Developing a Test Plan

Neville Turbit

Overview

In a previous document we covered the development of a test strategy. This white paper covers the development of a test plan. It provides a checklist of testing situations and guidelines on how to create a test plan.

Test Resource Utilisation

Before we start, it is useful to think about the sorts of skills you need to do each part of the testing. In other words how do you structure testing resources to best utilise their time?

It takes considerable business knowledge to understand the situations that need to be tested. For example, in a transaction based system, you need to have an in depth understanding of the business rules for the area in order to know how the system should respond.

To actually carry out the testing requires a lesser skill. If a person has the input, and expected output, the keying does not require a highly skilled person. As long as the results follow the script, the work can be carried out by mid level staff in terms of business knowledge. In fact, it is often the least experienced staff that will identify problems in testing because they start with less assumed knowledge.

Test Scenarios and Test Scripts

The first step in developing a test plan is to develop test scenarios. A test scenario is something like"

"When we enter a new client, there should be a warning if the person is already registered."

The scenario does not include the test scripts which would look like this.

- "Enter customer Smith and Co. and save
- Re-enter the same customer
- You should now get a warning message asking you if this is the same name as the existing customer."

The script above is not very detailed however we will cover what a script looks like later in the document. The purpose is to show the difference between a test scenario and a test script. The script contains the actual *information* to be entered while the scenario describes a *situation* to be tested.

Creating Test Scenarios

It is your most skilled resources that should be creating the test scenarios. They best know what the system should do to assist them. The first point of call for developing the scenarios is the business requirements. They should identify the functionality to be tested.

There will however be other things to test. Typically these include things like screen layout, validity of information displayed, speed, correct data capture etc. Below is a list of things to consider in testing. Some of these will apply to user acceptance testing and some will apply to other types of testing

- Unit testing
- Performance Testing
- Integration Testing

Report and Enquiry Function

- All field names on the report are correct and consistent between reports.
- All screen and report layouts are the same.
- Blank lines occur where they should
- Date and time are printed where they should be
- Detail lines which contain only zero values do not print
- Field names on reports and enquiry screens are correct.
- Fields are edited correctly
- Fields print their full name.
- It meets the requirements as specified in the Business Analysis documents and/or Change Requests
- Meets report coding and layout standards
    - Negative values appear as required. CR, -,
    - Zero values print as required (blank, 0.00, 00.00 etc)
- Only fields requested appear
- Only records selected appear
- Page breaks occur where they should
- Page numbers change on each page
- Program name and number are printed correctly (if required)
- Report is printed in the correct sort sequence
- Rounding errors do not occur
- The report layout matches the users' requirements
- The report/screen layout is appropriate and easy to read
- Totals add up correctly
- Totals add up consistently across reports and screens
- Totals are reset when they should be
- Totals do not get truncated

Input Screen Tests

- All field names across the screens are consistent and correct.
- All screen layouts are the same
- Automatic duplication of data from one record to another where appropriate
- Insert, update and delete operations function correctly across screens

- Keyboard usage the same
- The correct records appear on the screen
- The system behaves consistently always giving uniform results
- Totals add up correctly and consistently across screens
- Character Tests
    - Skipping mandatory fields
    - Skipping optional fields
    - Numerics only in numeric fields (no alphabetic or characters)
- Date and Time Value Tests
    - Start and end of month and years
    - Correct format of date and time
    - Month is in the range '01' - '12'
    - Date is greater than or equal to today's date
    - 'Date to' field greater than or equal to 'date from' field
- Field Value Tests
    - Zeros where appropriate and printed in correct format
    - Minimum positive and negative values
    - Maximum field sizes
    - Pick lists available and display description
    - Field level help
- Range Tests
    - One less than lower/upper limit
    - Equal to lower/upper limit
    - One higher than lower/upper limit
    - Middle of range
    - Maximum field's value equal or greater than minimum field's value
    - Maximum field's value not less than minimum field's value
- Table Validation Tests
    - Equal to first value in table
    - Equal to middle value in test
    - Equal to last value in table
    - Not in table
- Terminal Screen Specific Tests (if appropriate)
    - All field names are correct
    - Operates correctly on all terminals it is supposed to i.e. dumb terminals, PC, Windows 98, 2000, XP, Windows NT, etc

Update Function Tests

- Add a record already on file
- Add a record not on file

- Change a record on file
- Delete a record
- Enquire on a record not on file
- Enquire on a record on file
- On delete, checks for foreign keys and correctly follows referential integrity rules.
- System generated primary key is correct
- Try adding invalid data to the new record

Security

- Unauthorised users do not have access to the system
- User has access to authorised data
- User has access to authorised menu options/functions
- User is only able to do authorised functions i.e. insert, update, delete, enquire

Input Screen Tests

- Are function keys used consistently
- Are there pull-down windows (pick lists) available where appropriate
- Error messages and warnings should be meaningful and consistent
- It should be easy to navigate from one screen to next
- Next enter incorrect data into each field. Enter invalid dates, times, negative values, codes, etc.
- Screens should have a consistent 'look and feel'
- Try to 'crash' the system by entering invalid data and pressing invalid sequence of keys

User Documentation

- Easy to follow
- Easy to navigate or find information
- Reflect what the system requires the user to do

Hardware, Network, Printers

- Test that all appropriate hardware, network and printers are available and function correctly

Integration Test Examples

- Access to data and functionality in other systems
- Conformance to programming standards
- Consistency of data held in this and other systems
- Consistent look and feel with other parts of the system which have been previously implemented.
- Integration with other functions and programs
- Passing of information between systems
- Performance characteristics eg. Elapsed times of queries, movement of cursor etc.

- Record locking
- Timing issues related to data transfer
- Validity of data coming from, and passed to other systems

Performance or Stress Testing

- Access to archived records
- Bandwidth issues
- Batch processing windows
- Complex transaction processing
- Crash recovery time
- Month end and year end runs
- Performance on machines with a minimum configuration
- Performance over WAN
- Performance under peak transaction volumes
- Performance whilst batch processes running
- Printer queues
- Response time for enquiry
- Response time for printing – particularly documents with graphics
- Screen refresh times
- Storage and retrieval of overnight reports
- Time for backup and restore
- Time to load the System
- Time to save records
- UPS performance

Example Test Scenario

The following is an example of a test scenario. Note that there is no specific data identified for testing. It is purely the "situation" that should be tested.

Data input, modification and deletion

The following inputs, modifications and deletions will be tested.

Screen 1

No.	Field	Type	Input	Result	Notes
D.1.1	*Name*	*Alphanumeric*	*Current Client*	*Error Message*	
D.1.2			*New Client & try to save*	*Prompt for address*	*Address is mandatory*
D.1.3			*New Client with name exceeding 50 chars*	*Error Message*	*Max size is 50 chars*

No.	Field	Type	Input	Result	Notes
D.2.1	*Address*	*Alphanumeric*	*Enter a new client and give an address the same as an existing client*	*Warning message*	
D.2.2	*Etc.*				

Example Test Scripts

Having developed the scenarios, particular test cases can be specified using the test scripts. These sheets can be used to check off the testing as it is completed. They can be prepared by either the people who prepared the scenarios or by lesser skilled staff who might not have the depth of knowledge to develop all the scenarios but can find examples to test.

They should be reviewed by the people who developed the scenarios for completeness. It may be appropriate to prepare this as a spreadsheet so that data can be sorted more effectively.

Test Scripts

Field	Scenario	No.	Input	Expected Result	Tested By	Date	Notes	Completed Successfully
Client Name	Enter an incorrect client number	D.2.7	X1234	Error Message "This client does not exist"				
	Enter a client from another Branch	D.3.8	Enter into NSW branch client VIC123	Error Message "This client is from another branch. Make client a NSW Client (Y/N)				
		D.3.9	Client is made a local client	Order can be processed and client appears on local client screen.				
		D.3.10	Client is not made a local client	Order cannot proceed. Error Message "Order cannot be processed against another branch client"				

Random Testing

Allow time for people to play with the system. They will find more creative ways to break it than you will ever think of. The following is a real example.

In an insurance system I had implemented, annual statements for investment accounts were calculated by adding 12 months interest to the closing balance from the prior year. The closing balance was stored on the system. Any payments during the year were treated as follows. Work out the number of days since the payment and add interest at a daily rate.

One new account had a balance of around $1,000 and interest of $75,000. Fortunately we picked it up before the invoice was mailed. But how could that happen? The logic for the calculation was straight forward.

Investigation turned up the reason. The year was 1999 but the person entering the payment miss-keyed. They put in 1899. The system did not have a prior balance so it identified the policy as a new policy. It then calculated 100 years of interest!

There were two lessons we learned. Firstly the input screen should have a validation for the year when it is input, and secondly we needed to run exception reports to pick up obviously wrong interest amounts. We would never have thought to test such a situation but a "rogue" user who was trying to break the system may have found it.

Summary

It is critical to successful testing that a test strategy is put in place before the test plan. Without a strategy, the test plan will most likely fail due to lack of supporting infrastructure.

The critical thing with a test plan is to take a top down approach. Identify the areas requiring testing (test scenarios) then flesh out the details with the test scripts. Use only the level of resource skills that are required. There is no point in having a highly skilled resource entering data all day.

Finally, accept that you will not think of everything. Let your most feral users onto the system and watch what they turn up.

Neville Turbit has had over 15 years experience as an IT consultant and almost an equal time working in Business. He is the principal of Project Perfect. Project Perfect is a project management software consulting and training organisation based in Sydney Australia. Their focus is to provide creative yet pragmatic solutions to Project Management issues.

Project Perfect sell "Project Administrator" software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called "Method H"™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au