

TEST PLANNING AND EXECUTION IN THE CLASSROOM

Gordon W. Couturier
The University of Tampa
gcouturier@ut.edu

Abstract

Testing is a critical part of system development and maintenance. Development of test plans [Pressman, 2001] during system analysis and design and before implementation is crucial to the production of a high quality system. Students graduating with any computer related degree should be trained in developing and executing test plans since the first position they take upon graduation will probably be in the software maintenance environment which is heavily dependent upon testing. Hence, system test planning must be taught while in the systems requirements specification stage and executed in the implementation stage. Process test planning needs to be completed in the design phase and executed in the implementation phase.

Keywords: Test Planning, Test Execution, Systems Analysis, Systems Design

Introduction

From the development of user/system requirements to the maintenance of an installed system, test planning and testing is a critical component of a designer's expertise. Testing includes unit testing, integration testing, system testing and regression testing [Rational, 2001]. Students need to be taught how to plan, develop and execute these various types of tests at the appropriate times in a two semester Systems Analysis and Design course.

Test Planning and Execution in System Analysis and Design

In systems analysis and design, during the System Development Life Cycle (SDLC) processes, [Dennis et. al., 2002], test planning must be included as follows:

1. Project Initiation and Planning
2. Systems Analysis - Understand Existing System, Develop System/User Requirements and **Develop System Test Plan**
3. System Implementation Selection
4. System/User Requirements Prioritizing and Time Boxing
5. System Design - Design Database, Processes and **Develop Unit Test Plans** and Components and **Develop Integration Test Plans**
6. Implementation - Implement Database, Code Processes and **Execute Unit Test Plans**, Implement Integration Code and **Execute Integration Test Plans**, and **Execute System Test Plan**
7. Version Release (Conversion)
8. Repeat steps 2 through 7 for succeeding version releases using new unit, integration and system tests along with **Regression Testing** to validate that the addition of new features and functionality did not harm previously implemented functions.

The model for developing and executing test plans can be illustrated as follows:

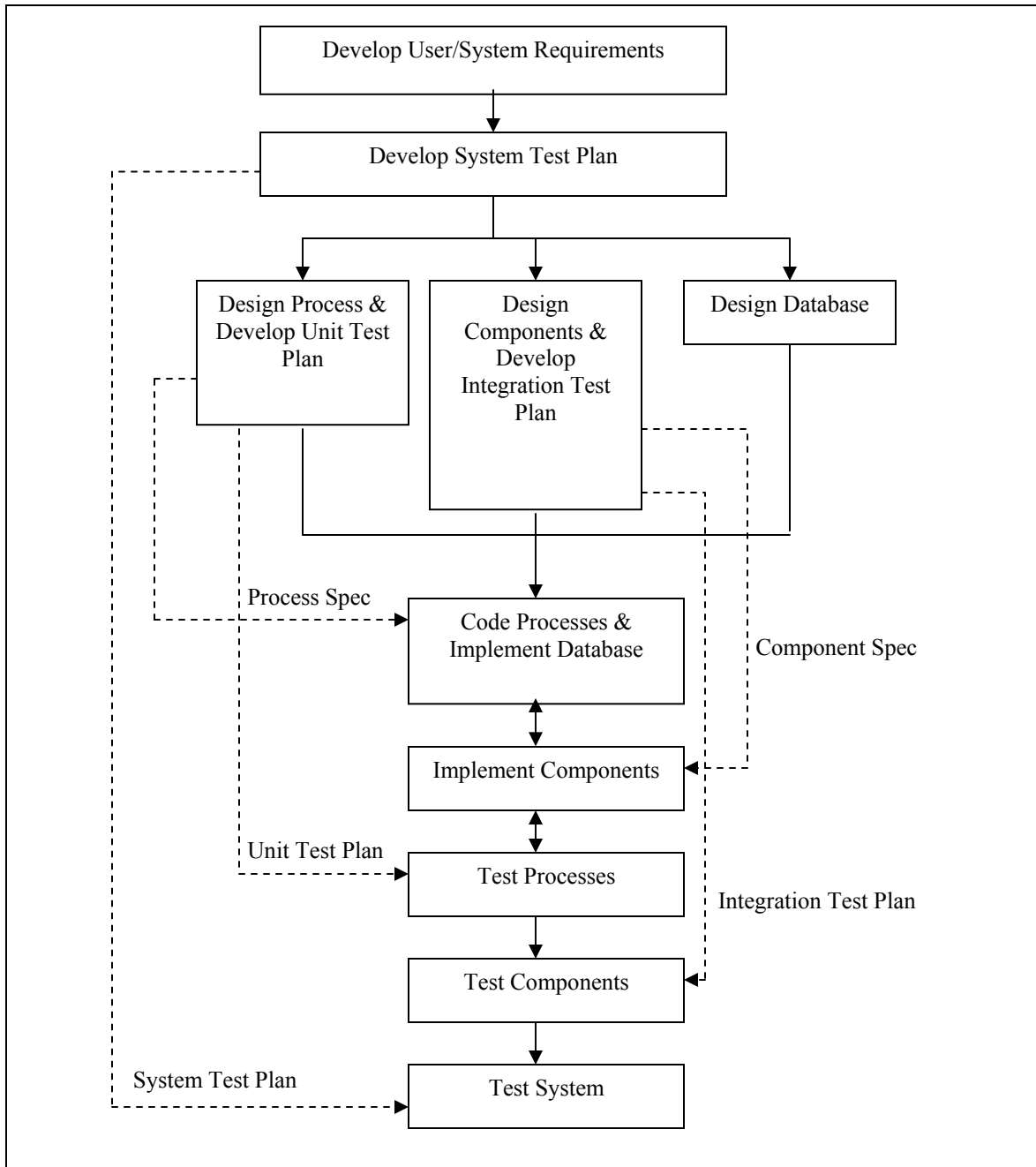


Figure 1. Model for the Development and Execution of Test Plans.

System/User Requirements and System Test Planning

The system/user requirements can be divided into many categories:

1. **Operational Requirements** (performance, information, economic, efficiency and service requirements)

2. **Control Requirements** (administration, change control, etc)
3. **Security Requirements** (system security, output security, access security, etc)
4. **Backup & Recovery Requirements** (frequency of backup, time for recovery, storage of backups, etc)
5. **Installation, Support And Maintenance Requirements**
6. **User Documentation Requirements**
7. **User Training Requirements**

System test planning is the development of normal and abnormal tests (makes sure that common user errors are detected and handled without the system failing) of specific requirements to make sure that a system has met user/system requirements after the system has been implemented. These tests should be developed before the decision on how the system is to be implemented to eliminate bias in the development of the tests. Therefore, system test plans should be made concurrently with the development of system/user requirements. Once the system/user requirements and system test plans have been made, system implementation decisions can then be made.

The procedure to develop a system test plan is to take each system and user requirement and develop tests of both normal and abnormal operation. As an example, assume that one of the system requirements is to require user sign-on using an ID and Password. Note that for every system requirement, there may be four or more tests to test both normal and abnormal operation. The test plan, after execution by the test person, would look as follows:

Table 1. Completed execution record of a system feature test plan.

System Test Plan					
System Name: <u>Jewelry Online</u>					
STP# <u>JO023</u> Version# <u>01</u>					
Plan Date: <u>June 6, 2002</u>					
Requirement Being Tested: <u>User signs-on with ID and Password</u>					
Test Planner Name: <u>John A. Smith</u> Employee Number: <u>007809</u>					
Test Date: <u>August 5,2002</u>					
Tester Name: <u>Mary Robinson</u> Employee Number: <u>010302</u>					
Test#	Test	Expected Result	Pass	Fail	Comment
1	At sign-on screen, enter a valid ID and Password, and click "Submit."	Menu selection screen appears	√		
2	At sign-on screen, enter a valid ID and invalid Password, and click "Submit."	Error screen appears advising user of "Invalid Password" error.	√		
3	At sign-on screen, click "submit" without anything entered.	Error screen appears advising user of "No Entry Made" error.		√	"Invalid ID" error message received.

4	At sign-on screen, enter invalid ID and any Password, and click "Submit."	Error screen appears advising user of "Invalid ID" error.	√		
---	---	---	---	--	--

Unit Test Planning and Execution

A unit is an individual programming segment that typically implements one simple function and has one input and one output (This is the definition of high cohesion found in Systems Analysis & Design (SAD) textbooks). In conventional SAD books, a unit test plan would be generated for each primary process in the lowest level data flow diagram decomposition. In object oriented analysis and design, the methods of an object class may be considered the primary processes of the system. This primary process or method (unit) is tested by itself in isolation from the rest of the system.

Unit test planning is performed during system design and before coding, when the system has been decomposed into its elementary functions or object classes. Unit test execution occurs after the software unit has been coded.

In the development of unit test plans, students have the tendency to assume that the rest of the system is there and they utilize other system functions to test the unit. Instead, they must be taught that the input must be initialized by hand and then the unit is executed to determine if the correct output is achieved. This is very difficult for students to comprehend.

To illustrate this test plan, the following User Access object with a process initiated by a message is shown:

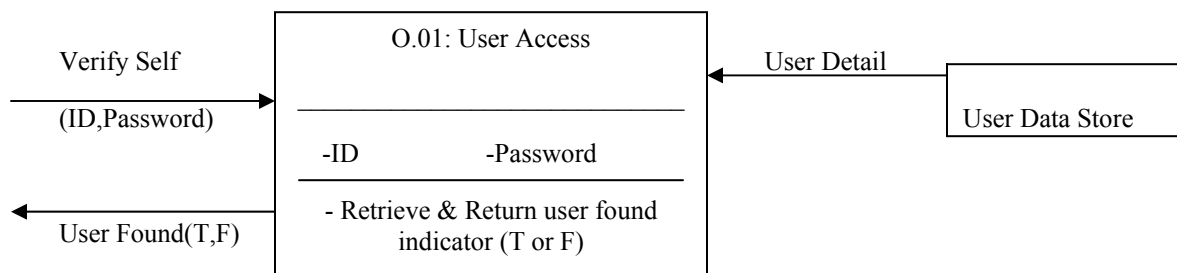


Figure 2. UML of User Access object's primary process and attributes.

The test plan for this primary process is as follows:

Table 2. Unit Test Plan for the User Access Object process in Figure 2.

Unit Test Plan					
UTP# <u>002</u>					
Version# <u>01</u>					
System Name: <u>Jewelry Online</u>					
Plan Date: <u>June 7, 2002</u>					
Unit Name: <u>Retrieve User Detail</u> Unit ID: <u>1.2P</u>					
Test Planner Name: <u>Mary R. Brown</u> Employee Number: <u>06799</u>					
Test Date: _____					
Tester Name: _____ Employee Number: _____					
Test#	Test	Expected Result	Pass	Fail	Comment
1	Set a known User ID and Password into the input message parameters (use user that has an entry in the User table) and execute the process	Stop at the exit instruction and verify that a "T(rue)" is being returned			
2	Set an unknown User ID and Password into the input message parameters and execute the process	Stop at the exit instruction and verify that there a "F(false)" is being returned			

Again, note that two tests are performed; one for the case where a user specified is in the User data store and the second for the case where the given user has no corresponding entry in the User data store.

Integration Test Planning and Execution

Integration test planning occurs in system design as objects are combined together to produce components, modules or subsystems (usually with some software "glue"). Execution of integration testing is performed whenever software units are combined to implement higher level modules or components. Of course, the final integration results in the completed system and the system test plan is then executed.

Again, the test planner must differentiate what is inside an integrated module versus what is outside to be sure that the tests are truly testing only this module's functionality. Input parameters must be set up and the tests are then executed. As an example, the next higher composition of the above unit module may be the "User Validation" component consisting of combining parts of the USER I/O object, USER object and the USER ACCESS object to implement this component. The component is shown below:

Sign-in Form

ID/Password

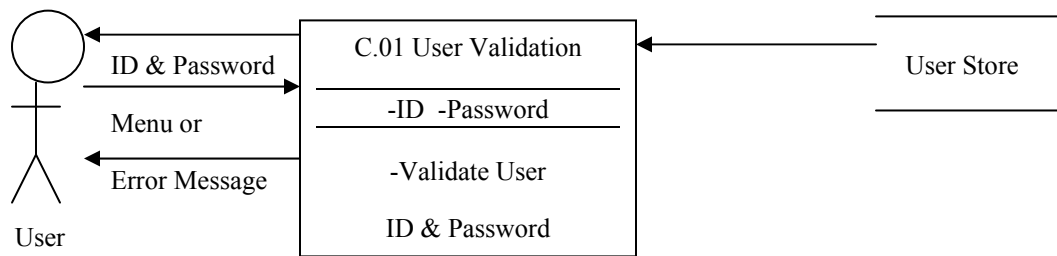


Figure 3. A component used for User Validation.

For the integration testing of the “User Validation” module, the following test plan could be developed:

Table 3. Integration (Component) Test Plan

Integration Test Plan					
ITP# <u>045</u>					
Version# <u>02</u>					
System Name: <u>Jewelry Online</u>					
Plan Date: <u>June 12, 2002</u>					
Component Name: <u>Validate User</u> Component ID: <u>COM12</u>					
Test Planner Name: <u>Greg Jones</u> Employee Number: <u>009899</u>					
Test Date: _____					
Tester Name: _____ Employee Number: _____					
Test#	Test	Expected Result	Pass	Fail	Comment
1	At sign-on screen, enter a valid ID and Password, and click “Submit.”	Menu selection screen appears			
2	At sign-on screen, enter a valid ID and invalid Password, and click “Submit.”	Error screen appears advising user of “Invalid Password” error.			
3	At sign-on screen, click “submit” without anything entered.	Error screen appears advising user of “No Entry Made” error.			
4	At sign-on screen, enter invalid ID and any	Error screen appears advising user of			

	Password, and click “Submit.”	“Invalid ID” error.			
--	----------------------------------	---------------------	--	--	--

Notice that the integration test in this case is the same as that portion of the system test plan that tests the User Sign on with ID and Password feature. This may happen but not necessarily.

Another example of an integration test plan for an E-commerce web site might be a checkout subsystem test plan. The E-commerce system will activate the checkout subsystem whenever a Checkout button is clicked. The executed test plan for this checkout subsystem would be as follows:

Table 4. Example of integration testing of the checkout subsystem of an E-commerce site.

Integration Test Plan					
ITP# <u>146</u>					
Version# <u>01</u>					
System Name: <u>Jewelry Online</u>					
Plan Date: <u>June 15, 2002</u>					
Component Name: <u>User Checkout</u> Component ID: <u>COM132</u>					
Test Planner Name: <u>Greg Jones</u> Employee Number: <u>009899</u>					
Test Date: <u>July 21, 2002</u>					
Tester Name: <u>Mary Robinson</u> Employee Number: <u>010302</u>					
<u>Test#</u>	<u>Test</u>	<u>Expected Result</u>	<u>Pass</u>	<u>Fail</u>	<u>Comment</u>
1	User clicks “Checkout” from catalog screen after putting several items in shopping cart.	Display appears showing all items in shopping cart, the quantity of each item, their unit price, total price, total cost of all items, tax, shipping & handling charge and total due. Number of quantity for each item can be changed.	√		
2	Change quantity of one item to 0 and click	Display appears with the item whose quantity	√		

	“Recalculate”	was set to 0 removed. New tax, shipping & handling charge, and total is displayed.			
3	Set all quantities to 0 and click “Recalculate”	Display shows No items in shopping cart.			
4	Restore items to cart that were deleted. Check that they reappear by clicking “Checkout.” Add 2 to the quantity of one item and click “Recalculate”	Display appears with the item whose quantity was increased correctly set at new quantity. New tax, shipping & handling charge, and total is displayed.		√	Tax was not updated. Shipping, handling and total calculated correctly.
5	Click “Return to shopping”	The display returns to the screen from which the original “Checkout” button was clicked.	√		
6	Click “Checkout” again and then click “Continue with payment”	The display first changes to the checkout screen and then, after clicking “Continue with payment,” the payment option screen appears.	√		

Please note that the payment subsystem is another component which this checkout subsystem activates.

Summary

Test planning is an extremely important part of systems analysis and design. Students need to be taught how and when to develop and execute test plans because their first jobs in their profession will probably be in software maintenance where these skills are vital. Test plans (unit, integration, system and regression) have to be properly developed and at the proper time during the system analysis and design cycle. Successful execution of well designed test plans on a system will result in the delivery of a high quality system.

References

1. Dennis A, B. H. Wixom and D. Tegarden (2002). “The System Development Life Cycle” in **Systems Analysis & Design: An Object-Oriented Approach with UML** (pp. 3-7). Hoboken, NJ: John Wiley & Sons, Inc.
2. European Union Data Grid Project (2001). **Rational Unified Process Test Plan** . Retrieved October, 2003, from

http://hep-proj-grid-fabric.web.cern.ch/hep-proj-grid-fabric/admin_procedures/docs/others/rup.pdf

3. R.S. Pressman & Associates, Inc. (2001). **Test Specification: Document Template**. Retrieved October, 2003, from <http://www.rspa.com/docs/Testspec.html>