. Don't spend any time here.

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

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

15

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

There are three concepts to convey in this section:

- A **Test Idea**—A single test idea as described in the slide text.
- A **Test-Ideas List**—A list of test ideas applicable to a specific target of testing.
- A **Test-Ideas Catalog**—A generalized collection of test ideas that can be applied to other testing targets in a similar context.

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.

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

16

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Exercise 2.3: Brainstorm Test Ideas (1/2)

- ◆ We're about to brainstorm, so let's review...
- ◆ Ground Rules for Brainstorming
 - The goal is to get lots of ideas. You brainstorm together to discover categories of possible tests—good ideas that you can refine later.
 - There are more great ideas out there than you think.
 - Don't criticize others' contributions.
 - Jokes are OK, and are often valuable.
 - Work later, alone or in a much smaller group, to eliminate redundancy, cut bad ideas, and refine and optimize the specific tests.
 - Often, these meetings have a facilitator (who runs the meeting) and a recorder (who writes good stuff onto flipcharts). These two keep their opinions to themselves.

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

17

Rational
the software development company

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*.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Do this as a group brainstorm exercise, according to the brainstorming rules covered on the previous slide.

Exercise 2.3: Brainstorm Test Ideas (2/2)

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

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

18

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

This slide keeps participants from looking ahead.

(Intentionally blank slide)

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

19

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Here is an answer key.

(In a minute, we'll generalize the exercise.)

Murphy's Law of Zero: If you can enter a zero into a field, someone can divide by it.

Test for overflows on all fields. A large percentage of security problems involve buffer overflows.

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

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

20

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Now let's step back.

In the RUP, the Test Ideas List is input to the activities:

- Implement Test
- Define Test Details
- Develop Test Guidelines
- Determine Test Results

Discussion 2.4: Where Do Test Ideas Come From?

- ◆ Where would you derive Test Ideas Lists?
 - Models
 - Specifications
 - Customer complaints
 - Brainstorm sessions among colleagues
- ◆ How do you create them in your company?

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

21

Rational
the software development company

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

⌚ Transition Slide. Don't spend any time here.

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**

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


22

Rational
the software development company

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

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

 **Brainstorm.** Lead the students in a discussion of how to generalize what they discovered.

This is a good task for working groups to do together in an actual business environment. Groups define their lists for various types of fields, as well as for many different types of functions or features.

Note: brainstorming sessions don't create final-version lists. It is an idea generator. You're looking for ideas that are good and can be made better.

After the brainstorming, a person can be assigned a specific list to refine, such as a list of test ideas for a given type of field. That person should take the rough notes from the brainstorming session, delete the ones that aren't useful, make more powerful versions of several of the other test ideas, organize them sensibly, and probably add a few more 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

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

23

Rational
the software development company

"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."

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

- As you refine the brainstorm list, look for variants on each test that might yield more powerful cases.
- For example, if a student suggests alpha characters (not numeric), point out that numbers are received (inputs) as ASCII characters 48-57 (Decimal). So, the characters whose codes are Decimal 47 ("/") and 58 (";") are interesting boundary cases.
- Do we use all of these test ideas all the time everywhere? Of course not. How much do you need to do to "check off the idea"? When are we comfortable with having done enough?
- It depends. For example, if the programmers do extensive unit testing in your company already, you might do these simple boundary tests very lightly, saving your time for complex scenarios.

A Catalog of Test Ideas for Integer-Input tests

- | | |
|---|---|
| • Nothing | • Non-digits |
| • Valid value | • Wrong data type (e.g. decimal into integer) |
| • At LB of value | • Expressions |
| • At UB of value | • Space |
| • At LB of value - 1 | • Non-printing char (e.g., Ctrl+char) |
| • At UB of value + 1 | • DOS filename reserved chars (e.g., "\ * . :") |
| • Outside of LB of value | • Upper ASCII (128-254) |
| • Outside of UB of value | • Upper case chars |
| • 0 | • Lower case chars |
| • Negative | • Modifiers (e.g., Ctrl, Alt, Shift-Ctrl, etc.) |
| • At LB number of digits or chars | • Function key (F2, F3, F4, etc.) |
| • At UB number of digits or chars | |
| • Empty field (clear the default value) | |
| • Outside of UB number of digits or chars | |

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

24

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Brainstorm exercise (part 1):

Where else would you want to have a test-idea catalog? Where would it make sense to develop (harvest) one in your company?

The Test-Ideas Catalog

- ♦ A test-ideas catalog is a list of related test ideas that are usable under many circumstances.
 - For example, the test ideas for numeric input fields can be catalogued together and used for any numeric input field.
- ♦ In many situations, these catalogs are sufficient test documentation. That is, an experienced tester can often proceed with testing directly from these without creating documented test cases.

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

25

Rational
the software development company

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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								

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

26

Rational
the software development company

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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Brainstorm exercise part 2:

Pick a topic of interest to the room and expand into test ideas list and matrix of ideas. Doesn't need to be input fields (probably shouldn't be.)

If the students don't have ideas, suggest installation testing as a technical example; internet stock trading as a business domain example.

NOTE: This exercise is most successful if you suggest a topic one day, encourage people to spend 15 minutes generating test ideas on their own overnight, and do the actual brainstorming session the next morning.

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

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved27**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.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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.

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

28

Rational
the software development company