STBE

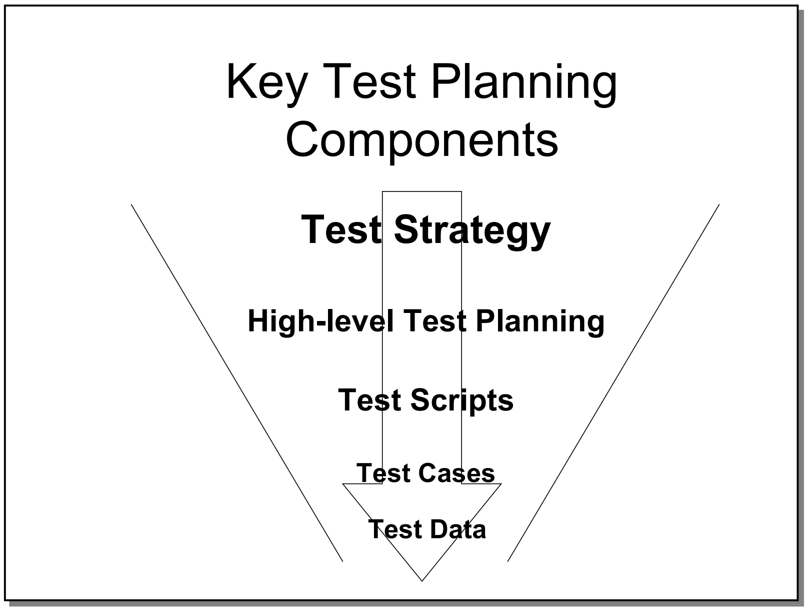
The Test Planning Process

Objectives

- Learn the test planning process from start to finish
- Understand how the test planning components fit together
- Learn how to write a test script

Synopsis

This module will take you from start to finish in the test planning process. You will learn about how all of the elements of the test plan fit together. You will also learn about test scripts and test cases and how to write them.



Key Test Planning Components

As we look at the test planning process in more detail, the following components will emerge as the major building blocks of a solid test plan. These components start at the high level strategy, proceed to a more detailed test plan, and then finally are seen as detailed tests.

- **Test Strategy**

This is a document that communicates the test approach to the rest of the organization. The test strategy can be developed very early in a project and requires only initial information to write. Whenever a new type of project or technology is being tested, the test strategy is one of the most important early test deliverables. We will examine what goes into writing a test strategy.

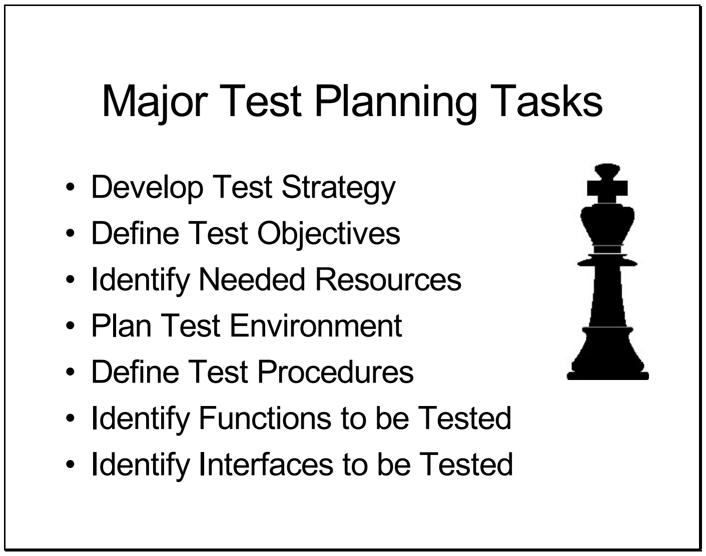
- **High-level Test Plan**

This is a document that describes the "who, what, when, where, and how" of the test. Test plans can be developed at a variety of levels, but we will focus primarily at the project or system level in this module.

- **Detailed Tests**

The high-level test plan will show which functions and attributes of the application are to be tested. The detailed test descriptions express these tests in terms that a tester can execute or can be translated into automated test scripts.

We will examine how to develop all of the above components and show examples of each.



Major Test Planning Tasks

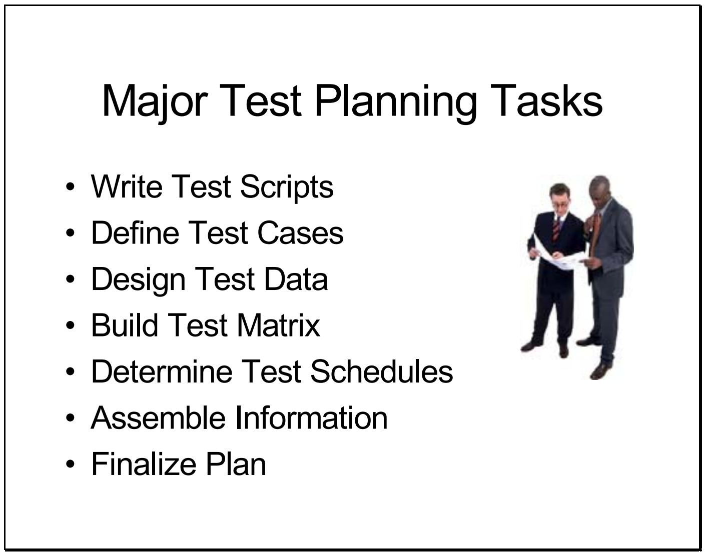
- Develop Test Strategy
- Define Test Objectives
- Identify Needed Resources
- Plan Test Environment
- Define Test Procedures
- Identify Functions to be Tested
- Identify Interfaces to be Tested

Major Test Planning Tasks

In this module, a 14 step process is presented for test planning. You may or may not need to perform all 14 steps, depending on your application, environment and risk. The 14 tasks are shown on these slides and will be explained in detail.

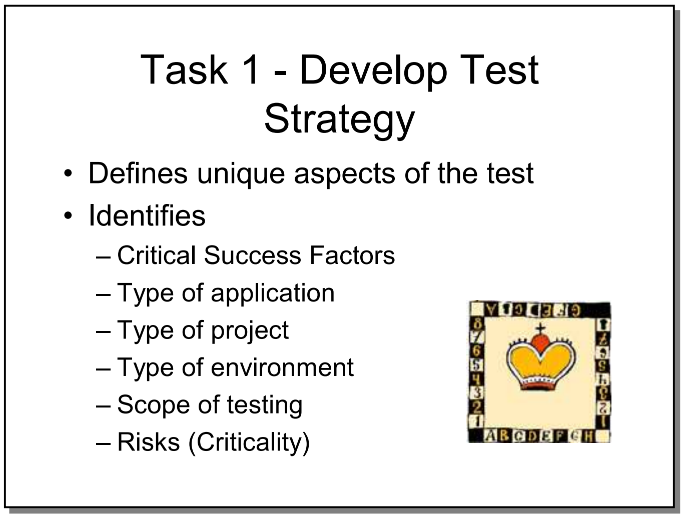
Something significant to note is that the planning process starts at the strategic level and gets more detailed until the end. Then, the details are assembled and summarized into a test plan.

The test plan can be written from any perspective, such as system testing, user acceptance testing, performance testing, etc.



Major Test Planning Tasks

- Write Test Scripts
- Define Test Cases
- Design Test Data
- Build Test Matrix
- Determine Test Schedules
- Assemble Information
- Finalize Plan



Task 1 - Develop Test Strategy

- Defines unique aspects of the test
- Identifies
  - Critical Success Factors
  - Type of application
  - Type of project
  - Type of environment
  - Scope of testing
  - Risks (Criticality)

Test Planning Task 1 – Develop Test Strategy

The test strategy is a high level document that describes the basic nature and unique aspects of the test. The test strategy is a tool for communication so that everyone on the project knows how testing will be conducted. Test strategies can be written early in the project, which allows for early testing.

A typical test strategy will identify:

- **Critical Success Factors**

Critical success factors are the attributes of an application that must be present for it to be considered a success. Examples of CSFs include correctness, usability, reliability, portability, interoperability and security. CSFs relate directly to test typs and project risks. The key is to select only 4 or 5 factors as critical, as each factor will require specific tools, people and processes.

- **Type of application**

This describes the purpose and objectives of the application, such as: batch, on-line, web-based, 32 bit Windows, e-commerce, data access, company intranet, customer extranet, etc.

- **Type of environment**

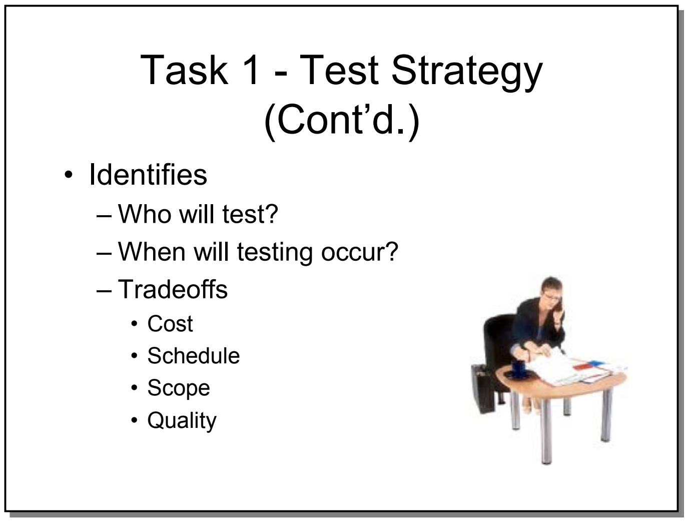
This describes the type of operating environments or platforms the application will be subject to. This can also include business and user environment, as well as physical environments, such as outdoors, public access, restricted access, etc.

- **Scope of testing**

This describes what will and will not be tested.

- **Risks**

Risks are any aspect of the application under test that exposes the organization to a loss.



Task 1 - Test Strategy (Cont'd.)

- Identifies
  - Who will test?
  - When will testing occur?
  - Tradeoffs
    - Cost
    - Schedule
    - Scope
    - Quality

Test Planning Task 1 – Develop Test Strategy

– Who Will Test?

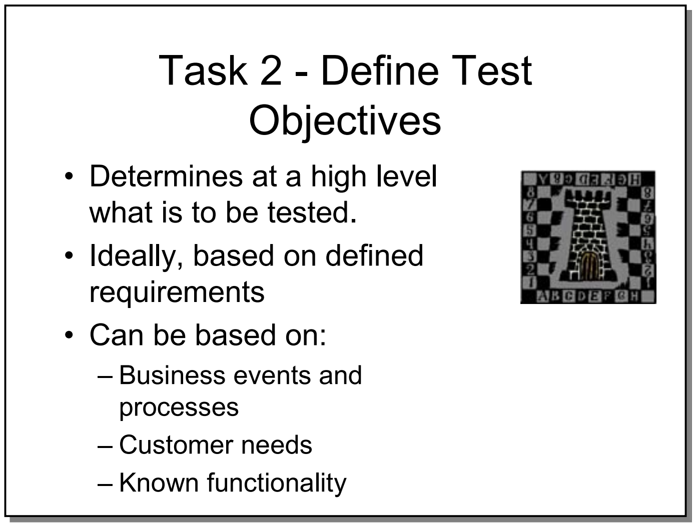
Exact names may not be known, but types of people can be defined.

– When Will Testing Occur?

Exact dates may not be known, but test activities can be matched with associated development activities.

- Tradeoffs

Tradeoffs are those aspects of a project that can be slipped to allow the project to be completed successfully. The key components of a project that are typically subject to tradeoffs are: scope, cost, schedule, and quality.



Task 2 - Define Test Objectives

- Determines at a high level what is to be tested.
- Ideally, based on defined requirements
- Can be based on:
  - Business events and processes
  - Customer needs
  - Known functionality

Test Planning Step 2 - Define Test Objectives

The test objectives should relate directly to project or system objectives. For each system or project objective, there should be a test objective.

You may choose to define your system test objectives by breaking them down by functional area or smaller sub-functions.

In setting test objectives, you are formally defining all the things the test should accomplish. In the ideal world, test objectives would be based on detailed defined requirements. However, testers do not often get the level of detail in requirements to write objectives. As an alternative, test objectives can be obtained from:

- **Business events and processes**

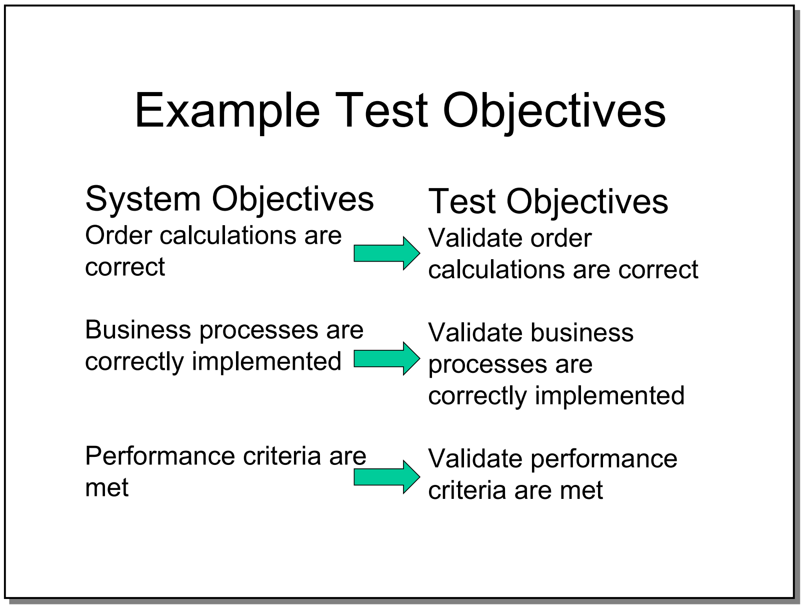
For example, you may know that customers will be placing certain types of transactions, requesting certain services, etc.

- **Customer needs**

You may know that customers will need quick response time on inquiries, or the ability to cancel transactions.

- **Known functionality**

There may be specific functionality that is very basic, but essential to the operation of the software. This would include functions such as searches, buttons, controls, etc.

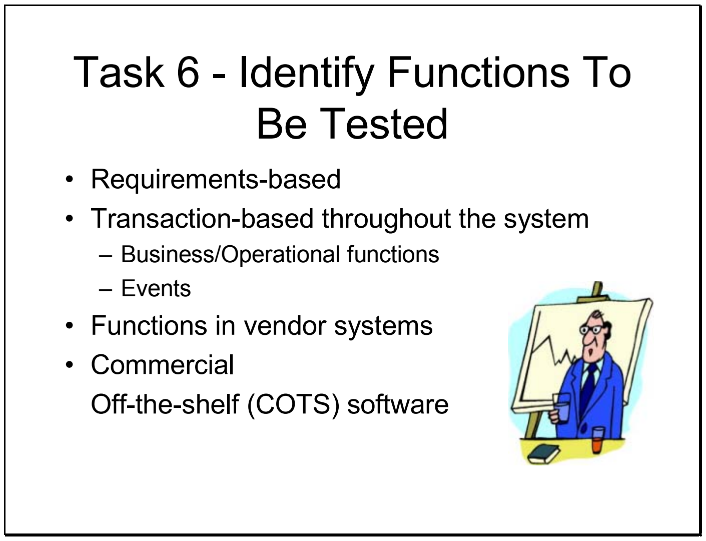


Example Test Objectives

System Objectives	Test Objectives
Order calculations are correct	Validate order calculations are correct
Business processes are correctly implemented	Validate business processes are correctly implemented
Performance criteria are met	Validate performance criteria are met

Example Test Objectives

In the above slide, the one-to-one relationship between system objectives and test objectives is shown.



Task 6 - Identify Functions To Be Tested

- Requirements-based
- Transaction-based throughout the system
  - Business/Operational functions
  - Events
- Functions in vendor systems
- Commercial Off-the-shelf (COTS) software

Test Planning Step 6 - Identify Functions To Be Tested

The next step is to define the system functions to be tested. A good way to view these functions to be tested is by transactions that traverse the systems in your organization.

Functions to be tested can be derived from a variety of sources:

- **Requirements**

Formally documented system or application requirements are usually the best source to find functions to be tested. However, much of the applicability of requirements for testing depends on if they contain enough detail and accuracy for testing. Furthermore, requirements may tend to change over the course of the project so the test process must be able to accommodate changing requirements.

- **Transactions**

Transactions are often a good source of test cases because they simulate real-world conditions that span many smaller functions. Examples of transactional tests include: posting accounts receivables, employee time approval, etc. These kinds of tests can be seen as business or operation functions, or as events in the business that is known to occur.

- **Functions in Vendor Systems**

In the case of vendor software, the development requirements are usually unknown, as is the detailed functionality. However, there is often a need to validate a vendor system's correctness and performance before it is placed into production use. User guides can be a starting place for designing test cases. Many times, the testers rely on generic interface tests, such as queries and updates, and transactional tests to be a source of tests.

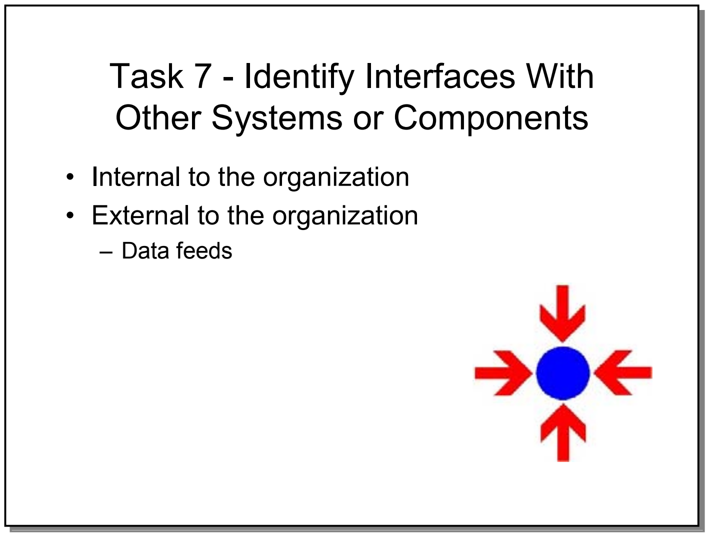
- **Commercial Off-the-Shelf (COTS) Software**

In some applications, there is a required dependence on off-the-shelf software from major vendors. While the tester might not be required to test all of the functionality in a COTS application, many times integration, performance, and ease of use are test concerns.

Test Planning Worksheet
Functions To Be Tested

System: Payroll System

#	Function to be Tested	Priority
1.	Pay rate calculation	High
2.	Vacation days calculation	Moderate
3.	Tax rate application	High



Task 7 - Identify Interfaces With Other Systems or Components

- Internal to the organization
- External to the organization
  - Data feeds

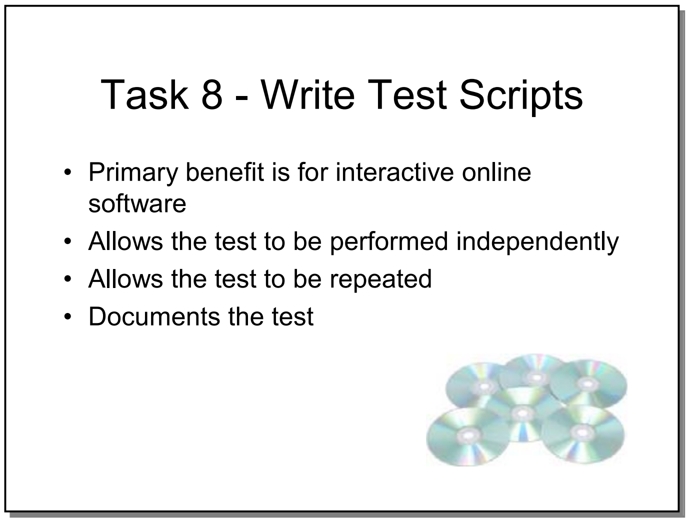
Test Planning Step 7 - Identify Interfaces With Other Systems

In this step, you define the interfaces with other systems both internal and external to your organization. To do this, you will need to identify the transactions that are passed between systems. For interfaces internal to your organization, the information required for this identification will likely be easier to obtain than those external to your organization.

Test Planning Worksheet
Identify Interfaces to Be Tested

System: Payroll System

#	Interfaces to be Tested	Other System(s)	Internal/ External	Priority
1.	Hours worked from timekeeping system	Timekeeping	I	High
2.	Taxes paid to IRS	IRS tax reporting	E	High
3.	Sick days used in year to personnel	Personnel	I	Low



Task 8 - Write Test Scripts

- Primary benefit is for interactive online software
- Allows the test to be performed independently
- Allows the test to be repeated
- Documents the test

Test Planning Step 8 - Write Test Scripts

Test scripts are a very helpful tool for testing interactive online software, such as CICS transactions. Test scripts add a great degree of rigor to your testing effort, but they do require a significant effort to write manually. The choice of whether or not to use test scripts depends on amount of test planning time and resources available.

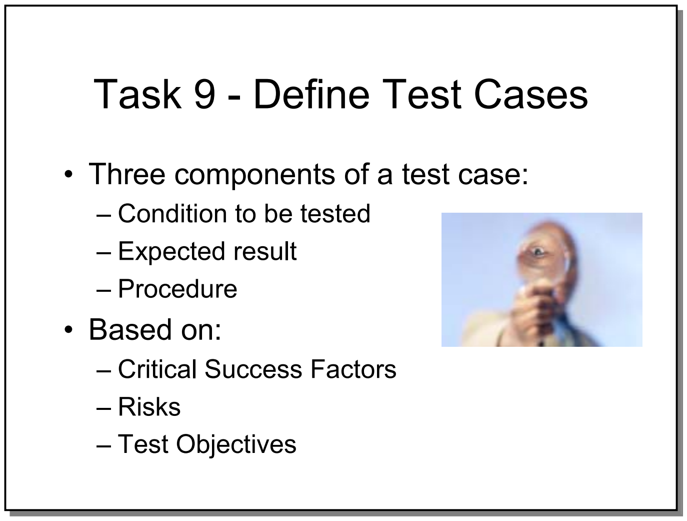
If the test is to be performed by someone other that the designer of the test, scripts are almost a necessity to convey exactly what should be done in performing the test. In addition, test scripts allow the test to be repeated. This is needed if a defect is found and the fix needs to be re-tested.

Another major benefit of test scripts is that they document the test. If there is ever any doubt after the test about which functions were tested, test scripts are a good reference.

Test Planning Worksheet
Test Script

ID#: PY001 **System:** Payroll System **Performed By:** ____________________________ **Date:** _____________

Step #	Module ID	Action	Expected Result	Observed Result	Pass/ Fail	Defect #
1.	PY001	Type employee name and ID in employee entry screen.	If employee is on file, basic information is displayed correctly. If employee is not on file, a message is displayed “Employee not on file. Do you wish to add?” Type “Y” and press ENTER key. The employee is added and message is displayed “Employee successfully added.”			
2.	PY001	Enter a pay rate effective in two pay periods.	Rate accepted.			
3.	PY005B	Run batch payroll job for the next pay period.	Payroll calculated correctly. The pay rate entered in step 2 is not applied.			



Task 9 - Define Test Cases

- Three components of a test case:
  - – Condition to be tested
  - – Expected result
  - – Procedure
- Based on:
  - – Critical Success Factors
  - – Risks
  - – Test Objectives

Test Planning Task 9 - Define Test Cases

Although people define test cases differently, the definition used in this course defines a test case as consisting of three components:

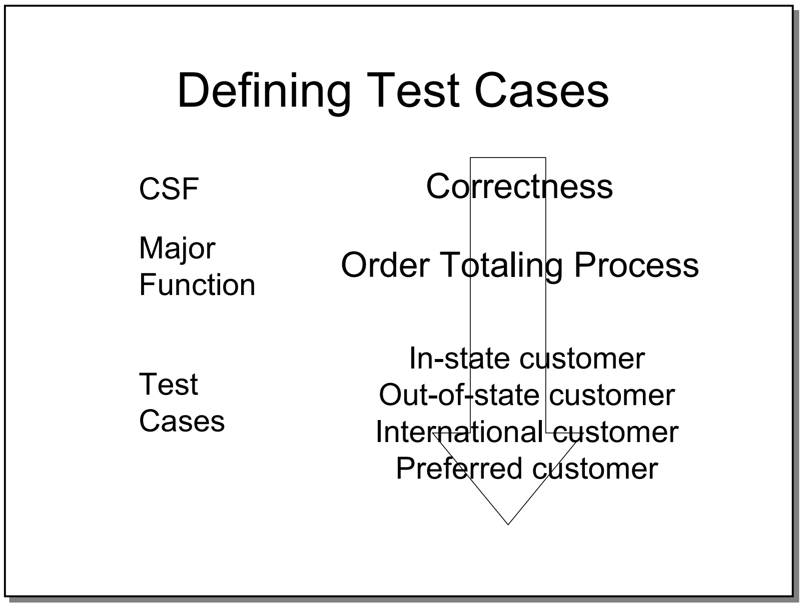
- **Three components of a test case:**

  -Condition to be tested

  -Expected result

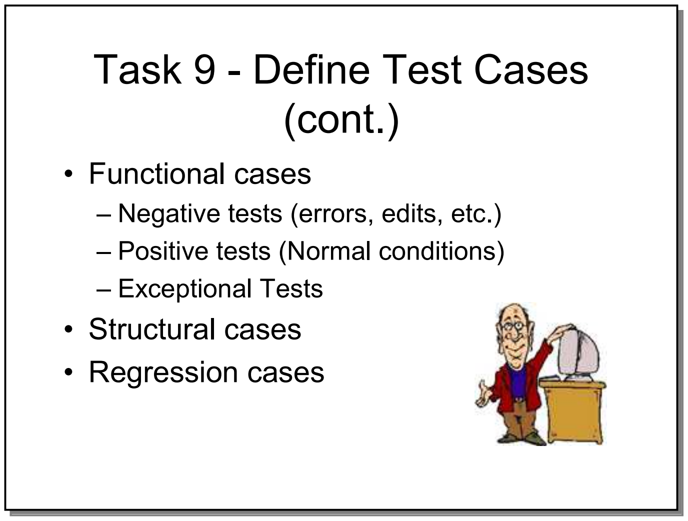
  -Procedure

This format of a test case allows you to define tests with a procedural test script. The advantage of such an approach is that it allows greater freedom when testing object-oriented and graphical user interfaces (GUIs). There is enough detailed information in a test case description as defined above to perform a complete test without a test script.



Defining Test Cases

In the above slide, the relationship between Critical Success Factors, major functions and test cases are shown with examples.



Task 9 - Define Test Cases (cont.)

- Functional cases
  - Negative tests (errors, edits, etc.)
  - Positive tests (Normal conditions)
  - Exceptional Tests
- Structural cases
- Regression cases

Test Planning Task 9 - Define Test Cases

In this step, we construct the detailed test cases that describe the conditions to be tested and expected results from the tests. These cases may be associated with test scripts, or can be performed without test scripts.

- **Functional Test Cases**

These are the cause/effect tests that are often referred to as "black-box" tests. These tests can be categorized as:

Negative tests, which are designed to invoke error messages and failure codes.

Positive tests, which are designed to simulate correct normal processing. (In normal daily processing over 90% of transactions will normally fall into this category.)

Exceptional tests, which are designed to simulate situations that occur only once in a while. These are significant to test, as most of the work in building an application goes into writing the code to process exceptional situations.

- **Structural Test Cases**

These test cases are designed based on knowledge of the internals of an application, such as code logic, data structures, interfaces, etc. Independent testers usually lack the time and perspective to design such tests, which is why it is important that developers perform this level of testing.

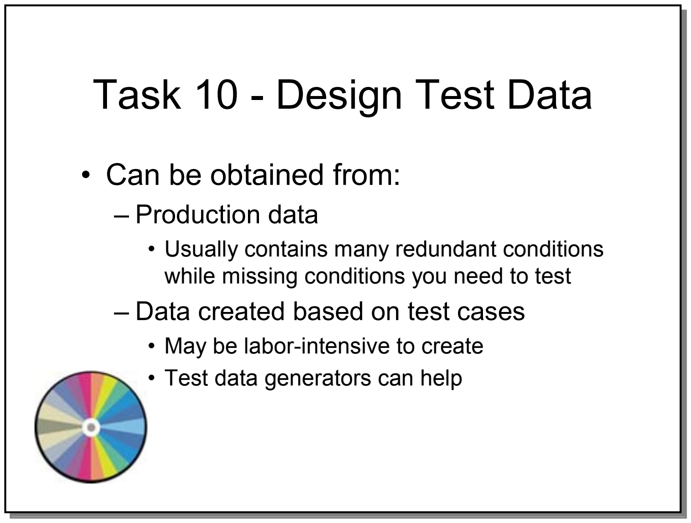
- **Regression test cases**

These are test cases that should be tested every time a change is made to validate an unintended defect is not introduced to the system.

Test Planning Worksheet
Test Cases

System: Notepad application

#	Condition	Expected Result	Procedure
1.	Open file by selecting from dialog box – no document currently displayed.	File selection box displayed. No message box displayed.	1) Open Notepad 2) Open a document
2.	Open file by selecting from dialog box – document currently displayed has changed without saving.	Message box displayed with message asking if the user wants to save.	1) Open Notepad 2) Open a document 3) Change the document 4) Open another document
3.	Open file by selecting from dialog box – document currently displayed has not changed since last save.	File selection box displayed. No message box displayed.	1) Open Notepad 2) Open a document 3) Open another document



Task 10 - Design Test Data

- Can be obtained from:
  - Production data
    - Usually contains many redundant conditions while missing conditions you need to test
  - Data created based on test cases
    - May be labor-intensive to create
    - Test data generators can help

Test Planning Task 10 - Design Test Data

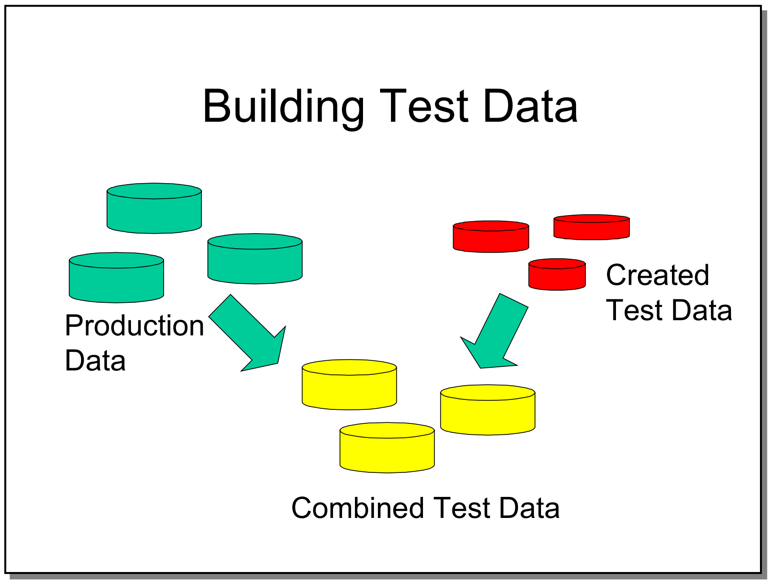
- **Can be obtained from:**

- Production data

There are problems with using production data for testing. Basically, you get a lot of the same transactions, while missing the test cases you really need. Plus, for web-based applications, production data might not even be available.

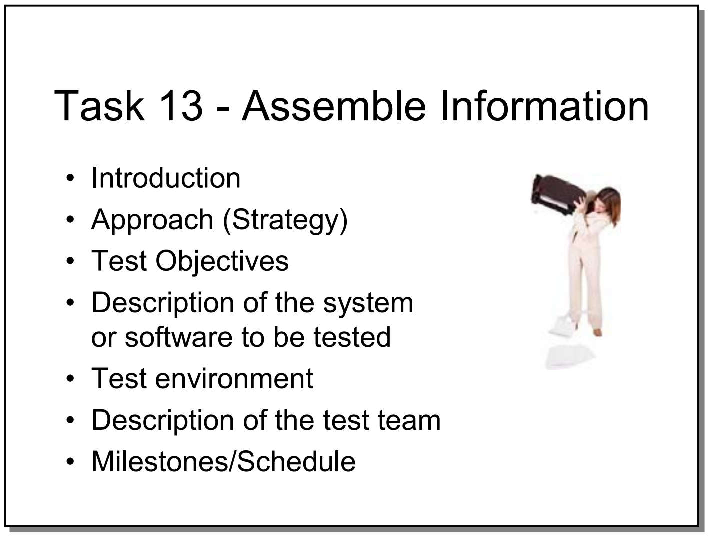
- Data created based on test cases

Test data based on designed test cases may be the cleanest way to go, but it takes time to build test data from scratch. There are test data generators available to help build test data for PC databases.



Building Test Data

In the above slide, the effect of building test data by combining production data and created test data is shown.



Task 13 - Assemble Information

- Introduction
- Approach (Strategy)
- Test Objectives
- Description of the system or software to be tested
- Test environment
- Description of the test team
- Milestones/Schedule

Test Planning Task 13 – Assemble Information

There are many different test standards available (IEEE, etc.). The following sections are commonly found in high-level test plans.

The test plan standard should be a part of your testing standards and processes. The test plan can include the following:

- **Introduction**

This is a general description of the project.

- **Approach**

This is the test strategy

- **Test objectives**

These are high-level objectives relating to system and project objectives.

- **Description of the System or Software to be Tested**

A brief, general description of the system to be tested.

- **Environment**

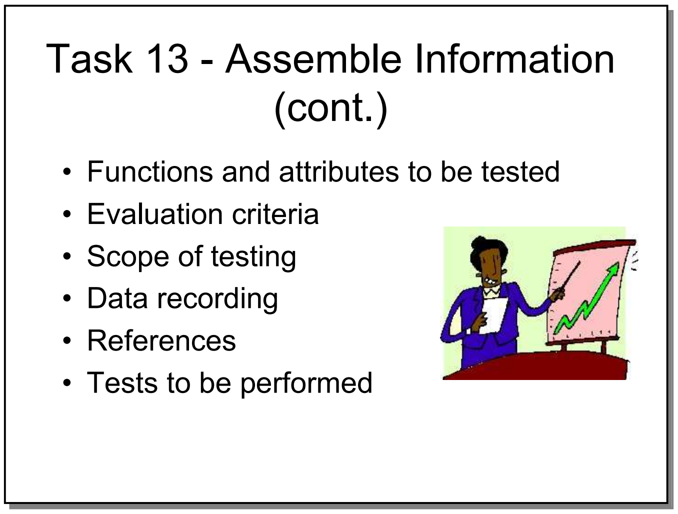
A description of the test environment, including specific hardware, software, data, procedures, etc.

- **Test Team**

This is a description of the people who will be planning, executing and evaluating acceptance testing.

- **Milestones and Schedule**

This tells when testing will start and start at the task level. Also, major milestones are shown in this section.



Task 13 - Assemble Information (cont.)

- Functions and attributes to be tested
- Evaluation criteria
- Scope of testing
- Data recording
- References
- Tests to be performed

Test Planning Task 13 – Assemble Information (Cont'd.)

- **Functions and Attributes to be Tested**

This section tells what parts of the system will be tested, both in terms of what the application or system should "do" and what it should "be."

- **Evaluation criteria**

This includes how test results will be evaluated to determine pass/fail status.

- **Data Recording**

This describes how the test results will be documented and how defects will be reported.

- **References**

These are any books or documents that are used as a basis for any aspect of the test. Examples include: test process documentation, testing books, test standards, system requirements, etc.

- **Tests to be Performed**

These are high-level descriptions of the tests to be performed. Since these can get voluminous, a common approach is to simply reference where the tests are documented.

Representative System Test Plan Outline

1. Introduction

   1.1 System Overview
   1.2 Test Objectives
   1.3 Scope of Testing

2. Approach

   2.1 Assumptions and Constraints
   2.2 Levels of Test Coverage
      2.2.1 Software Components
      2.2.2 Functions
      2.2.3 Interfaces
      2.2.4 Critical Requirements
      2.2.5 Business Requirements
   2.3 Test Tools
   2.4 Test Types to Be Performed (Regression, Conversion, etc.)
   2.5 Test Data

3. Plan

   3.1 Test Team
   3.2 Major Tasks and Deliverables
   3.3 Test Environment
   3.4 Training

4. Features to be Tested

5. Features Not to be Tested

6. Test Procedures
   6.1 Test Execution
      6.1.1 Test Cases
      6.1.2 Order of Testing
   6.2 Pass/Fail Criteria
   6.3 Suspension Criteria and Resumption Requirements
      6.3.1 Normal Criteria
      6.3.2 Abnormal Criteria
   6.4 Defect Management

7. Risks and Contingencies

8. Appendix
   8.1 Gantt Chart