System Testing Standards

Version 2.03

Author: **Vincent. Coen**
Date: **18th January 2025**

DOCUMENT CONTROL SHEET

Version	Name	Changes	Date
1.00	V. Coen	Draft Issue	06/05/1994
2.00	V. Coen	Additions for Tier Tech	02/06/1997
2.01	V. Coen	Additions for A. N. Other	11/12/1998
2.02	V. Coen	Additions for ACAS	14/04/2001
2.03	V. Coen	Font,size adjustments	24/04/2024

Authorised By:

Date: Version:

DISTRIBUTION LIST

Name	Company	Copies

Contents

1. Introduction.

1.1 Scope and Objectives of this Manual.

Whilst testing, in itself, is not the only aspect of developing quality systems, it represents the last stage at which we may ensure that the product we will deliver is of the highest quality. It is for this reason that special care and attention should be paid to this phase.

Testing is a fundamental requirement for the development, and successful implementation, of any product, and this is no less true of computer systems than of any other type of product. Comprehensive testing of the systems that we develop has three major benefits:

- Customer confidence in the testing strategy;
- Satisfaction that we are providing the users with the best possible solution.

The objectives of this Manual are to define:

- The processes which must be undertaken in the conventional Test Cycle;
- The procedures that should be followed;
- The responsibilities at each stage of the Test Cycle;
- The documentation needs at each stage.

It is important to note here that we must be in a position to demonstrate that:

Testing

Special attention should be paid to the following aspects of testing:

a) the test results should be recorded as defined in the relevant specification;

b) any discovered problems and their possible impacts on any other parts of the software should be noted and those responsible notified so the problems can be tracked until they are resolved;

c) areas impacted by any modifications should be identified and retested;

d) test adequacy and relevancy should be evaluated;

e) the hardware and software configuration should be considered and documented.

Validation

Before offering the product for delivery and acceptance, the supplier should validate its operation as a complete product, when possible under conditions similar to the application environment as specified in the contract.’

(The TickIT Guide to Software Quality Management System Construction and Certification).

1.2 Scope and Objectives of the Test Cycle.

The primary objective of testing is to ensure that the developed system conforms to the requirements as defined in the Business Requirements Specification and the Functional Specification. For this reason, the criteria against which the system will be judged, and accepted, should have been identified in those documents. If this is not the case, then acceptance criteria will have been defined in another document. This document may be defined by the Client.

Therefore, the scope of the Testing phase of the S.D.L.C. (Software Development Life Cycle) is restricted to the Test Plans.

To facilitate the testing of the completed system, the Test Plans should be broken down into the standard Test Cycle elements:

- Unit Testing;
- Link Testing;
- System Testing;
- Volume Testing;

User Acceptance Testing.

These are the classic elements in most, if not all, Test Cycles for any software development. There is often an additional stage that being:

- Regression Testing

This is the concept of re-testing from the beginning, every time, regardless of where in the test cycle a problem is identified.

Each of these is discussed in more detail below.

The Test Cycle should normally follow the sequence, as outlined in Figure 1 below, however it is acceptable to have overlaps - it is possible to begin one stage of the Cycle before the previous one has completed.

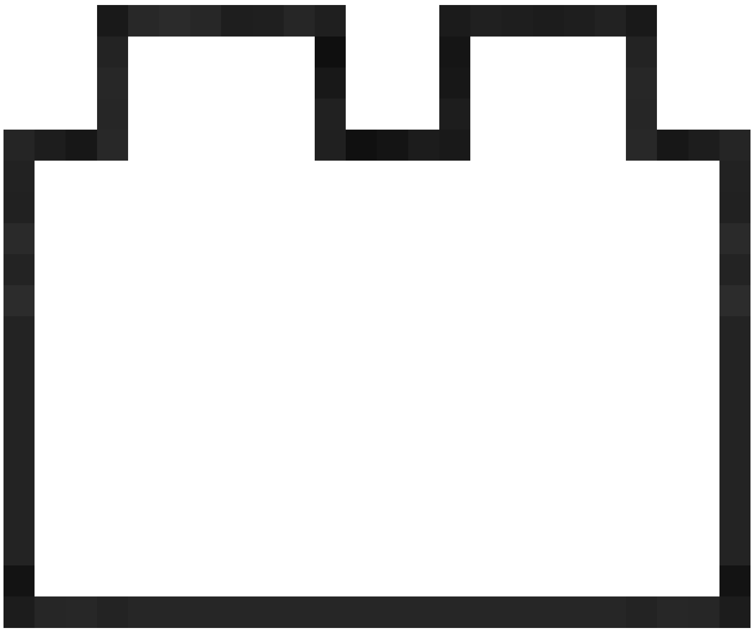

Figure 1 - Project Test Cycle.

Prior to the commencement of the Test Cycle, the Project Manager (or designated deputy) should construct a Test Plan. The Test Plan should be part of the Project Plan, or a separate document referred to in the Project Plan. The Test Plan should be created ***as early as possible*** in the S.D.L.C. For example, the Unit Test specification may be created immediately after the Functional Specification sign-off. The Test Plan must identify all of the major components of the project's test cycle, and within each component:

- What is to be tested;
- Who is to perform the test (if this is known);
- How it is to be tested;
- How long should it take;
- What the expected results should be.

A suggested structure to the Test Plan document is contained in Appendix B.

2. Unit Testing.

2.1 Scope and Purpose.

The purpose of Unit Testing is to ensure that:

* Business logic for both positive and negative conditions conforming to the Functional Specifications.
* Accessing Database tables is processed in the correct and most efficient manner, i.e., all SQL statements have been checked with the use of ‘Explain’ where applicable.
* All data entered is stored correctly, in the correct file or DB table and in the right format;
* All data required to be displayed is retrieved correctly from the right file or DB table, and in the right format;
* All fields on all screens within the module conform to the format and validation rules as defined in the Functional Specification;
* Navigation into, and out of, screens within a module conforms to the Functional Specification definitions;
* Validation error messages are correct, meaningful, and appropriate to the error condition.

Unit Testing needs to test all functionality that can be checked without integrating the module, **including** database or file processing. Therefore, Unit Testing is limited to the module itself. There is no requirement, at this stage, to go beyond the confines of the module. However it is important to remember that a module may consist of more than one form, and any interaction between these forms must be fully tested as well as all business logic.

2.2 Procedures.

Test Plans for Unit Testing should have been created following the development of the Functional Specification. It is the responsibility of the Project Manager to ensure that these are available prior to being required.

The Team Leader is responsible for giving the programmer the generic Program Unit Test Checklist - an example of a GUI element is included as Appendix C. The **format** of the Checklist is not important, and it is not essential to conform to the example, however the **content** is very important in that it **must** show all processing requirements. Additional checklists must be included to cover all business logic that the program is charged with. For this purpose the Functional Specification must be used to construct the checklist and **not** the code. The use of the word ‘program’ should also mean module or run unit where required.

Unit Testing will be conducted by the programmer on completion of the program, and prior to the release of the program to Link Testing. This should be the norm, however it is acceptable for another programmer to perform Unit Testing.

As it is the originating programmer performing the Unit Testing, it should not be necessary to raise Fault Reports for any problems identified - these can be fixed immediately, and re-tested.

It is **not** recommended that Unit Testing is conducted after being released from the control of the programmer. It is the programmer’s responsibility to ensure that Unit Testing is done as faults found at this stage take less time and therefore have the lowest cost to resolve. It is recognised, however, that this is not always possible. When this happens, it will be mandatory to raise Fault Reports for any problems identified - refer to the Fault Reporting Procedure for more information.

It is not acceptable for a programmer to release a program to the Test Team without some **basic** structured testing having been done. The Program Unit Test Checklist represents the **minimum** level of testing that must be performed.

As each element in the Checklist is tested against the program, the appropriate Tick Box should be completed with one of the following responses:

- A ✔, which denotes that the test has been performed, and passed;
- A ✖, which denotes that the test has been performed, and failed. The programmer must then resolve the problem and re-test. A further ✔ inside the Tick Box will then indicate that the problem has been resolved;
- The comment ‘N/A’ which indicates that the corresponding test is not applicable in this instance (for example the Radio Button tests for a program that does not contain any Radio Buttons).

When all Unit Testing has been completed (indicated by a Checklist that now only contains ‘N/A’ or ✔), the Checklist should be signed, and dated, by the programmer. The program may now be released to the next stage in the Testing Cycle. The Program Unit Test Checklist should be handed to the Team Leader to confirm completion, and it will then be filed in the Project Folder.

The Project Test Completion Form (Unit Test Section) must be signed and dated by the Team Leader, and filed in the Project Folder.

2.3 Program Unit Test Checklist.

The Unit Testing can be further sectionalised into the following areas:

- Functional. This section deals with the way that a program works, with specific reference to the Functional Specification, in isolation from other aspects of the overall system;
- Navigational (GUI). This section deals with the way that the program navigates, both internally (considering tabbing and Form-to-Form communication), and externally (in terms of its movement from and to other system elements);
- Cosmetic (GUI). This section deals with the look and feel of the program in the overall context of (a) the user’s specific requirements; (b) its conformance to the programming standards for Screen Design; and (c) its conformance to the accepted standards in terms of the IBM Common User Access (CUA), or the Microsoft Windows Interface: An

Application Design Guide in Microsoft Windows SDK, both of which constitute a common standard for GUI Application Systems.

One advantage of this sub-division is that the Unit Testing may be undertaken by a team of testers, each with their own focus of attention on specific aspects of testing.

Unit Testing should consist of the following checks (refer to Appendix C for a detailed GUI process checklist). Obviously non GUI modules do not require the GUI test check lists but must include a check list containing master test cases created from the functional specifications.

Functional (Business).

- Business logic;
- Database table accessing including SQL processing;
- Data processing and display (if applicable);
- Millennium 2000 date compliant throughout;
- Error recovery;
- Field overflow protection;
- Internal tables overflow protection;
- Invalid data recovery.

Functional (GUI).

- Check Boxes;
- Combo Boxes;
- Drop Down Boxes;
- MEW (Multiple Entity Windows) Format validation;
- Multi-select Boxes;
- Single-select Boxes;
- Delete Push Button from Named List;
- Dialog Push Buttons validation;
- Dialog Push Button validation (Delete);
- Dialog Push Buttons validation (OK and Cancel);
- Expansion Button validation;
- MEW Delete Push Button;
- Push Button resulting in Dialog;
- Refresh Button Test;
- Date validation;
- Hot Key tests;
- Mandatory Fields validation;
- Name and Address Text fields;
- Decimal Number Tests;
- Whole Number Checks;
- Time Format validation;

- Text - Limited Length Free Format.

Navigational (GUI).

- Radio Button with Dialog Expansion;
- Icon Menu Checks;
- Icon Menu Navigation;
- Icon Menu Tabbing;
- General Menu Tests;
- Screen Menu Tests;
- Return Key Test;
- Tabbing.

Cosmetic (GUI).

- Display only fields;
- Protected Text Test;
- Screen Appearance;
- Error Messages;
- Spelling;
- Cursor Appearance.

Additional tests should be defined as appropriate to the individual application, for example:

- Specific tests defined to validate areas of significant importance. These tests must be defined explicitly, and with expected results.
- Field dependency tests, where differing values in one field affect the existence of, or values in other fields.
- Calculations. Where calculations are required, ensure their accuracy. For example, ensure that ‘Age Next Birthday’ is correctly calculated from ‘Date of Birth’ and Current Date. Protection against invalid data being present in any of the source fields used for calculations etc.

2.4 Who performs Unit Testing.

Unit Testing should be performed by the programmer responsible for the program or module. Using the Unit Test Check list, the programmer should run through the appropriate tests relevant and test his or her own work. This should be done prior to releasing the program or module for any further testing. It must be emphasised that the programmer has a **responsibility** for generating **quality** programs, and therefore must take on some of the testing effort. It is **not** acceptable for a module to go to the next stage of testing without this exercise being performed.

The results of the Unit Test must be recorded, and filed.

Optionally, the Unit Test may be performed by any person assigned to the task by the Project Manager, and this person may be anybody on the project, or any other project, although this is to be avoided wherever possible. As the Unit Test is a significant check on module functionality, it does require a familiarity with the system being developed.

3. Link Testing.

3.1 Scope and Purpose.

The purpose of Link Testing is to ensure that:

- ∗ Information entered, processed and stored in one module is successfully retrieved and manipulated by another module, in the correct format;
- ∗ Navigation between modules is correct, both forward and backward;
- ∗ Ticking and greying/un-greying of Menu Options is performed correctly.

Link Testing is the process of ensuring that different modules communicate with each other, and pass data between each other correctly, via the system and the database. Data entered in one module will inevitably be required by another, and it is this interaction between these modules that must be fully tested as well as the business logic as defined in the functional specifications.

3.2 Procedures.

The Test Plans for the Link Testing stage should have been created following the development of the Functional Specification. It is the responsibility of the Project Manager to ensure that these are available prior to being required.

Link Testing will be conducted by the Test Team allocated to the Project. The Test Team should consist of a group of independent individuals, dissociated from the development team. It is accepted, however, that this will not always be the case. The important aspect to ensure in this, and all future stages of Testing, is that the originating programmer **must not** be involved. If the programmer is involved in testing his/her own programs, then assumptions will be made with regard to the functionality. This is not the purpose of Testing.

As Link Testing proceeds, any problems identified **must** be recorded on a Fault Report Form. Refer to the Fault Reporting Procedure for more information. Note: It is recommended that **only one** Fault is recorded per Fault Report Form. The reasons for this are:

- ∗ It is easier to respond to one problem at a time;
- ∗ It is easier to analyse, statistically, the volume of problems reported on a system - more of this in section 8;
- ∗ Not all reported Faults are program Faults. Some may be caused by the Operating System, Data, Programming Language or Development Tool, Operator error, misinterpretation of the specification, or a fault within the test specification. In cases such as these, it may be possible to reject a reported Fault, but not if a genuine program bug is recorded on the same Fault Report form.

As each Fault Report is completed, it should be recorded on to the Fault Reporting System.

When Link Testing has been successfully completed, the Project Test Completion Form (Link Testing Section) must be signed, and dated, by the

Team Leader of the development team or the Test Team (see Appendix A). The module may now be released to the next stage in the Testing Cycle. The Link Test authorisation should be handed to the Project Manager to confirm completion, and it will then be filed in the Project Folder.

3.3 Who performs Link Testing.

Link Testing **must** be performed by any member of the project team who is familiar with the integration requirements of the programs or modules being tested. This will therefore vary from project to project, and also within the project, depending upon the system requirements. Link Testing is best performed by the programmer(s) responsible for the modules or programs being integrated. The project manager should ensure that the relevant resources are allocated to each Link Test.

4. System Testing.

4.1 Scope and Purpose.

The purpose of System Testing is to ensure that:

- Information entered, and therefore stored, in one module is successfully retrieved and manipulated by another module, in the correct format;
- Navigation between modules is correct, both forward and backward;
- Ticking and greying/un-greying of Menu Options is performed correctly.

System Testing is the process of ensuring that different modules communicate with each other, and pass data between each other correctly, via the system and the database. Data entered in one module will inevitably be required by another, and it is this interaction between these modules that must be fully tested as well as the business logic as defined in the functional specifications.

This is very similar in practice to Link Testing, but the purpose of System Testing is to ensure that the entire system works as a cohesive unit. For this reason, it is necessary to test the whole system from start to finish with a known set of data.

System Testing is performed with a number of Use Cases or Case Studies supplied by the Client. These should have been provided before System Testing commences, and should consist of working examples of how the system should operate, with input data, and output expected results. These Cases therefore represent the Test Plans for this stage of testing.

4.2 Procedures.

The Project Manager is responsible for ensuring that the Client provides Use Cases or Case Studies for System Testing, but it is the responsibility of the Client to ensure that they:

- Are accurate;
- Are correct;
- Fully test most, if not all, aspects of the system being developed.

On receipt of the Cases, they should be checked to ensure that they are accurate. This is important in order to avoid any confusion that may arise when identifying differences between the Cases and the System. Often, Faults are reported because the system produces a different result to that of the Use Case Study, and it is later discovered that the Case Study was wrong. Unfortunately, however, a lot of time has been spent in proving this. Time spent ‘up front’ in verifying the Case Studies will prevent many of these problems occurring.

The Project Manager should define how much verification should be performed, how detailed a check should be made, and which resources are allocated.

The Test Team is responsible for processing the Case Studies, and the Test Team Leader should allocate resources as appropriate to the task.

As System Testing proceeds, any problems identified **must** be recorded on a Fault Report Form. Refer to the Fault Reporting Procedure for more information. Note: It is recommended that **only one** Fault is recorded per Fault Report Form. The reasons for this are as stated in Link Testing.

As each Fault Report is completed, it should be recorded on to the Fault Reporting System.

When System Testing has been successfully completed, the Project Test Completion Form (System Testing Section) must be signed, and dated, by the Team Leader of the development team or the Test Team (see Appendix A). The module may now be released to the next stage in the Testing Cycle. The System Test authorisation should be handed to the Project Manager to confirm completion, and it will then be filed in the Project Folder.

4.3 Who performs System Testing.

As System Testing is concerned with the overall functionality of the system, the person(s) best suited to performing this role must have a clear and unambiguous understanding of what the system is trying to achieve. As Use Cases or Case Studies should have been provided by the client or business user for this phase of the test cycle, it is important to involve the user at this time.

Therefore, System Testing may be conducted by many people, including users, programmers, team leader, or a test team specifically created for the project.

5. Volume Testing.

5.1 Scope and Purpose.

The purpose of Volume Testing is to ensure that the system works correctly:

- With volumes of data comparable to those to be expected in a 'live' environment;
- On a hardware configuration identical to that planned for actual use.

Volume Testing addresses any outstanding issues that may have been specified in the Business Requirements Specification, such as usability, response times, data migration, and upload/download of data, as well as confirming the results of previous test stages.

5.2 Procedures.

Volume Testing may be conducted either at the Client/Business User production site, or at another location. It may also be performed either by Client Personnel, or by Development staff.

There are, therefore, no hard and fast rules governing Volume Testing. However the procedure for processing reported Faults remains the same. It is the responsibility of the Project Manager to ensure that, when Volume Testing is conducted at the Client site or by Client personnel, they are aware of, and conform to, the Fault Reporting Procedure.

As Volume Testing proceeds, any problems identified **must** be recorded on a Fault Report Form. Refer to the Fault Reporting Procedure for more information. Note: It is recommended that **only one** Fault is recorded per Fault Report Form. The reasons for this are as stated in Link Testing.

As each Fault Report is completed, it should be recorded on to the Fault Reporting System.

When Volume Testing has been successfully completed, the Project Test Completion Form (Volume Testing Section) must be signed, and dated, by the person responsible for Volume Testing (see Appendix A). As stated before, this may be a representative from the Client organisation or the Test Team Leader. If Volume Testing is undertaken by a Client representative, it is the responsibility of the Project Manager to ensure that the correct procedure is followed. The program may now be released to the next stage in the Testing Cycle. The Volume Test authorisation should be handed to the Project Manager to confirm completion, and it will then be filed in the Project Folder.

6. User Acceptance Testing.

6.1 Scope and Purpose.

User Acceptance Testing (UAT), by definition, is the process of conducting tests against the system by the Client organisation. It is therefore inappropriate to define the scope of User Testing, as this will be at the discretion of the Client/Business User. The main purpose, however, is to ensure that the Client is satisfied with the operation of the system, and to see it working at first hand, thus building Client confidence.

6.2 Procedures.

User Acceptance Testing may be conducted either at the Client site, or at another location. It may also be performed either by Client Personnel alone, or with the System Test team or/and the development team in attendance. Note: It is recommended that the development team maintain a presence during UAT, and it is the responsibility of the Project Manager to organise this.

There are, therefore, no hard and fast rules governing User Testing. However the procedure for processing reported Faults remains the same. It is the responsibility of the Project Manager to ensure that, when User Testing is conducted, the Client personnel are aware of, and conform to, the Fault Reporting Procedure.

As User Testing proceeds, any problems identified **must** be recorded on a Fault Report Form. Refer to the Fault Reporting Procedure for more information. Note: It is recommended that **only one** Fault is recorded per Fault Report Form. The reasons for this are as stated in Link Testing.

As each Fault Report is completed, it should be recorded on to the Fault Reporting System.

When User Testing has been successfully completed, the Project Test Completion Form (User Testing Section) must be signed, and dated, by the person from the Client organisation responsible for User Testing (see Appendix A). The User Test authorisation should be handed to the Team Leader to confirm completion, and it will then be filed in the Project Folder.

7. Regression Testing.

7.1 Scope and Purpose.

Regression Testing is the process whereby, in the event of detecting errors in the system (and fixing them), the entire Test Cycle is started again from the beginning (Unit Testing and onwards). The purpose of Regression Testing is to ensure that changes made to programs have not adversely affected the operation of the system in any way.

The most common cause of this is not fault fixes, but changes requested by the Client that were not incorporated into the Business Requirements Specification or Functional Specification. These changes will have been subject to Change Control, however the need to fully test the system again is essential.

It is not always necessary to perform Regression Testing. It is the nature of the changes effected to the system that define whether or not Regression Testing is appropriate. For example spelling mistakes would not constitute a reason for starting again, however the addition of a new data entry field on a screen and subsequent processing and printing of that field may force the need to review the entire Test process.

7.2 Procedures.

The Project Manager is responsible for making the decision to perform Regression Testing. For the purposes of this document it is suggested that Regression Testing is considered:

- After the implementation of a significant Change Request;
- After the fixing of a number of ‘A’ or ‘B’ class Faults (refer to the Fault Reporting Procedure for a definition of these);
- Prior to a new release of the software to the Client.

As Regression Testing is effectively re-running the previous Tests (Unit Testing to User Testing), the procedures as stated in earlier Sections of this document apply.

8. Statistics and Information Gathering.

As stated before, in the Introduction, Testing is a fundamental and very important part of the Software Development Life Cycle. It is also one of the areas in which we can always improve - a bug-free system is an impossible dream, but one we should always strive to achieve. Fortunately, the means exists for us to be able to measure improvements in software development. Testing is all about finding bugs! Proper testing gives us the opportunity to help ourselves to improve the way we work, and the type of work we do - no-one likes bug-fixing and maintenance. Therefore the better we become at development then, hopefully, the less time needs to be spent on maintenance.

By close adherence to the Test Cycle we can begin to gather information about those areas of system development that cause the greatest number of problems. Using this information, we can begin to focus on those areas in future systems developments, and thereby reduce the level of maintenance required.

Statistics are used to identify where we can improve the service offered to our users. The Test Cycle must ensure that Faults are reported accurately and comprehensively. We can then analyse the reported Faults in a number of ways to ensure that we do not make the same mistakes again. It will also provide us with a continuous, visible, indication of how we are improving. A variety of different tools are available for this purpose such as Imago T-Plan, PVCS Tracker, SQA Manager in conjunction with mail systems such as MS Outlook or Exchange.

9. Appendix A - Project Test Completion Form.

Stage	Authorisation	Date
Unit Test		
Link Test		
System Test		
Volume Test		
User Test		

This form must be filed in the Project Folder.

10. Appendix B - Structure of a Test Plan.

10.1 Introduction.

10.1.1 Glossary of Terms.

10.1.2 References.

List all related documents, including specifications, design documents, etc., pertaining to the system under test.

10.1.3 Changes to the document.

Outline changes from the previous issue, and any changes anticipated in the future.

10.1.4 Distribution.

Refer to the distribution process and distribution list (identify who will receive copies of this plan).

10.2 Test Objectives.

10.2.1 Overview of Product.

A description of the product to be tested, briefly, to provide relevant context for the reader. Actual documents will have been referenced in Section 1.2.

10.2.2 Identification of assemblages to be tested.

Sufficient detail should be given to identify what is to be tested (modules, assemblies, sub-systems, etc.).

10.2.3 Purpose(s) of Test.

Describe here the main objectives of the tests. What will the test achieve? Why is it being carried out?

10.2.4 Deliverables.

Describe both the end-product of the testing and the documentation of the testing.

10.3 Test Methods.

10.3.1 Strategy.

Describe the overall approach to the testing (top-down, bottom-up, mixed, incremental, structural, functional, etc.).

10.3.2 Tests to be carried out.

Provide more detail for each of the tests, or groups of tests, to be carried out (test identity, purpose, method, etc.). A separate specification will be required for each. These may be included in this document, or may be referenced here. A tabular presentation may be suitable.

10.3.3 Order of Testing.

List any requirements for any tests to be carried out in a particular order, and the rationale behind them.

10.3.4 Environment, Tools and Constraints.

Describe the environment for the tests, any tools that will be used, and any other constraints.

10.4 Entry Criteria (Input Data).

Describe the pre-requisites that must be satisfied before the testing can commence:

- ∗ Documentation that is needed.
- ∗ Previous activities that must have been completed.
- ∗ Hardware, software, test tools, harnesses, ‘scaffold’ code, etc.

10.5 Exit Criteria (Expected Results).

Define the criteria to be satisfied before the testing can be considered to be completed successfully. Wherever possible, the criteria should be stated in quantitative terms, and be economically measurable.

10.6 Schedule.

Provide a calendar of dates for the testing programme or, preferably, a Bar Chart/Gantt Chart. The schedule should also include a list of key milestones for the programme. Where the programme contains significant dependencies, a PERT network chart may be appropriate.

10.7 Reporting Mechanisms.

10.7.1 Management Reporting of Progress.

Describe how testing progress will be reported at the Project Management level.

10.7.2 Technical Progress and Problems.

Describe the arrangements for reporting technical progress and how problems will be reported and handled. Include, where necessary, liaison with the software developers unless this is covered by the Change Control procedure.

10.7.3 Recording of Test Results.

Describe where, and how, test results will be recorded, and the level of detail. It may be sufficient to reference an appropriate standard or procedure.

10.8 Version Plan.

10.8.1 Version Control Strategy.

Describe the overall strategy to be applied in the generation of new versions of the software during the course of the testing.

10.8.2 Conditions for Software Release.

What will be the conditions to be applied in the generation of a new version of the software?

10.8.3 Mechanisms for Configuration Management and Change Control.

Describe, or reference, the procedures for configuration management and change control.

10.9 Dependencies.

10.9.1 Internal Dependencies.

List any dependencies within the organisation.

10.9.2 External Dependencies.

List any dependencies external to the organisation.

10.10 Resource Requirements.

Record all the resources required for the testing.

11. Appendix C - Program Unit Test Check list (Gui).

The following ‘proforma’ is a check list of standard tests to be performed on a form or series of forms making up a ‘Unit’ of work within a project. Clearly, not all tests will be applicable to all forms (for example, if a form does not contain a ‘Date’ field, then the Date Validation Test does not need to be performed). The person responsible for the development of a project’s Test Plan should extract and modify (as appropriate) those tests applicable to each form.

TEST	TEST DESCRIPTION
Functional	
1	Box - Check
2	- Combo
3	- Drop Down
4	- MEW (Multiple Entity Windows)
5	- Multi-Select
6	- Single-Select
7	Button - DELETE
8	- Dialog Push Button (<--, New, -->, Delete)
9	- Dialog Push Button (Delete)
10	- Dialog Push Button (OK, Cancel)
11	- Expansion Button
12	- MEW Dialog Push Button (Delete)
13	- Push Button resulting in Dialog
14	- Refresh Button
15	Date including Millenium 2000 compliance)
16	Hot Keys
17	Mandatory on ‘Yes’ response
18	Name/Address
19	Number – Decimal
20	- Whole
21	Time (24 hour format)
22	Text - Limited Length Free Format
23	Data storage and display
Navigational	
24	Radio Button with Dialog Expansion
25	Icon - Menu
26	- Navigation
27	- Tabbing
28	Menu - Functional Area (General)
29	- Functional Area (Screen Specific)
30	Return Key
31	Tabbing
Cosmetic	
32	Display only fields
33	Protected Text
34	Screen Appearance
35	Error Messages
36	Spelling
37	Cursor Appearance

11.1 Functional.

11.1.1 Check Box.

Requirement: **Check Box validation**		
No	**Test Script**	**Expected Results**
1	Click on the Check Box	The Check Box is selected. A cross (5) appears in the box
2	Press Tab	The cursor positions itself on the next input field or push button. The Check Box is still selected.
3	Return to the Check Box. Click on the Check Box again.	The Check Box is deselected. The cross (5) disappears from the box.
4	Press Tab	The cursor positions itself on the next input field or push button. The Check is no longer selected.

11.1.2 Combo Box.

Requirement: **Combo Box validation**		
No	**Test Script**	**Expected Results**
1	Click on the Down Arrow	The correct Combo List is displayed, in the correct sequence, and the whole text string for each entry is visible (unless otherwise specified).
2	Select one of the options in the List	The selected option will be displayed in the main box.
3	Type in text until a click is heard	The typed text will replace the selected option in the main box.
4	Count the number of characters, including punctuation and spaces	The click occurs when attempting to type past the maximum length (as specified) and not before.
5	Click on the Down Arrow	The correct Combo List is displayed, in the correct sequence.
6	Select one of the options	The selected option will replace the typed text in the main box.
Note: If the Combo Box controls additional fields' visibility, these fields will be invisible on first entry to the screen. They will be made visible when visibility conditions are met, and will return to being invisible when the conditions are not met. Options should be chosen such that they fully test the visibility conditions.		

11.1.3 Drop Down List Box.

Requirement: Drop Down Box validation		
No	**Test Script**	**Expected Results**
1	Click on the Down Arrow	The correct Drop Down List is displayed, in the correct sequence, and the whole text string for each entry is visible (unless otherwise specified).
2	Select one of the options	The selected option will be displayed in the main box.
3	Repeat Steps 1 and 2 several times	When another option is selected, it replaces the existing contents of the main box.
Note: If the Drop Down List Box controls additional fields' visibility, these fields will be invisible on entry to the screen. They will be made visible when visibility conditions are met, and will return to being invisible when the conditions are not met. Options should be chosen such that they fully test the visibility conditions.		

11.1.4 MEW (Multiple Entity Windows) Format.

Requirement:	**MEW Format validation**	
No	**Test Script**	**Expected Results**
1	Select 'New' (double click)	Cursor moves to data entry fields with blanks or defaults.
2	Enter data and commit	New line of data appears in the MEW.
3	Select an existing entry	Focus moves to data entry fields with existing details.
4	Amend data and commit	Existing MEW entry updated.
5	Repeat steps 1 and 2 until there are at least two more entries in the MEW than will fit the display area	Standard scroll bar is positioned at the right hand side of the MEW.
6	Use the cursor to scroll up and down the MEW	Different areas of the MEW are visible, selected entry is unchanged.
7	Use the Tab key to position the cursor at the MEW	Different areas of the MEW are visible, selected entry changes.
8	Use the Up and Down Arrows to scroll up and down the MEW	Different areas of the MEW are visible, selected entry is unchanged.

11.1.5 Multi-Select List Box.

Requirement:	**Multi-Select Box validation**	
No	**Test Script**	**Expected Results**
1	Examine the list displayed	The correct List is displayed, and in the correct sequence.
2	Click on one of the options	The selected option is displayed.
3	Click on another option	When another option is selected, it is highlighted and the originally selected option remains highlighted.
4	Click on the same option	When an already selected option is clicked on, it is deselected.

11.1.6 Single-Select List Box.

Requirement:	**Single-Select Box validation**	
No	**Test Script**	**Expected Results**
1	Examine the list displayed	The correct list is displayed, and in the correct sequence.
2	Click on one of the options	The selected option is highlighted.
3	Click on another option	When another option is selected, it is highlighted and the originally selected option is de-highlighted.
4	Click on the same option	The selected option remains selected.

11.1.7 Delete Push Button From Named List.

Requirement:	**Delete Push Button Function validation**	
No	**Test Script**	**Expected Results**
1	With the named criterion not appearing in the List description, select the named criterion from the Criterion field	The named Dialogue appears with no Delete Push Button.
2	Press the Cancel Button	Control is returned to the Named List Dialogue.
3	With the named criterion appearing in the List description, select the named criterion	The Dialogue is displayed with data belonging to the selected item and the Delete Button is present.
4	From a Dialogue with existing data, press Delete	Verification message is displayed.
5	Press the Cancel Button	Control is returned to the named Dialogue - data is unchanged
6	Press Delete	Verification message is displayed.
7	Press the OK Button	Control is returned to the named List Dialogue - the correct line(s) of the List description has/have been deleted.

11.1.8 Dialogue Push Buttons Test (<--, New, -->, Delete)

Requirement:	**Dialogue Push Buttons Function validation**	
No	**Test Script**	**Expected Results**
1	Press ‘<--’	Message ‘Already positioned on 1st record’ is displayed.
2	Press ‘-->‘	Message ‘Already positioned on last record’ is displayed.
3	Press ‘New’	A second, blank, Dialogue screen should be displayed.
4	Enter data and press ‘<--’	The first Dialogue screen is displayed with the data intact.
5	Press ‘OK’	Control is returned to the invoking form.
6	Return to the Dialogue (i.e. press ‘Expansion’ Button or select ‘Yes’)	The first Dialogue screen is displayed with the data intact.
7	Press ‘-->‘	The second Dialogue is displayed with the data intact.
8	Press ‘Delete’	The first Dialogue screen is redisplayed with the data intact.
9	Press ‘-->‘	Message ‘Already positioned on last record’ is displayed.

11.1.9 Dialogue Push Button (Delete).

Requirement:	**Dialogue Push Button Function validation**	
No	**Test Script**	**Expected Results**
1	Invoke a Dialogue containing data	The Dialogue is displayed, overlaying the main form.
2	Press the Delete Button	A confirmation message is displayed, and on ‘OK’ the original form is now fully visible.
3	Re-invoke the same Dialogue	The Dialogue is displayed, overlaying the main form. No data is displayed.

11.1.10 Dialogue Push Button (OK and CANCEL).

Requirement:	**Buttons work as designated**	
No	**Test Script**	**Expected Results**
1	Without entering data, press ‘Ok’	Control is returned to invoking form - no data is added.
2	Without entering data, press ‘Cancel’	Control is returned to invoking form - no data is added.
3	Enter data (including mandatory data), and press ‘Ok’	Control is returned to invoking form - data is added.
4	Enter data (including mandatory data), and press ‘Cancel’	Control is returned to invoking form - no data is added.
5	Enter data (not including all mandatory data) and press ‘Cancel’. Note: This test is not appropriate if no data is mandatory	Control is returned to invoking form - no data is added.

11.1.11 Drop Down Push Button Dialogue.

Requirement: Drop Down Push Button Dialogue validation

No	Test Script	Expected Results
1	Select the option being tested	The appropriate, uncompleted, Dialogue is displayed.
2	Select ‘Ok’ without entering data	Error displayed because first field is mandatory.
3	Enter data and select the ‘Refresh’ Button	The Dialogue reverts to the original, uncompleted, Dialogue.
4	Enter data and select the ‘Cancel’ Button	Control is returned to the original screen, with no option selected.
5	Select the option being tested	The appropriate, uncompleted, Dialogue is displayed.
6	Enter data and select the ‘Ok’ Button	Control is returned to the original screen with the correct option selected and an expansion Button present.
7	Click on the expansion Button	The Dialogue is displayed with the data previously entered.
8	Change data and select the ‘Refresh’ Button	The Dialogue reverts to the data previously entered.
9	Change data and select the ‘Cancel’ Button	Control is returned to the original screen with the correct option selected and an expansion Button present.
10	Click on the ‘Expansion’ Button	The Dialogue is displayed with the original data, not the changes.
11	Change data and select the ‘Ok’ Button	Control is returned to the original screen.
12	Click on the ‘Expansion’ Button	The Dialogue is displayed with the latest changed data.

11.1.12 MEW Dialogue Push Button (Delete).

Requirement:	**MEW Delete Function validation**	
No	**Test Script**	**Expected Results**
1	Select 'New' on the MEW	There is no 'Delete' Push Button on the Dialogue screen.
2	Press the 'Cancel' Button	Control is returned to the Dialogue.
3	Select another line of the MEW	The Dialogue is displayed with data belonging to the selected item and the Delete Button is present.
4	From a Dialogue with existing data, press 'Delete'	Verification message is displayed.
5	Press the 'Cancel' Button	Control is returned to the Dialogue - data is unchanged.
6	Press 'Delete'	Verification message is displayed.
7	Press the 'Ok' Button	Control is returned to the invoking form - the correct line of the MEW has been deleted.

11.1.13 Push Button Resulting in Dialogue.

Requirement:	**Push Button Test resulting in Dialogue Form**	
No	**Test Script**	**Expected Results**
1	Click on the Push Button (screen)	The appropriate Dialogue is invoked and overlaid on the original screen.
2	Click on 'Ok' Push Button (Dialogue)	The original screen or Dialogue is re-displayed.

11.1.14 Refresh Button Test.

Requirement:	**Refresh Function validation**	
No	**Test Script**	**Expected Results**
1	Enter data into all fields on the screen, ensuring that all fields containing default entries are changed. Press the REFRESH Button	The same screen is re-displayed. All data entered is lost. All fields should contain the original default values.
2	On a screen where data has already been entered and saved, change data in all fields. Press the REFRESH Button	The same screen is re-displayed. All data changes are lost. All fields should contain the original values.

11.1.15 Date.

Requirement:	Correct Date validation	
No	**Test Script**	**Expected Results**
1	ABCDE	Error condition, or field blanks out.
2	30/02/94	Error condition, or field blanks out.
3	30 Jan 94	30/01/1994 is displayed.
4	MIN Date - 1 day	Error condition.
5	MAX Date + 1 day	Error condition.
6	MIN Date	Minimum Date is displayed.
7	MAX Date	Maximum Date is displayed.
8	30.04.1994	30/04/1994 is displayed.
9	30/4/94	30/04/1994 is displayed.
10	9/9/99	09/09/1999 is displayed.
11	01/1/00	01/01/2000 is displayed.
12	30/9/2000	30/09/2000 is displayed.
13	30/9/2049	30/09/2049 is displayed.
14	29/2/2000	29/02/2000 is displayed (year 2000 is a leap year).
15	29/2/00	29/02/2000 is displayed.

11.1.16 Hot Key Test.

Requirement:	Hot Key Function validation	
No	**Test Script**	**Expected Results**
1	For each Hot Key listed, press 'Alt' and the letter or symbol given	Should be identical to clicking on the relevant Menu Title or Push Button with the mouse.
2	Check for duplicate Hot Keys	No duplicated Hot Keys should exist.

11.1.17 Mandatory on 'Yes' Response.

Requirement:	Mandatory Field validation	
No	**Test Script**	**Expected Results**
1	Respond 'Yes' to the relevant question, and leave all mandatory fields blank	Error condition is produced for each mandatory field in error, in the order given. The screen cannot be left until all mandatory fields have been completed whilst 'Yes' has been selected.
2	Select an Icon to move to another screen	Error condition produced for each mandatory field in error, in the order given.
3	When error condition met, enter valid data in the first mandatory field and select the Icon again	Error condition produced for next mandatory field in error, in the order given.
4	Repeat step 3 until no more errors are notified	Correct screen is invoked.
5	Return to original screen, and delete entries in all mandatory fields. Repeat steps 2 - 4	As steps 2 - 4.
6	Type spaces into all mandatory fields and repeat steps 2 - 4	As steps 2 - 4.

11.1.18 Name and Address Text.

Requirement:	**Field Format and Length validation**	
No	**Test Script**	**Expected Results**
1	Type in text less than the maximum length, including upper and lower case alphabetic and numeric characters, punctuation, and spaces. Ensure that the first character is lower case	Text appears as typed. The size of the box matches the standard template for a field of that length.
2	Tab forwards	Text remains unchanged by tabbing, except for the first character of each word, which becomes upper case.
3	Tab backwards	Text remains unchanged by tabbing, except for the first character of each word, which becomes upper case.
4	Type in more text until a click is heard	Text scrolls left, depending on the characters typed.
5	Count the number of characters in the field, including punctuation and spaces	The click occurs when attempting to type past the maximum length defined, and not before or after.

11.1.19 Decimal Number.

Requirement:	**Decimal Number validation**	
No	**Test Script**	**Expected Results**
1	ABC	Blanks out or does not allow characters.
2	30/4/94	30.0000 displayed to correct decimal places.
3	9.9	9.90000 displayed to correct decimal places.
4	.9	0.90000 displayed to correct decimal places.
5	9999999 until click is heard	Field size is correct.
6	Minimum specified value - 0.1	Error condition.
7	Maximum specified value + 0.1	Error condition.
8	Minimum specified value	Minimum is displayed.
9	Maximum specified value	Maximum is displayed.

11.1.20 Whole Number Test.

Requirement:	**Field Format and Length validation**	
No	**Test Script**	**Expected Results**
1	ABC	Blanks out or does not allow characters.
2	30/4/93	30 displayed, or does not allow '/' to be typed.
3	9.9	10 displayed, or does not allow '.' to be typed.
4	9999999 until click is heard	Field size is correct.
5	009	9 is displayed.
6	Minimum specified value - 1	Error condition.
7	Maximum specified value + 1	Error condition.
8	Minimum specified value	Minimum is displayed.
9	Maximum specified value	Maximum is displayed.
Notes: For each of the above test scripts, type in the data and press Tab. For short numbers, ignore the tests that are not possible.		

11.1.21 Time Format.

Requirement:	**Standard Time validation**	
No	**Test Script**	**Expected Results**
1	ABCDE	Blanks out.
2	24:01	Blanks out.
3	00:00	00:00 is displayed.

11.1.22 Text - Limited Length Free Format.

Requirement:	Field Format and Length validation	
No	**Test Script**	**Expected Results**
1	Type in text less than the maximum length, including upper and lower case alphabetic & numeric characters, spaces, & punctuation	Text appears as typed. The size of the text box matches the standard template for a field of that length.
2	Tab forwards	Text remains unchanged by Tabbing.
3	Tab backwards	Text remains unchanged by Tabbing.
4	Type in text until a click is heard	Text scrolls left, depending on the characters typed.
5	Count the number of characters in the field, including spaces and punctuation	The click occurs when attempting to type past the maximum length, and not before or after.

11.1.23 Data Storage and Display.

No	Test Script	Expected Results
1	Enter data and select appropriate Save Button or Menu	Contents of appropriate file(s) or DB tables updated with the data entered
2	Return to the form and re-display data	Screen displays the data correctly, in the format specified
3	Enter data and select appropriate Cancel Button or Menu	Contents of appropriate file(s) have not been updated

11.2 Navigational.

11.2.1 Radio Button with Dialogue Expansion.

Requirement:	**Radio Button validation**	
No	**Test Script**	**Expected Results**
1	Forward Tab to appropriate Radio Button	Forward Tabbing highlights, but does not select the ‘Yes’ Button.
2	Press the right Arrow	The ‘No’ Button is selected.
3	Press the left Arrow	The appropriate uncompleted Dialogue is displayed.
4	Select ‘Ok’ without entering data	Error displayed because first field is mandatory.
5	Enter data and select the ‘Refresh’ Button	The Dialogue reverts to the original uncompleted Dialogue.
6	Enter data and select the ‘Cancel’ Button	Control is returned to the original screen with no Buttons selected.
7	Click on the ‘Yes’ Button	The appropriate uncompleted Dialogue is displayed.
8	Enter data and select the ‘Ok’ Button	Control is returned to the original screen with the ‘Yes’ Button selected and an ‘Expansion’ Button present.
9	Press Tab, then the Space Bar	The Dialogue is displayed with the data previously entered.
10	Enter data and select the ‘Refresh’ Button	The Dialogue reverts to the data previously entered.
11	Enter data and select the ‘Cancel’ Button	Control is returned to the original screen with the ‘Yes’ Button selected and an ‘Expansion’ Button present.
12	Click on the ‘Expansion’ Button	The Dialogue is displayed with the original data, not the changes.
13	Select the ‘Ok’ Button	Control is returned to the original screen.
14	Select the ‘No’ Button	Warning condition is met.
15	Press ‘Esc’	Control is returned to the original screen with the ‘Yes’ Button selected and an ‘Expansion’ Button present.
16	Select the ‘No’ Button	Warning condition is met.
17	Select the ‘Ok’ Button	Control is returned to the original screen with the ‘No’ Button selected and the ‘Expansion’ Button removed.
18	Press ‘Del’	The ‘No’ Button is de-selected and the ‘Yes’ Button remains unselected.
19	Select ‘Yes’	The appropriate uncompleted Dialogue is displayed.

20	Enter data and select ‘Ok’	Control is returned to the original screen with the ‘Yes’ Button selected and an ‘Expansion’ Button present.
21	Press ‘Del’	Warning condition is met.
22	Press ‘Esc’	Control is returned to the original screen with the ‘Yes’ Button selected and an ‘Expansion’ Button present.
23	Press ‘Del’	Warning condition is met.
24	Select the ‘Ok’ Button	The ‘Yes’ Button is de-selected, the ‘No’ Button remains unselected, and the ‘Expansion’ Button is removed.

11.2.2 Icon Menu.

Requirement:	**Icon Menu validation**	
No	**Test Script**	**Expected Results**
1	From the Menu being tested, select the Icon Menu by clicking on the Menu Bar	Icon Menu drops down and contains the correct entries in the correct order. Correct letters are underlined as 'Hot Keys'.
2	De-select the Icon Menu by clicking on the Menu Bar	Icon Menu disappears.
3	Re-select the Icon Menu, and select a Menu Item by clicking with the mouse	Correct screen is invoked for the selected Menu Item.

11.2.3 Icon Navigation Test.

Requirement:	**Icon Navigation**	
No	**Test Script**	**Expected Results**
1	For each of the Icons at the base of the screen: click on the Icon to jump to another screen	The screen represented by the Icon is displayed.
2	For each of the Icons at the base of the screen, return to the original screen	The original screen is returned to.

11.2.4 Icon Tabbing.

Requirement:	**System to be usable without the Mouse**	
No	**Test Script**	**Expected Results**
1	Press PF6	Title Bar de-highlighted.
2	Press forward Tab Button once for each Icon on the Icon Strip	Icons highlighted in turn from left to right, wrapping round (so that leftmost Icon is highlighted immediately after the furthest right).
3	Press backward Tab Button once for each Icon on the Icon Strip	Icons highlighted in turn from right to left, wrapping round (so that rightmost Icon is highlighted immediately after the furthest left).
4	Press PF6	Title Bar is re-highlighted.

11.2.5 Functional Area Menu (General).

Requirement:	**Functional Area Menu Test**	
No	**Test Script**	**Expected Results**
1	From any screen within the Functional Area, select the Area Menu by clicking on the Menu Bar	Menu drops down and contains the correct entries in the correct order. Correct letters are underlined as 'Hot Keys'.
2	Select each Menu Entry in turn by clicking with the mouse	'Default' screen for the selected Functional Area is invoked.
3	Select each Menu Entry in turn by typing the 'Hot Key'	'Default' screen for the selected Functional Area is invoked.

11.2.6 Functional Area Menu (Screen).

Requirement:	**Menu functions correctly**	
No	**Test Script**	**Expected Results**
1	From the screen being tested, select the Functional Area Menu by clicking on the Menu Bar	Menu drops down and contains the correct entries in the correct order. Correct letters are underlined as 'Hot Keys'.
2	De-select the Menu by clicking on the Menu Bar	Functional Area Menu disappears.
3	Re-select the Menu, and select a Menu Item by clicking with the mouse	Correct screen is invoked for the selected Menu Item.

11.2.7 Return Key Test.

Requirement:	**Return Key Function**	
No	**Test Script**	**Expected Results**
1	On the screen being tested, select PF6 to move on to the Icon Strip	Icon Bar is highlighted.
2	Tab to the relevant Icon	Icon is highlighted.
3	Press the Return Key	Correct screen is invoked for the selected Icon.

11.2.8 Tabbing Test.

Requirement:	**Tabbing is in correct order**	
No	**Test Script**	**Expected Results**
1	Select the screen or Dialogue to be used	The first field in the list will be highlighted on first entry to the screen.
2	Press TAB repeatedly without entering any data, until the initial position is returned to	Tabbing will highlight each field or Icon in the order specified. Invisible fields will not be displayed. For Radio Buttons, the first one will be highlighted when none in the group are selected, otherwise the selected one will be highlighted.
3	Press Reverse TAB repeatedly without entering any data, until the initial position is returned to	Reverse Tabbing will highlight each field or Icon in the reverse order to that specified. Invisible fields will not be displayed. For Radio Buttons, the last one will be highlighted when none in the group are selected, otherwise the selected one will be highlighted.
4	Set the conditions which make the invisible fields visible	All fields will be visible.
5	Press TAB repeatedly until the initial position is returned to	Tabbing will highlight each field or Icon in the order specified, including fields previously invisible.
6	Press Reverse TAB repeatedly until the initial position is returned to	Reverse Tabbing will highlight each field or Icon in the reverse order to that specified, including fields previously invisible.

11.3 Cosmetic.

11.3.1 Display Only.

Requirement:	**Display Only validation**	
No	**Test Script**	**Expected Results**
1	Attempt to select the Display Only field by tabbing and clicking using the mouse	The cursor should not enter this field. It must not be possible to enter data in this field.

11.3.2 Protected Text Test.

Requirement:	Protected Text validation	
No	**Test Script**	**Expected Results**
1	Use the mouse to position the cursor on the protected field	Cursor does not change shape.
2	Click the mouse button	No effect.
3	Attempt to type in data	Contents of field cannot be over typed.

11.3.3 Screen Appearance.

No	Test Script	Expected Results
1	Check screen conforms to the screen dump in the specification	Screen matches, exactly, the screen dump in the specification
2	Ensure layout of screen is consistent	Screen is consistent with design standards as specified
3	Check positioning of objects	Screen Objects are positioned correctly, and consistently

11.3.4 Error Messages.

No	Test Script	Expected Results
1	Enter invalid data in all fields	Check that the error messages displayed are: meaningful; correct; explanatory; consistent; and appropriate to the type of error (Stop or Warning)
2		Ensure navigation to and from the Error Dialog Box is correct
3		On Warnings, check functioning of both OK and Cancel Buttons

11.3.5 Spelling.

No	Test Script	Expected Results
1	Check spelling of all labels, field headings, screen headings, and Objects	Spelling is correct

11.3.6 Cursor Appearance.

No	Test Script	Expected Results
1	Position cursor over data entry fields	Cursor shape changes to ‘I-beam’
2	Position cursor over ‘protected’ fields	Cursor remains as, or changes to, pointer format
3	Navigate to another Form	Cursor changes to an ‘hourglass’ (6) format, and remains until the next form is fully displayed