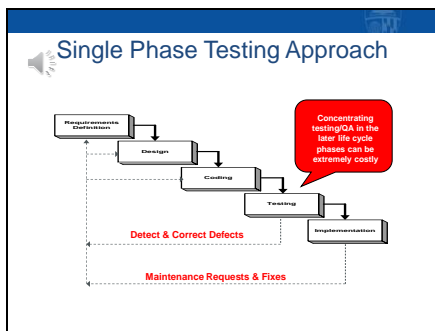


1



In this lecture, I'm going to discuss testing activities and testing work products throughout the project life cycle.

2



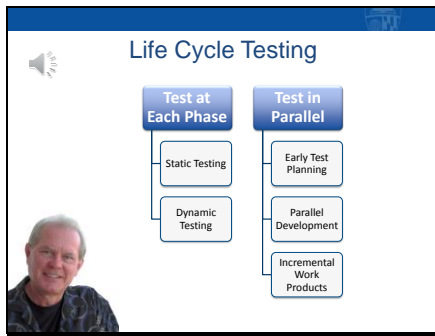
Before focusing on tester activities and work products, I want to say a few words about what I refer to as a single phase approach to testing versus a life cycle phase approach.

In most software project life cycles, there is a phase devoted to testing. The testing phase usually occurs later in the life cycle as illustrated here. For many projects, most of the testing activities may be performed in this single phase or step...and that can be problematic for reasons we've talked about earlier in the course, regarding error amplification and relative costs to find and fix problems. In this diagram, the dashed arrows represent typical feedback and error correction flows. As you can see, the first feedback usually occurs during the testing phase...when test failures begin to indicate problems.

If a waterfall life cycle model is used, defect detection and repair costs can be significant. If an incremental life cycle model is used, defect detection and repair costs are not as extensive provided that the timeline for delivering increments is short...so incremental delivery is one way to help mitigate these costs.

Another way is to follow what I like to call a life cycle approach to testing activities and work product delivery. Let's see how that works.

3

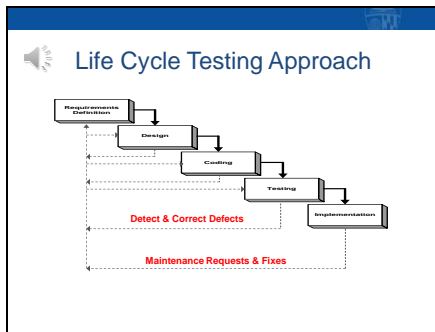


The life cycle approach to testing is characterized by two primary things: Some type of testing is performed at each major phase of the software life cycle, and testing activities are performed essentially in parallel to development activities.

Testing may be static testing...which is accomplished by using the review or inspection models we discussed earlier in the course...or it could be dynamic testing...meaning that the software is exercised using test data.

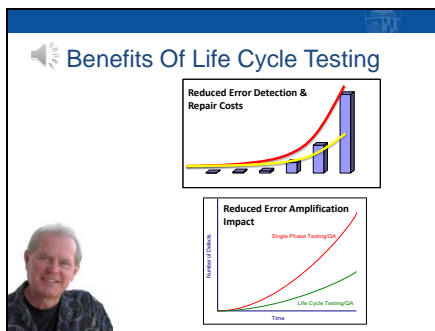
Testing in parallel means that while the developers are developing, the testers are also working on activities like developing a test plan, and developing test case data and scripts...and they will produce those work products at specific points in the project life cycle.

4



This diagram illustrates what happens with the feedback paths when a life cycle approach is used. There is more frequent error detection and repair activity and some error detection and correction feedback at every major life cycle phase.

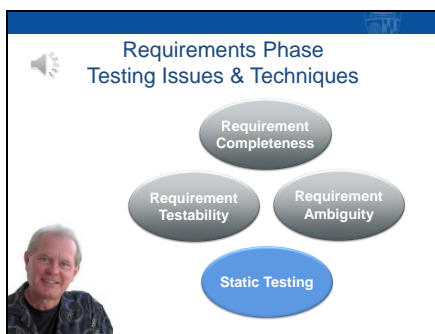
5



These illustrations point out pictorially the benefits of the life cycle approach. Error detection and repair costs are lessened, the error amplification impact is lessened...and the bottom line is lower overall costs and shortened project schedule.

Let's take a look at the life cycle approach phase by phase.

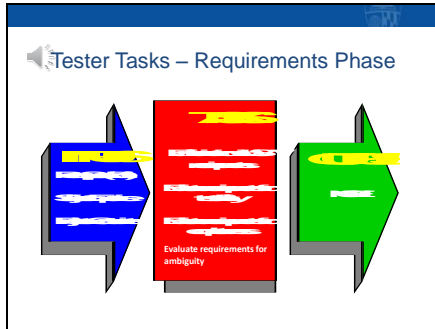
6



Let me start with the requirements phase. Major testing issues that can be addressed at the requirements phase include requirements completeness, requirements testability, and requirements ambiguity. These issues are somewhat interrelated. For example, a requirement may be untestable because it is ambiguous or incomplete.

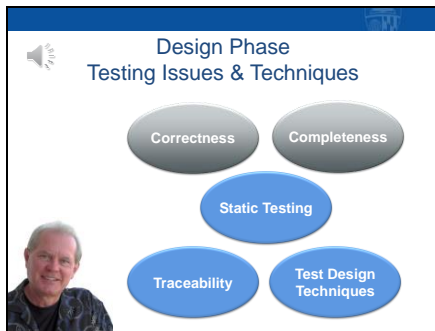
The type of testing that would be used at this point in a project would be static testing...namely conducting requirements walkthroughs or inspections. A tester's interest in those reviews would be to assure that the three issues are addressed.

7



In terms of tester work products...there are typically no formal work products produced at this point in a project.

8



During the design phase, the main design issues are typically correctness and completeness. From the designer viewpoint, these issues would apply to the software product design. From the tester viewpoint, these issues would be applied to the testing work products that are produced during the design phase.

In terms of techniques, static testing...walkthroughs or inspections...would be performed on product design and testing work products. In addition, traceability and test design techniques can be used. Traceability can be applied to the test plan to ensure the plan adequately covers the requirements. Test design techniques can be used to design actual test cases.

9

The table is titled 'Test Traceability Matrix'. It maps requirements to test sets. The rows represent requirements: UC 1, UC 2, UC 3, and NF RQT 20. The columns represent test sets: TEST SET 1, TEST SET 2, TEST SET 3, TEST SET 4, and TEST SET 5. 'X' marks indicate which test sets cover which requirements.

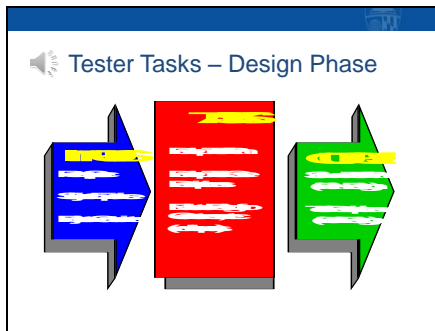
	Test Sets/Suites				
	TEST SET 1	TEST SET 2	TEST SET 3	TEST SET 4	TEST SET 5
UC 1	X				
UC 2	X	X	X		
UC 3			X		
.....					
NF RQT 20					X

Here's an example of a test traceability matrix. Each row corresponds to a requirement. In this matrix, I've used a row for each use case and a row for each non-functional requirement. The columns of the matrix correspond to test sets. Recall that a test set consists of one or more test cases.

In this example, test set one will be used to test use case one, test sets one, two, and three will be used to test use case two...and so forth. If there are any blank rows, that would indicate that we haven't designed any tests to test that use case or requirement...or, if we're using an incremental life cycle model, it might indicate

that requirement isn't scheduled to be tested in the current development cycle. Of course, we'd have to ultimately look at the specific test cases in each test set to determine if the coverage is really adequate...so the matrix is really a first-level look.

10



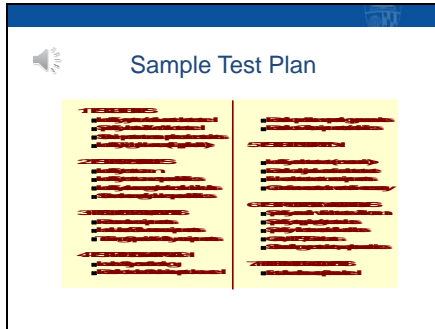
Here's a list of typical tester tasks and deliverables during the design phase. Two deliverables that would be produced under the life cycle approach are a test plan and a test description document. Samples of these documents will be illustrated shortly.

In terms of timing, a test plan would be scheduled for delivery about half way through the design phase, or at the end of preliminary design if a preliminary design phase is used. The test descriptions would be scheduled for delivery ideally by the end of the design phase.

In situations where there is no independent test group, I like to recommend that the developers produce a test plan during this phase...before coding begins...as well as at least a partial test description deliverable. This recommendation is very consistent with the notion of test-driven development.

When there is an independent test team, I also recommend that the testers and the developers review the test plan. Whenever I have witnessed this in my client organizations...it has been a win/win situation...and the developers have often changed their detailed design approach due to increased insight obtained from seeing things from the test viewpoint.

11

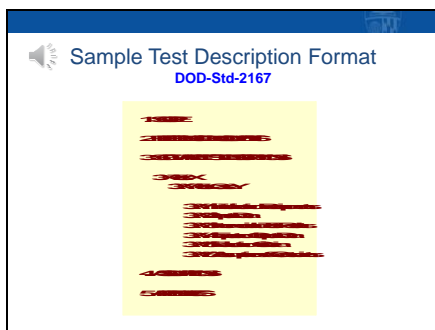


Here's a sample test plan template. This is actually one that I helped to develop for several clients. Most of the sections are self-evident. Some of the information might not be applicable for smaller projects...but the contents are very scalable. This plan has been used on single person projects and projects with dozens of team members. In practice, each "bullet" can be implemented using a form or a table, so the document is easily assembled and can be light on narrative.

Let me direct your attention to section five. In this test plan, testers identify test sets...not specific test cases. Recall the triangle program example from an earlier lecture. I identified a number of different test sets for testing that product, but I didn't provide specific test case data. Recall that the description of each of the test sets was adequate enough for us to estimate how many test cases would actually be produced.

Now...some test plans actually require that specific test cases be included...and that's okay. I like the idea of the test set first, followed by test cases later, because the test plan can be drafted and evaluated earlier in the project life cycle since less effort is required.

12

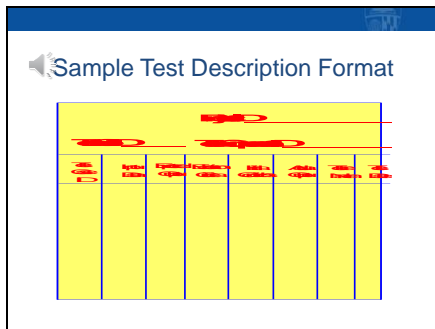


Besides a test plan, a test description was another work product that I identified as being delivered by the end of the design phase. So...what exactly is a test description document? It's a document that describes the individual test cases.

This template is from an old DOD standard. I first became familiar with it when I consulted with DOD contractors who had to implement processes to comply with the standard. I really liked this particular document. It starts with taking each test...what I've been calling a test set...and decomposing it into one or more test cases. For each test case, any initialization requirements, input data, and so forth must be documented. Recall that in the triangle testing example, my first test set contained three test cases...one for each valid triangle type. So...in this

template...section 3.1 would describe those three test cases.

13



A screenshot of a presentation slide titled "Sample Test Description Format". The slide shows a table with a yellow header and a white body. The header row contains the following text: "Test Case ID", "Test Case Description", "Test Case Version", "Test Case Status", "Test Case Author", "Test Case Reviewer", "Test Case Date", and "Test Case Status". The table body is currently empty, with only the first row visible.

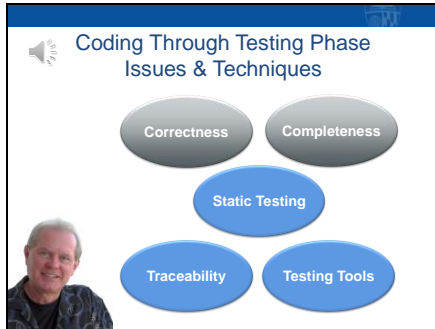
Test Case ID	Test Case Description	Test Case Version	Test Case Status	Test Case Author	Test Case Reviewer	Test Case Date	Test Case Status

Here's another style for a test description document. This is another example of one I designed for a client. This one is more like a spreadsheet. You would have a spreadsheet, or document section, for each test set.

As an example, for the triangle program, there would be a spreadsheet for the first test set, and since there were three test cases in that test set, three rows would be filled in to document the information for each test case.

At this point in a project we'd still be in the design phase...but look at how much in the way of formal test activities can be done...and how much insight there would be into the quality of the testing process. If there are deficiencies, they can be corrected with relatively little effort and cost and impact on the project schedule.

14

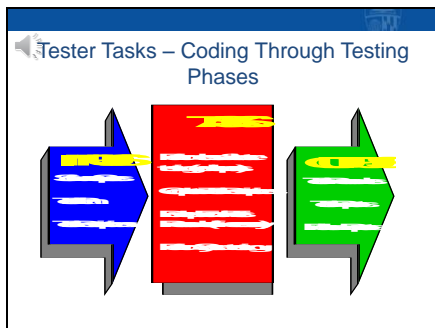


A similar set of correctness and completeness issues applies during the coding through testing phases. Of course, we would be performing dynamic testing during this period, but static testing could also be performed on additional test deliverables, and developers could also perform static testing on the code.

Traceability might also be applied to trace test cases back to requirements...but if the idea of test sets are used, test cases would only need to be traced back to their respective test sets since the test sets had already been traced to the requirements...so additional time is saved.

Testing tools might also be used during this period to help generate the actual test case data and to verify the extent to which tests actually cover code components.

15



Additional testing work products might be test procedures, possibly test reports, and what are typically called problem reports.

Test procedures, also called test scripts, document the detailed step-by-step instructions for executing each test case. They can consume a lot of effort, and some testing tools can generate them to save time and cost. A benefit of using test scripts is that tests can be repeated pretty exactly by anyone. A downside is that they can be time consuming to produce.

Test reports, if used, summarize the tests and their results for tests that were executed during a particular reporting period. Not everyone uses test reports, but they are not uncommon in contractual situations.

Problem reports, which can go by other names, are reports that get generated when a test fails...meaning when the actual and expected results don't match. The problem reports are routed back to the development team members for resolution.