

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

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

We just looked at test ideas;
now we're going to the
other end of the
management scale:

How do we decide what we
are going to do in the
project? In the iteration?

Our focus in this module
will cover:

- ♦ Test plans: What level of detail is appropriate?
- IEEE Standard 829-type templates
- The issues of time, cost, flexibility and maintainability
- Requirements for test docs
- Formulating a mission statement for test docs.



Principles of Software Testing for Testers Module 4: Define Evaluation Mission

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

There is an infinite number of tasks that we could do as testers. We know not to enter blindly. But how do we agree with our stakeholders on what's important?

That's the role of the test mission – to negotiate stakeholder buy-in for the test plan and to focus our priorities as testers.

A test mission is critical if you ever hope to get through the decision process for selecting what work **MUST** be done to insure this project meets quality goals (given finite time and resources).

Module 4 Agenda

- ♦ **Definition of the workflow:**
Define Evaluation Mission
- ♦ Defining the mission of the test group
- ♦ Defining the goal for test documentation

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:

⌚ Context Slide. Avoid spending too much time here: spend enough time to help the students take stock of what they have learned so far. Answer any related questions they have, then move on.

Main points to review:

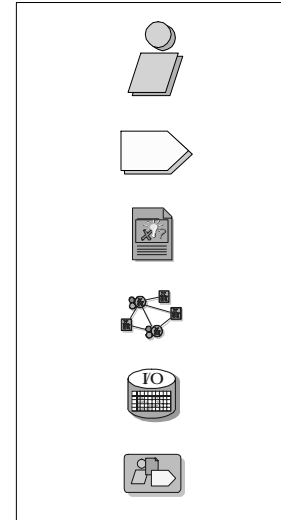
- Roles, Activities and Artifacts.
- Workflow Details.
- Iterations.

Optionally, you might also recap:

- Quality
- Test Ideas

Review: Where We've Been

- ♦ In the introductory module we discussed the concepts of *quality* and *test ideas*
- ♦ In the last module, we introduced some of the basic elements in the RUP Test content
- ♦ We'll use those basic RUP elements throughout the remainder of the course to help provide context for what we'll learn.



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:

⌚ Context Slide. Avoid spending too much time here: spend enough time to give the students a high-level understanding. Answer any related questions they have, then move on.

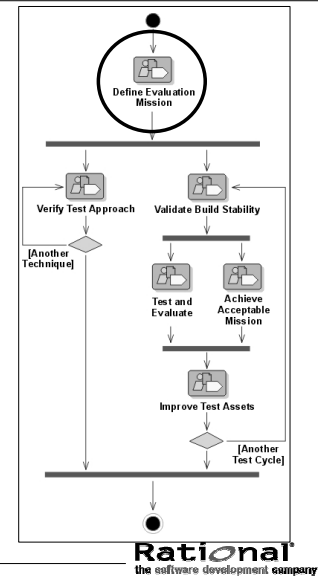
The main point is to briefly explain the context and scope of this module relative to the rest of the course.

Explain that the Test and Evaluate Workflow Detail will be delivered in two parts: the first focusing on Test Techniques, the second on Evaluating the results of the Tests.

Explain briefly that Test & Evaluate is the core RUP Workflow Detail for the Software Tester. Relate it back to the Define Evaluation Mission Workflow Detail in which Mission and Test Approach were discussed.

Define Evaluation Mission

- ◆ In this module, we begin with the workflow detail Define Evaluation Mission
- ◆ The Mission focuses on the high-level objectives of the test team for the current iteration
 - What things should motivate us to test?
 - Why these things (and not others)?



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

4

The purpose of this workflow detail is to:

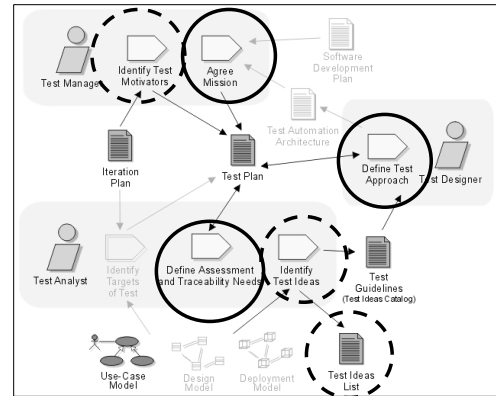
- Identify the appropriate focus of the test effort for the iteration.
- Gain agreement with stakeholders on the corresponding goals that will direct the test effort
- For each iteration, work is focused mainly on:
 - Identifying the objectives for, and deliverables of, the testing effort
 - Identifying a good resource utilization strategy
 - Defining the appropriate scope and boundary for the test effort
 - Outlining the approach that will be used
 - Defining how progress will be monitored and assessed.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Define Evaluation Mission

- ◆ This module focuses on the activities that capture the goals of our testing efforts.
- ◆ We will look at different *Missions* that test teams use, and consider the implications on the corresponding *Test Approach* those teams take.
- ◆ These are the activities that create the *Test Plan*.



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

5

Rational
the software development company

Here are the roles, activities and artifacts RUP focuses on in this work.

In earlier modules, we discussed how identifying test ideas is a useful way to reason about tests early in the lifecycle without needing to completely define each specific test. We also looked at some of the basic elements that are used to define the Rational Unified Process.

In this module we'll look at how different test teams need to use different evaluation missions, depending on their specific context.

In the next module, we'll talk more about applying different techniques in our tests effort.

Note that diagram shows some light shaded elements: these are additional testing elements that RUP provides guidance for which not covered directly in this course. You can find out more about these elements by consulting RUP directly.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

There are an infinite number of tasks that we could do as testers. We know not to enter blindly. But how do we agree with our stakeholders on what's important?

That's the role of the test mission – to negotiate stakeholder buy-in for the test plan and to focus our priorities as testers.

A test mission is critical if you ever hope to get through the decision process for selecting what work **MUST** be done to insure this project meets quality goals (given finite time and resources).

Module 4 Agenda

- ♦ Definition of the workflow:
Define Evaluation Mission
- ♦ **Defining the mission of the test group**
- ♦ Defining the goal for test documentation

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved6**Rational**
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

This example is artificial, but it sets up a simple context for our discussion of tradeoffs.

The goal here is to facilitate a discussion that brings out the students' differing assumptions.

Ask the students to vote on which is the better group and have students from each side explain their thinking.

Play devil's advocate, a little bit, to help the discussion crystallize.

Exercise 4.1: Which Group is Better?

Testing Group 1

	Found pre-release
Function A	100
Function B	0
Function C	0
Function D	0
Function E	0
Total	100

Testing Group 2

Function A	50
Function B	6
Function C	6
Function D	6
Function E	6
Total	74

From Marick,
Classic Testing Mistakes

Two groups test the same program.

- The functions are equally important
- The bugs are equally significant

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

7

Rational
the software development company

Imagine giving the same product to two completely independent test groups. The product has lots of functions, but you pick out five that you consider the most important:

- These 5 functions are equally important.
- You expect that on average, the bugs found in any one of the functions will be as serious or significant as bugs found in the others.

Test Group 1 starts with a broad test, trying all the functions with easy values. No obvious bugs show up in Functions B, C, D or E, but Function A is clearly broken from the start.

Over the next few weeks, Test Group 1 keeps testing the program. Their goal is to find lots of bugs. They spend a little more time on Functions B, C, D and E, but primarily focus on Function A, where it is all too easy to find bugs. Test Group 1 never finds any bug in Function B, C, D, or E, but they find 100 in Function A.

Test Group 2 starts with essentially the same broad test and finds the same result. Function A appears to be full of bugs. As to B, C, D, and E, no obvious bugs show up in the first round of testing.

Test Group 2 follows a different testing strategy. They do a lot more testing of B, C, D, and E – even though they don't find as many bugs in these functions, because they want to achieve a certain baseline level of coverage. As a result, they find only 50 bugs in A and a few bugs in B, C, D, and E.

WHICH IS THE BETTER TESTING GROUP?

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

This is just a blank page, here to prevent them from accidentally seeing the next slide before you want them to see it.

Notes

This slide was intentionally left blank.

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

8

Rational
the software development company

Your Notes:

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Get the discussion to continue. Alternate between students who think that Group 1 is better and students who think that Group 2 is better.

The numbers found are the same as on the last slide. The difference is that we extrapolated to post-sale results. Now people see the effect of the testing prioritization. One group achieved broader testing and found 50% of all the bugs, spread across all modules. The other group achieved more intense testing of the highest risk area and found more bugs but unevenly spread.

If the goal is to minimize tech support cost, Group 1 is probably doing the better testing job.

If the goal is to evaluate the quality of the overall product, Group 1 is doing a weak job because we have sketchy information about the other 4 functions.

In practice, some students will switch groups after seeing this slide and your explanation of it.

Exercise 4.2: Which Group is Better?

	Found pre-release	Found later	Total
Function A	100	0	100
Function B	0	12	12
Function C	0	12	12
Function D	0	12	12
Function E	0	12	12
Total	100	48	148
Function A	50	50	100
Function B	6	6	12
Function C	6	6	12
Function D	6	6	12
Function E	6	6	12
Total	74	74	148

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

9

Rational
the software development company

Here's some more data.

The company shipped the product. Six months later, we look at customer support call data and we see a bunch of new bugs found by customers.

- The first group found all the bugs in Function A but missed 48 bugs in B,C, D and E.
- The second group found half the bugs of each function.
- The first group found 100 of the 148 bugs
- The second group found 74 of the 148 bugs, but they were more evenly distributed across functions.

WHICH GROUP IS BETTER? WHY?

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

How good a test group is depends on how well the group satisfies its mission.

How good its mission is depends on the test group's interaction with the rest of the organization.

- Group 1 is a failing group if the organization's goal for testing is assessment
- Group 2 is failing if the organization's goal is maximize bug count.
- The mission as seen by the test group must match the mission as seen outside of the test group, or whatever the test group achieves will be perceived by others as inadequate.

James Bach uses an interesting metaphor here (ref. *Lessons Learned* in Software Testing, page 1), when he calls the test team the "headlights of the project".

Of the two sensible answers to the mission that we've considered so far, which one fits you?

Note that this is closely tied to your definition of quality. Are you looking for:

- Conformance to specs, **or**
- Nonconformance to user expectation?

Discussion point:

- What is your mission?

So? Purpose of Testing?

- ♦ The typical testing group has two key priorities.
 - Find the bugs (preferably in priority order).
 - Assess the condition of the whole product (as a user will see it).
- ♦ Sometimes, these conflict
 - *The mission of assessment is the underlying reason for testing, from management's viewpoint. But if you aren't hammering hard on the program, you can miss key risks.*

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:

In addition to thinking about your own mission, think about other companies.

Where would you guess Boeing's mission is? How about Microsoft's?

Is the test activity creative or investigative?

How independent do you think a test group should be?

What happens if you do not negotiate the mission?

Missions of Test Groups Can Vary

- ♦ Find defects
- ♦ Maximize bug count
- ♦ Block premature product releases
- ♦ Help managers make ship / no-ship decisions
- ♦ Assess quality
- ♦ Minimize technical support costs
- ♦ Conform to regulations
- ♦ Minimize safety-related lawsuit risk
- ♦ Assess conformance to specification
- ♦ Find safe scenarios for use of the product (find ways to get it to work, in spite of the bugs)
- ♦ Verify correctness of the product
- ♦ Assure quality

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved11**Rational**
the software development company

Even if the test group has an overall mission, its objectives will vary over the life of the project. For example, a group whose primary role was defect-hunting through most of the project might be expected to provide quality evaluations as the project gets closer to its planned release date.

It is important for the test group to decide its guiding objectives for each iteration, and to reassess these as part of the preparation for each iteration.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Exercise guidelines

Possible exercise/discussion:
What would be an appropriate mission for testing a PBX? A computer game?

If you do this as an exercise, split the class into project teams. Teams should include the test manager, product development manager, marketing manager, and the company's lawyer. For larger teams, add the tech support manager and the sales manager. If this is an in-house development group, consider replacing the marketing manager with the customer representative.

Have each group tackle the same type of product and negotiate their mission. At the end of the exercise, compare missions across groups and get to the reasoning underlying the differences in choices.

- Then transition to the next slide by saying,
Now let's look at mission a little differently. How many of the product's total bugs do you think the test team is finding?

Exercise 4.3: What Is Your Mission?

- ♦ Pick a company and a product
 - Probably your own
 - If you don't want to use your current one, pick one everyone knows
- ♦ Form project teams
- ♦ What's the test mission?

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

12

Rational
the software development company

Optional Exercise

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Test activity does not happen independent of a development process. Take a look at your development process. How much unit testing do your developers do and how is its coverage tracked?
Do testers know what the devs are already doing? Do testers know the risks by dev asset? Do they know whether boundary testing is being done in devt?

Thinking about public vs. private bug rates takes a very different look at the problem.

A Different Take on Mission: Public vs. Private Bugs

- ♦ A programmer's public bug rate includes all bugs left in the code at check-in.
- ♦ A programmer's private bug rate includes all the bugs that are produced, including the ones fixed before check-in.
- ♦ Estimates of private bug rates have ranged from 15 to 150 bugs per 100 statements.
- ♦ What does this tell us about our task?

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved13**Rational**
the software development company

A programmer's public bug rate includes all bugs left in the code when it is given to someone else (such as a tester.) Rates of one bug per hundred statements are not unusual, and several programmers' rates are higher (such as three bugs per hundred).

A programmer's private bug rate includes all the bugs that are produced, including the ones already fixed before passing the program to testing.

Estimates of private bug rates have ranged from 15 to 150 bugs per 100 statements. Therefore, programmers must be finding and fixing between 80% and 99.3% of their own bugs before their code goes into test. (Even the sloppy ones find and fix a lot of their own bugs.)

What does this tell us about our task?

It says that we're looking into the programmer's (and tools') blind spots. Merely repeating the types of tests that the programmers did won't yield more bugs. That's one of the reasons that an alternative approach is so valuable.

Conclusion: Unless the tester's methods are different from the programmer's, the tester will be going over already well tested grounds.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

The Mission guides your testing objective in the iteration. Now we're going to think about the **How** question: The Test Approach.

Defining the Test Approach

- ◆ The test approach (or “testing strategy”) specifies the techniques that will be used to accomplish the test mission.
- ◆ The test approach also specifies how the techniques will be used.
- ◆ A good test approach is:
 - Diversified
 - Risk-focused
 - Product-specific
 - Practical
 - Defensible

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved14**Rational**
the software development company

Diversified. Include a variety of techniques. Each technique is tailored to expose certain types of problems, and is virtually blind to others. Combining them allows you to find problems that would be hard to find if you spent the same resource on a narrower collection of techniques.

Risk-focused. Tests give you the opportunity to find defects or attributes of the software that will disappoint, alienate, or harm a stakeholder. You can't run all possible tests. To be efficient, you should think about the types of problems that are plausibly in this product or that would make a difference if they were in this product, and make sure that you test for them.

Product-specific. Generic test approaches don't work. Your needs and resources will vary across products. The risks vary across products. Therefore the balance of investment in different techniques should vary across products.

Practical. There's no point defining an approach that is beyond your project's capabilities (including time, budget, equipment, and staff skills). For example, you won't be likely to succeed if you try to build a fully automated test plan if you have a team full of non-programmers.

Defensible. Can you explain and justify the work that you are doing? Does your approach allow you to track and report progress and effectiveness? If you can't report or justify your work, are you likely to be funded as well as you need?

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

It would be helpful to crack the book, look at the table pages (257-259), and talk about some of the heuristics.

For example, Bach suggests that you maximize diversity in your testing.

- Is this always practical?
- Do you risk trading off depth against breadth?
- Has anyone in the class worked on a project in which significant bugs were missed that might have been more easily found by a technique that wasn't used?

Heuristics for Evaluating Testing Approach

- ♦ James Bach collected a series of heuristics for evaluating your test approach. For example, he says:
 - Testing should be optimized to find important problems fast, rather than attempting to find all problems with equal urgency.
- ♦ Please note that these are heuristics – they won't always be the best choice for your context. But in different contexts, you'll find different ones very useful.

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

15

Rational
the software development company

A heuristic is a rule of thumb, a rule that is useful but not always correct.

Example of a heuristic

The test approach should focus most effort on areas of potential technical risk, while still putting some effort into low risk areas just in case the risk analysis is wrong.

Basis for this heuristic

Complete testing is impossible, so we have to select the tests to run. Ideally, we would run the tests that promise to provide the most useful information. However, no risk analysis is perfect. We have to put some effort into checking out the areas that appear to be low risk, just in case.

The full list is included as a table in the course textbook, *Lessons Learned in Software Testing*, pages 257-259.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

There are an infinite number of tasks that we could do as testers. We know not to enter blindly. But how do we agree with our stakeholders on what's important?

That's the role of the test mission – to negotiate stakeholder buy-in for the test plan and to focus our priorities as testers.

A test mission is critical if you ever hope to get through the decision process for selecting what work **MUST** be done to insure this project meets quality goals (given finite time and resources).

Module 4 Agenda

- ♦ Definition of the workflow:
Define Evaluation Mission
- ♦ Defining the mission of the test group
- ♦ **Defining the goal for test documentation**

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved16**Rational**
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

This is an overview slide. We'll cover the topics listed here in the next slides.

The underlying topic can be summarized as follows:

- What test documentation makes sense for your project? Don't start with "How?" or "What should it look like?", but start with, "***What is it supposed to do?***"

What Test Documentation Should You Use?

- ♦ Test planning standards and templates
 - Examples
 - Some benefits and costs of using IEEE-829 standard based templates
 - When are these appropriate?
- ♦ Thinking about *your* requirements for test documentation
 - Requirements considerations
 - Questions to elicit information about test documentation requirements for your project

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

17

Rational
the software development company

IEEE Standard 829 for software test documentation is a standard initially published by the Institute for Electrical and Electronics Engineers (1983) and later approved by the American National Standards Institute.

The standard describes a wide range of types of information that can be included in test documentation. For examples, see the next slide.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

No IEEE standard 829 standard test plan is required to have all this information, but the typical 829 template calls for it. Test cases are often described, in practice, in great detail.

Why spend so much time on IEEE 829? Most test planning templates, including the example template given in RUP, are based on 829. Whether the student has heard the 829 name or not, most of the discussions he or she has read or heard about regarding what a test plan should contain, trace back to standard 829.

IEEE Standard 829 for Software Test Documentation

- ◆ Test plan
- ◆ Test-design specification
- ◆ Test-case specification
 - Test-case specification identifier
 - Test items
 - Input specifications
 - Output specifications
 - Environmental needs
 - Special procedural requirements
 - Intercase dependencies
- ◆ Test-procedure specification
- ◆ Test-item transmittal report
- ◆ Test-log

*We often see
one or more
pages per
test case.*

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

18

Rational
the software development company

In the course text, *Lessons Learned in Software Testing*, there are two conflicting lessons:

- *Lesson 145: Use the IEEE Standard 829 for test documentation.*
- *Lesson 146: Don't use the IEEE Standard 829.*

These two lessons contrast circumstances under which the standard is appropriate and under which it is not.

A critical point to recognize here is that test documentation is not free and can be very expensive.

It is common to see a full page of documentation for a simple test case and many pages for complex test cases. It probably takes 1 to 8 hours to write a page of test documentation (Technical writers take about 8 hours per page on software user manuals, when you include the research, writing formatting and editing time. For more on the productivity of tech writers, see JoAnn Hackos, *Managing Your Documentation Projects*, Wiley.)

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

The cost of testing documentation should not determine the project strategy. This will only create inertia – we can't afford to do X important tasks because it costs too much to change the document.

Sometimes 829 is valuable:

- Evidence against future liability
- Contractual fulfillment
- Multisite coordination
- Safety-critical projects
- Regulatory requirements

For more ideas, see *Lessons*, pages 129-136.

=====

"Invariant regression testing?" refers to repetition of exactly the same tests time after time. This is sometimes the right thing to do (e.g. build verification testing) but sometimes building variation into testing is more valuable than building in exact repeatability. The problem of "inertia" is this: If you fully document every test, and face a maintenance cost to the test documentation every time you change a test, you are less likely to vary tests – not because variation is technically undesirable, but because the documentation process makes it expensive.

Considerations for IEEE 829

- ♦ What is the documentation cost per test case?
- ♦ What is the maintenance cost of the documentation, per test case?
- ♦ If software design changes create documentation maintenance costs, how much inertia do we build into our system? How much does extensive test documentation add to the cost of late improvement of the software? How much should we add?
- ♦ What inertia is created in favor of invariant regression testing?
- ♦ Is this incompatible with exploratory testing? Do we always want to discourage exploration?

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

19

Rational
the software development company

Some other questions to consider are:

- What is the impact on high-volume test automation? If the documentation cost per test case is high, how can you afford to create a multi-million test case project?
- How often do project teams start to follow 829 but then give it up mid-project? What does this do to the net quality of the test documentation and test planning effort?
- What requirements are filled by following a template based on 829?
- Which stakeholders gain a net benefit from IEEE standard documentation?
- What benefits do they gain and why are those benefits important to them?

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Consider your project plan as a whole:

- Frequency of iterations
- Ability to manage change on requirements, design model and test plan
- Extent of parallel development

Requirements for Test Documentation

- There are many different notions of what a good set of test documentation would include. Before spending a substantial amount of time and resources, it's worth asking what documentation should be developed (and why?)
- *Test documentation is expensive and it takes a long time to produce. If you figure out some of your main requirements first, you might be able to do your work in a way that achieves them.*

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

20

Rational
the software development company

Lessons Learned in Software Testing provides 18 questions (page 136-140) that you can use to guide your analysis of your test documentation requirements.

We'll consider one example on the next slide.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

This discussion is most relevant for those testing groups who think about specific types of tasks or artifacts.

In the RUP, we typically discuss the testing approach in terms of mitigation of risk. However, as we'll see later, this is not the only approach to thinking about testing.

When groups think in terms of test documentation artifacts, it is important to drill into the real requirements of those artifacts.

Test Docs Requirements Questions

♦ **Is your test documentation a *product* or a *tool*?**

- A product is something that you give to someone else to use. They pay for it. You will probably follow whatever standard they request, subject to their willingness to pay for it.
- In contrast, if the documentation is merely an in-house tool, it doesn't have to be any more complete, more organized, or more tidy than the minimum you need to help you meet your objectives.

Principles of Software Testing for Testers
Copyright © 2002 Rational Software, all rights reserved21**Rational**
the software development company

Here are some additional examples:

- Is software quality driven by legal issues or by market forces?
- How quickly is the design changing?
- How quickly does the design specification change to reflect design change?
- Is testing approach oriented toward proving conformance to specs or nonconformance with customer expectations?
- Does your testing style rely more on already-defined tests or on exploration?
- Should test docs focus on what to test (objectives) or on how to test for it (procedures)?
- Should the docs ever control the testing project?
- If the documentation controls parts of the testing project, should that control come early or late in the project?
- Who are the primary readers of these test documents and how important are they?
- How much traceability do you need? What documents (specifications or requirements) are you tracing back to and who controls them?

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

The purpose statement for test documentation should follow from the test mission statement.

Get explicit stakeholder agreement on the purpose and nature of documentation you are expected to produce.

Write a Purpose Statement for Test Documentation

- ◆ Try to describe your core documentation requirements in *one sentence* that doesn't have more than three components.
- ◆ Examples:
 - The test documentation set will primarily support our efforts to find bugs in this version, to delegate work, and to track status.
 - The test documentation set will support ongoing product and test maintenance over at least 10 years, will provide training material for new group members, and will create archives suitable for regulatory or litigation use.

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

22

Rational
the software development company

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Exercise 4.4: Purpose for Your Test Documentation?

- ♦ Use the company and product from Ex. 4.3
- ♦ Reform project teams
- ♦ What's the test *documentation* goal?

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

23

Rational
the software development company

Optional Exercise

Regroup into your project teams and take ten minutes to discuss the exercise. Write down the answer, so you can share it with the group.

Principles of Software Testing for Testers Instructor Notes

Instructor Notes:

Review: Define Evaluation Mission

- ♦ What is a Test Mission?
- ♦ What is your Test Mission?
- ♦ What makes a good Test Approach (Test Strategy)?
- ♦ What is a Test Documentation Mission?
- ♦ What is your Test Documentation Goal?

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

24

Rational
the software development company