

**Office of the
Government Chief Information Officer**

**GUIDELINES FOR
APPLICATION SOFTWARE TESTING**

[G20]

Version : 1.4

Sep 2005

©The Government of the Hong Kong Special Administrative Region

The contents of this document remain the property of and may not be reproduced in whole or in part without express permission of the Government of the HKSAR

Table of Contents

1. PURPOSE.....	1-1
2. SCOPE 2-1	
3. REFERENCES	3-1
3.1 STANDARDS	3-1
3.2 OTHER REFERENCES	3-1
4. DEFINITIONS AND CONVENTIONS	4-1
4.1 DEFINITIONS	4-1
4.2 CONVENTIONS	4-1
5. OVERVIEW	5-1
5.1 PROJECT ORGANISATION	5-1
5.1.1 Test Control Sub-Committee	5-1
5.1.2 Independent Test Group	5-1
5.2 TESTING ACTIVITIES.....	5-1
5.3 TESTING DOCUMENTATION.....	5-2
5.4 TEST PLANNING & CONTROL	5-2
5.4.1 Progress Control	5-2
5.4.2 Quality Control / Assurance	5-2
5.4.3 Resource Estimation	5-2
6. GENERAL CONCEPTS OF TESTING.....	6-1
6.1 TESTING OBJECTIVES	6-1
6.2 TESTING STRATEGY	6-1
6.3 LEVELS OF TESTINGS	6-2
6.4 GENERAL TESTING PRINCIPLES.....	6-3
6.5 COMPLEMENTARY REVIEWS.....	6-4
7. LEVELS OF TESTING.....	7-1
7.1 UNIT TESTING	7-1
7.1.1 Scope of Testing	7-1
7.1.2 Activities, Documentation and Parties Involved	7-1
7.1.3 Practical Guidelines	7-1
7.2 LINK TESTING	7-2
7.2.1 Scope of Testing	7-2
7.2.2 Activities, Documentation and Parties Involved	7-2
7.2.3 Practical Guidelines	7-3
7.3 FUNCTION TESTING	7-4
7.3.1 Scope of Testing	7-4
7.3.2 Activities, Documentation and Parties Involved	7-4
7.3.3 Practical Guidelines	7-4
7.4 SYSTEM TESTING.....	7-5

7.4.1 Scope of Testing	7-5
7.4.2 Activities, Documentation and Parties Involved	7-5
7.4.3 Practical Guidelines	7-5
7.5 ACCEPTANCE TESTING.....	7-7
7.5.1 Scope of Testing	7-7
7.5.2 Activities, Documentation and Parties Involved	7-7
7.5.3 Practical Guidelines	7-7
8. TEST DOCUMENTATION	8-1
8.1 INTRODUCTION	8-1
8.2 TEST PLAN	8-2
8.2.1 Purpose of Document	8-2
8.2.2 Outline of Document	8-2
8.3 TEST SPECIFICATION	8-4
8.3.1 Purpose of Document	8-4
8.3.2 Outline of Document	8-4
8.4 TEST INCIDENT REPORT.....	8-5
8.4.1 Purpose of Document	8-5
8.4.2 Outline of Document	8-5
8.5 TEST PROGRESS REPORT	8-7
8.5.1 Purpose of Document	8-7
8.5.2 Terminology	8-7
8.5.3 Outline of Document	8-8
8.6 TEST SUMMARY REPORT	8-9
8.6.1 Purpose of Document	8-9
8.6.2 Outline of Document	8-9
9. TEST PLANNING AND CONTROL	9-1
9.1 TEST PLANNING	9-1
9.2 TEST GROUP	9-1
9.3 INDEPENDENT TEST CONTROL SUB-COMMITTEE	9-2
9.3.1 Introduction	9-2
9.3.2 Terms of reference of the Sub-Committee	9-2
9.3.3 Proposed membership of the sub-committee.....	9-2
9.3.4 Project Structure	9-3
10. AUTOMATED TOOLS FOR TESTING	10-1
10.1 INTRODUCTION	10-1
10.2 TEST DATA GENERATOR.....	10-1
10.3 STATIC ANALYSERS	10-1
10.4 DYNAMIC ANALYSERS.....	10-1
11. SUMMARY	11-1

11.1 TESTING DOCUMENTATION AND PARTIES INVOLVED 11-1
11.2 TESTING ACTIVITIES IN SYSTEM DEVELOPMENT LIFE CYCLE..... 11-2

APPENDICES

- A. Checklist on Unit Testing
 - B. Checklist on Link Testing
 - C. Checklist on Function Testing
 - D. Checklist on System Testing
 - E. Checklist on Acceptance Testing
 - F. Checklist for Contracted out Software Development
-

1. PURPOSE

The major purpose of this document is to provide a set of application software testing guidelines, to ensure that computer systems are properly tested, so as to pursue reliable and quality computer systems.

2. SCOPE

This document, as its name suggested, is to give a set of guidelines to be referenced by application project teams on the planning and carrying out of testing activities for the application software. **It is important that users of this document (i.e. application project teams) should not treat these guidelines as mandatory standards, but as a reference model; and should tailor the guidelines according to individual project's actual situation.**

This document at its current version is most suitable to development projects following the SDLC which is defined in the department's Information Systems Procedures Manual. As for the maintenance projects, these guidelines may need to be adjusted according to projects' actual situation. Such adjustments will be the responsibility of individual project team.

It is intended that the guidelines would be applicable to both the in-house-developed and contract-out software projects.

3. REFERENCES

3.1 STANDARDS

Nil

3.2 OTHER REFERENCES

The Art of Software Testing by Glenford J. Myers

A Structured Approach to Systems Testing by William E. Perry

IEEE Standard for Software Test Documentation by IEEE

NCC IT Starts Developers' Guide by NCC

4. DEFINITIONS AND CONVENTIONS

4.1 DEFINITIONS

Nil

4.2 CONVENTIONS

Nil

5. OVERVIEW

This document, in essence, suggests a reference model for the planning and carrying out of Application Software Testing in OGCIO. The following serve as an overview of the model :

5.1 PROJECT ORGANISATION

5.1.1 Test Control Sub-Committee

A Test Control Sub-Committee is set up to co-ordinate, monitor and resolve priority conflict on the testing activities. The emphasis here is on the necessity of these co-ordination activities. Therefore for those small-sized projects not justifying existence of such sub-committee, its function is still required but is to be achieved through discussion meeting between project team and user representatives.

(Refer to Section 9.3)

5.1.2 Independent Test Group

Where resource constraints permitted, an independent Test Group is set up to carry out the testing activities. The emphasis here is on the independent role of the Test Group, which does not necessarily mean dedicated resources.

(Refer to Section 9.2)

5.2 TESTING ACTIVITIES

To test out a computer system from the following 5 different perspectives (the emphasis here is on different perspectives, which do not necessarily mean different testing phases):

- (i) To validate individual program modules against program specifications (Unit Testing);
- (ii) To validate program linkages against design specifications (Link Testing);
- (iii) To validate integrated software against functional specifications (Function Testing);
- (iv) To validate the integrated software against specifications on operating environment (System Testing); and,
- (v) To validate the integrated software against end-user needs (Acceptance Testing).

(Refer to Section 7)

5.3 TESTING DOCUMENTATION

To document testing activities through the use of

- (i) Test Plan
- (ii) Test Specification
- (iii) Test Incident Report
- (iv) Test Progress Report
- (v) Test Summary Report

(Refer to Section 8)

5.4 TEST PLANNING & CONTROL

5.4.1 Progress Control

To monitor day-to-day progress of the testing activities through the use of Test Progress Reports.

(Refer to Section 8.5)

5.4.2 Quality Control / Assurance

Testing documentation be compiled by Test Group, cross-checked by Quality Assurance Staff¹, and endorsed by Test Control Sub-committee / Project Committee.

5.4.3 Resource Estimation

Project teams to submit testing metrics information to the Metrics Team if one exists for system development and maintenance. It is advisable for the latter to update the metrics information to a centralized database for future test planning references.

¹ Quality Assurance Staff should be the Systems Manager or his/her delegates. However, those who are the members of the Test Group should not take up the quality assurance role for the project.

6. GENERAL CONCEPTS OF TESTING

6.1 TESTING OBJECTIVES

Testing is the process of executing program(s) with the intent of finding errors, rather than (a misconception) of showing the correct functioning of the program(s). The distinction may sound like a matter of semantics, but it has been observed to have profound effect on testing success. The difference actually lies on the different psychological effect caused by the different objectives: If our goal is to demonstrate that a program has no errors, then we will tend to select tests that have a low probability of causing the program to fail. On the other hand, if our goal is to demonstrate that a program has errors, our test data will have a higher probability of finding errors.

Specifically, testing should bear the following objectives :

- (a) to reveal design errors;
- (b) to reveal logic errors;
- (c) to reveal performance bottleneck;
- (d) to reveal security loophole; and
- (e) to reveal operational deficiencies.

All these objectives and the corresponding actions contribute to increase quality and reliability of the application software.

6.2 TESTING STRATEGY

There are two strategies for testing software, namely White-Box Testing and Black-Box Testing.

White-Box Testing, also known as Code Testing, focuses on the independent logical internals of the software to assure that all code statements and logical paths have been tested.

Black-Box Testing, also known as Specification Testing, focuses on the functional externals to assure that defined input will produce actual results that agreed with required results documented in the specifications.

Both strategies should be used, according on the levels of testings.

6.3 LEVELS OF TESTINGS

There are 5 levels of Testings, each of which carries a specific functional purpose, to be carried out in chronological order.

	<u>Testing Strategy Applied</u>
(a) Unit Testing - Testing of the program modules in isolation with the objective to find discrepancy between the programs and the program specifications	White Box Test
(b) Link Testing - Testing of the linkages between tested program modules with the objective to find discrepancy between the programs and system specifications	White Box Test
(c) Function Testing - Testing of the integrated software on a function by function basis with the objective to find discrepancy between the programs and the function specifications	Black Box Test
(d) Systems Testing - Testing of the integrated software with the objective to find discrepancy between the programs and the original objectives with regard to the operating environment of the system (e.g. Recovery, Security, Performance, Storage, etc.)	Black Box Test
(e) Acceptance Testing - Testing of the integrated software by the end-users (or their proxy) with the objective to find discrepancy between the programs and the end-user needs	Black Box Test

6.4 GENERAL TESTING PRINCIPLES

The following points should be noted when conducting training:

- (a) As far as possible, testing should be performed by a group of people (referred to in this document as Test Group) different from those performing design and coding of the same system. Please refer to Section 9.2 for its description.
- (b) Test cases must be written for invalid and unexpected, as well as valid and expected input conditions. A good test case is one that has a high probability of detecting undiscovered errors. A successful test case is one that detects an undiscovered error.
- (c) A necessary part of a test case is a definition of the expected outputs or results.
- (d) Do not plan testing effort on assumption that no errors will be found.
- (e) The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.
- (f) Testing libraries should be set up allowing Regression test be performed at system maintenance and enhancement times.
- (g) The later in the development life cycle a fault is discovered, the higher the cost of correction.
- (h) Successful testing is relying on complete and unambiguous specification.

6.5 COMPLEMENTARY REVIEWS

The testings mentioned in section 6.3 essentially follow a Bottom-Up approach, which has one associated shortcoming, i.e. requirement and design errors could only be identified at a late stage. To circumvent this, it is most important that the following reviews should also be performed :

(a) Requirement Review

- to review the requirement specification with the objective to identify requirement items that are Incomplete, Incorrect, Inconsistent, and that are Not testable.
- proposed participants : Analyst, designer, test group, users and quality assurance staff.

(b) Design Review

- to review the design specification with the objective to identify design items that are Incomplete, Incorrect, Inconsistent and that are Not testable.
- proposed participants : Designer, test group, computer operators (on operational procedures), quality assurance staff, users (on human interface and manual procedures) and where possible design specialists from vendors.

(c) Program Walkthrough

- to walkthrough, at least the most critical ones, program modules with the objective to identify errors as against the program specifications.
- proposed participants : Designer, programmer, and lead programmer.

(Please refer to Section 11.2 as to when should these reviews be carried out)

7. LEVELS OF TESTING

7.1 UNIT TESTING

7.1.1 Scope of Testing

Unit Testing (or Module Testing) is the process of testing the individual subprograms, subroutines, or procedures in a program. The goal here is to find discrepancy between the programs and the program specifications.

7.1.2 Activities, Documentation and Parties Involved

- (a) Designers to include test guidelines (i.e. areas to be tested) in the design and program specifications.
- (b) Programmers to define test cases during program development time.
- (c) Programmers to perform Unit Testing after coding is completed.
- (d) Programmers to include the final results of Unit Testing in the program documentation.

7.1.3 Practical Guidelines

- (a) Testing should first be done with correct data, then with flawed data.
- (b) A program unit would only be considered as completed after the program documentation (with testing results) of the program unit have been submitted to the project leader/SA.
- (c) If there are critical sections of the program, design the testing sequence such that these sections are tested as early as possible. A 'critical section' might be a complex module, a module with a new algorithm, or a I/O module.
- (d) There are three types of the test that the Unit Testing should satisfy :
 - (i) Functional Test
Execute the program unit with normal and abnormal input values for which the expected results are known.
 - (ii) Performance Test (Optional)
Determine the amount of execution time spent on various parts of the program unit, response time and device utilization by the program unit.
 - (iii) Stress Test (Optional)
Stress test is designed to test the program unit with abnormal situations. A great deal can be learned about the strengths and limitations of a program unit by executing it in a manner that demands resources in abnormal quantity, frequency, or volume.
- (e) Please refer to Appendix A for a checklist on Unit Testing.

7.2 LINK TESTING

7.2.1 Scope of Testing

Link Testing is the process of testing the linkages between program modules as against the system specifications. The goal here is to find errors associated with interfacing. As a by-product of the testing process, the software modules would be integrated together.

It is worth noting that this level of testing is sometimes referred to as “Integration Testing”, which is understood to mean that the testing process would end up with the software modules in integration. However after some careful consideration, the term was abandoned as it would cause some confusion over the term “System Integration”, which means integration of the automated and manual operations of the whole system.

7.2.2 Activities, Documentation and Parties Involved

- (a) Test Group to prepare a Link Testing test plan.
- (b) Test Group to prepare a Link Testing test specification before testing commences.
- (c) Test Group, with the aid of the designers/programmers, to set up the testing environment.
- (d) Test Group to perform Link Testing; and upon fault found issue Test Incident Reports to Designers/programmers, who would fix up the liable errors.
- (e) Test Group to report progress of Link Testing through periodic submission of the Link Testing Progress Report.

7.2.3 Practical Guidelines

- (a) Both control and data interface between the programs must be tested.
- (b) Both Top-down and Bottom-up approaches can be applied.

Top-down integration is an incremental approach to the assembly of software structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module ('main program'). Modules subordinate (and ultimately subordinate) to the main control module are incorporated into the structure in either a depth-first or the breadth-first manner.

Bottom-up integration, as its name implies, begins assembly and testing with modules at the lowest levels in the software structure. Because modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available, and the need for stubs (i.e. dummy modules) is eliminated.

- (c) As a result of the testing, an integrated software should be produced.
- (d) Please refer to Appendix B for a checklist on Link Testing.

7.3 FUNCTION TESTING

7.3.1 Scope of Testing

Function Testing is the process of testing the integrated software on a function by function basis as against the function specifications. The goal here is to find discrepancy between the programs and the functional requirements.

7.3.2 Activities, Documentation and Parties Involved

- (a) Test Group to prepare Function Testing test plan, to be endorsed by the Project Committee via the Test Control Sub-Committee, before testing commences. (Please refer Section 9.3)
- (b) Test Group to prepare a Function Testing test specification before testing commences.
- (c) Test Group, with the aid of the designers/programmers, to set up the testing environment.
- (d) Test Group (participated by user representatives) to perform Function Testing; and upon fault found issue test incident reports to Designers/programmers, who fix up the liable errors.
- (e) Test Group to report progress of Function Testing through periodic submission of the Function Testing Progress Report.

7.3.3 Practical Guidelines

- (a) It is useful to involve some user representatives in this level of testing, in order to give them familiarity with the system prior to Acceptance test and to highlight differences between users' and developers' interpretation of the specifications. However, degree of user involvement may differ from project to project, and even from department to department, all depending on the actual situation.
- (b) User involvement, if applicable, could range from testing data preparation to staging out of the Function Testings.
- (c) It is useful to keep track of which functions have exhibited the greatest number of errors; this information is valuable because it tells us that these functions probably still contain some hidden, undetected errors.
- (d) Please refer to Appendix C for a checklist on Function Testing.

7.4 SYSTEM TESTING

7.4.1 Scope of Testing

Systems Testing is the process of testing the integrated software with regard to the operating environment of the system. (i.e. Recovery, Security, Performance, Storage, etc.)

It may be worthwhile to note that the term has been used with different environments. In its widest definition especially for the small scale projects, it also covers the scope of Link Testing and the Function Testing.

For small scale projects which combine the Link Testing, Function Testing and System Testing in one test plan and one test specification, it is crucial that the test specification should include distinct sets of test cases for each of these 3 levels of testings.

7.4.2 Activities, Documentation and Parties Involved

- (a) Test group to prepare a System Testing test plan, to be endorsed by the Project Committee via the Test Control Sub-Committee, before testing commences.
- (b) Test group to prepare a System Testing test specification before testing commences.
- (c) Test group, with the aid of the designers/programmers, to set up the testing environment.
- (d) Test group (participated by the computer operators and user representatives) to perform System Testing; and upon fault found issue test incident reports to the Designers/programmers, who would fix up the liable errors.
- (e) Test group to report progress of the System Testing through periodic submission of the System Testing Progress Report.

7.4.3 Practical Guidelines

- (a) Eight types of Systems Tests are discussed below. It is not claimed that all 8 types will be mandatory to every application system nor are they meant to be an exhaustive list. To avoid possible overlooking, all 8 types should be explored when designing test cases.
 - (i) Volume Testing
Volume testing is to subject the system to heavy volumes of data, and the attempt of which is to show that the system cannot handle the volume of data specified in its objective. Since volume testing being obviously expensive, in terms of both machine and people time, one must not go overboard. However every system must be exposed to at least a few volume tests.
 - (ii) Stress Testing
Stress testing involves subjecting the program to heavy loads or stress. A

heavy stress is a peak volume of data encountered over a short span of time. Although some stress test may experience 'never will occur' situations during its operational use, but this does not imply that these tests are not useful. If errors are detected by these 'impossible' conditions, the test is valuable, because it is likely that the same errors might also occur in realistic, less-stressful situations.

(iii) Performance Testing

Many programs have specific performance or efficiency objectives, such as response times and throughput rates under certain workload and configuration conditions. Performance testing should attempt to show that the system does not satisfy its performance objectives.

(iv) Recovery Testing

If processing must continue during periods in which the application system is not operational, then those recovery processing procedures/contingent actions should be tested during the System test. In addition, the users of the system should be involved in a complete recovery test so that not only the application system is tested but the procedures for performing the manual aspects of recovery are tested.

(v) Security Testing

The adequacy of the security procedures should be tested by attempting to violate those procedures. For example, testing should attempt to access or modify data by an individual not authorized to access or modify that data.

(vi) Procedure Testing

Computer systems may not contain only computer processes but also involve procedures performed by people. Any prescribed human procedures, such as procedures to be followed by the system operator, data-base administrator, or terminal user, should be tested during the System test.

(vii) Regression Testing

Regression testing is the verification that what is being installed does not affect any already installed portion of the application or other applications interfaced by the new application.

(viii) Operational Testing

During the System test, testing should be conducted by the normal operations staff. It is only through having normal operation personnel conduct the test that the completeness of operator instructions and the ease with which the system can be operated can be properly evaluated. This testing is optional, and should be conducted only when the environment is available.

- (b) It is understood that in real situations, due to possibly environmental reasons, some of the tests (e.g. Procedures test, etc.) may not be carried out in this stage and are to be delayed to later stages. There is no objection to such delay provided that the reasons are documented clearly in the Test Summary Report and the test be carried out once the constraints removed.

(c) Please refer to Appendix D for a checklist on Systems Testing.

7.5 ACCEPTANCE TESTING

7.5.1 Scope of Testing

Acceptance Testing is the process of comparing the application system to its initial requirements and the current needs of its end users. The goal here is to determine whether the software end product is not acceptable to its user.

7.5.2 Activities, Documentation and Parties Involved

- (a) User representatives to prepare an Acceptance Testing test plan, which is to be endorsed by the Project Committee via the Test Control Sub-Committee.
- (b) User representatives to prepare an Acceptance Testing test specification, which is to be endorsed by the Project Committee via the Test Control Sub-Committee.
- (c) User representatives, with the aid of OGCIO officers, to set up the testing environment.
- (d) User representatives to perform Acceptance Testing; and upon fault found issue test incident reports to the Designers/programmers, who will fix up the liable error.
- (e) User representatives to report progress of the Acceptance Testing through periodic submission of Acceptance Testing Progress Report.
- (f) OGCIO Systems Manager to keep the overall Test Summary Report as documentation proof.

7.5.3 Practical Guidelines

- (a) There are three approaches for Acceptance Testing, namely,
 - (i) A planned comprehensive test using artificial data and simulated operational procedures, and usually accompanied with the Big Bang implementation approach but can also be used as a pre-requisite step of other approaches.
 - (ii) Parallel run using live data and would normally be used when comparison between the existing system and the new system is required. This approach requires duplicated resources to operate both systems.
 - (iii) Pilot run using live data and would normally be used when the user is not certain about the acceptance of the system by its end-users and/or the public.

Users are responsible to select the approach(es) that is most applicable to its

operating environment.

(b)

(b) Precautions for OGCIO liaison officers

- (i) to communicate clearly to the users of their commitments on the testing
- (ii) some users may be physically involved for the first time; therefore sufficient presentation, introduction, and training will be very important
- (iii) development team must be available to resolve problems if required
- (iv) if possible, future maintenance team should be identified
- (v) ensure all tasks are completed before handover to the maintenance team

(c) Precaution for users

- (i) testing staff should be freed from their routine activities
- (ii) commitment is authorized

(d) In the case of contract-out projects, “user representatives” would mean OGCIO + User department. Similar to as in the case of inhouse projects that OGCIO will assist user departments in preparing the testing plan & specification, vendors’ assistance in these areas could be asked for.

(e) Please refer to Appendix E for a checklist on User Acceptance Testing.

8. TEST DOCUMENTATION

8.1 INTRODUCTION

The following summarises the testing documentation (highlighted by the number in bracket) to be produced in a project:

For each project

References of the 5 levels of testing as well as any necessary complementary reviews (refer to section 6.5) in the project plan (1)

For each of the 4 levels of testing (i.e. Link, Function, Systems and Acceptance Testing)

Prepare a test plan (2)

Prepare a test specification (3)

Prepare test incident reports for faults found (4)

Prepare periodic test progress reports (5)

End Level

Prepare test summary report after completion of the tests (6)

End Project

The above documentation will be subject to quality assurance checks for existence and completeness by the Quality Assurance Staff.

Note: For small-sized projects, test plan & test specification as for Link Testing, Function testing, Systems Testing could be combined.

8.2 TEST PLAN

8.2.1 Purpose of Document

To prescribe the scope, approach, resources, and schedule of the testing activities for a level of testing. To identify the items being tested, the features to be tested, the testing tasks to be performed, and the personnel responsible for each task.

This test plan should be prepared for each level of testing except Unit Testing.

8.2.2 Outline of Document

The testing plan should provide at least the following information :

(a) Test Items

List the functional items and software features to be tested.

For Link Testing, list the software items to be tested (which in most of the cases should be ALL software items).

For Function Testing, list the functions in the function catalogue (which in most cases should be ALL functions).

For Systems Testing, list the tests to be carried out.

(b) Test Tasks

Normally, there are the following 4 tasks:

- 1) Prepare test specification, regarding
 - (i) Test procedures
 - (ii) Test cases
 - (iii) Test data
 - (iv) Test environment
- 2) Set up of the testing environment
- 3) Load test data
- 4) Conduct tests

(*Do not plan on the assumption that each test case will only be executed once)

For each task, list the estimated effort required & duration.

For example,

Task No.	Task Description	Estimated Effort (man-weeks)	Relative Calendar week 0 1 2 3 4 5 6 7 8 9...
1	Prepare test specification on		
	-Test control procedures	#	XXXXXX
	-Test cases	#	XXXX
	-Test data	#	XXX
	-Test environment	#	XX
2	Set up of testing environment	#	X
3	Load test data	#	X
4	Conduct tests	#	XXXX

- (c) Responsibilities of relevant parties. For each party, specify their corresponding responsibilities for the testing levels.
- (d) Remarks. Describe any special constraint on the test procedures, identify any special techniques and tools requirements that are necessary for the execution of this test.

8.3 TEST SPECIFICATION

8.3.1 Purpose of Document

To specify refinements of the test approach, to identify the features to be tested, to specify the steps for executing the tests and specify the test case for each tests.

8.3.2 Outline of Document

The test specification should provide, to the least, the following information :

(a) Test Control Procedures

To specify the following:

- 1) Error Reporting procedures;
- 2) Change / Version control procedures of S/W modules;
- 3) Set-up & Wind-down procedures of the testing environment.

(b) Testing Environment

To specify at least the following items that are required in the testing progress:

- 1) H/W & System S/W required;
- 2) Number of terminals required;
- 3) Testing facilities / tools required;
- 4) Test database; and
- 5) Operations support / Operating hour.

(c) Test Termination Criteria

To specify the criteria (e.g. Failing on certain critical test cases, when no. of error reaches a certain limit, etc.) under which the testing would be terminated.

(d) Test Cases

Identify and briefly describe the test cases selected for the testing. For each test cases, specify also the steps (e.g. bring up screen, input data, keys pressed etc.), the expected outputs (e.g. message displayed, file changes, etc.), programs involved and specify whether the test cases has passed or failed after the testing has been conducted.

It should be noted that definition of test cases is a “design” process and do vary for different projects. Please refer to Appendices A to F for test case design checklists.

8.4 TEST INCIDENT REPORT

8.4.1 Purpose of Document

To document any event that occurs during the testing process which requires investigation. The report is to be issued to the designers/programmers for the errors found in the testing progress.

8.4.2 Outline of Document

The test incident report should provide the following information:

(a) Test-Incident-report Identifier

Specify the unique identifier assigned to the test incident report.

(b) Summary

Summarize the incident.

Identify the test items involved indicating their version/revision level.

(c) Incident Description

Provide a description of the incident. This description should include the following items:

- (i) Inputs
- (ii) Expected results
- (iii) Anomalies
- (iv) Date and time
- (v) Procedure step
- (vi) Environment
- (vii) Testers and Observers

(d) Impact

If known, indicate what impact this incident will have on test plan and test procedure specification.

(e) Results of Investigation

- (i) Classification of Incident
 - Design / Program error
 - Error related to testing environment
 - Others
- (ii) Action taken
 - Design / Program changes.
 - Testing Environment Changes
 - No action taken.

Test Incident Report

<u>Test Incident Report</u>	
TIR NO. : _____	Date : _____
Reported By : _____ _____	
Incident Description _____ _____ _____ _____	
Impact _____ _____ _____ _____	
Results of Investigation	
Investigated By _____	Date _____
Classification	Actions taken
<input type="checkbox"/> design/pgm error	<input type="checkbox"/> design/pgm changes
<input type="checkbox"/> error related to testing environment	<input type="checkbox"/> testing environ' changes
<input type="checkbox"/> others	<input type="checkbox"/> no action taken

8.5 TEST PROGRESS REPORT

8.5.1 Purpose of Document

In order that progress of the testing process be controlled properly, a periodic test progress report should be prepared by the test group, and submitted to the Test Control Sub-Committee / OGCIO Systems Manager.

Frequency of the report is suggested to be weekly or bi-weekly.

8.5.2 Terminology

No. of test cases specified : the total number of test cases that have been specified.

No. of test cases tried at least once : the number of the specified cases that have been put into test execution at least one time. (The quotient between this term and the pervious term gives the percentage of the specified test cases that have been executed at least once. More importantly for the complement of this quotient percentage, which gives the percentage of the specified test cases that no test runs have ever put against them so far)

No. of test cases completed : the number of the specified test cases that have been executed and generated the expected output.

8.5.3 Outline of Document

(a) Progress Control Summary.

	Levels of Testing : _____	
	Reporting Period : _____	
	(In the reporting period)	(Over-all)
No. of tests cases specified	_____	_____ (#1)
No. of test cases tried at least once	_____ % (of #1)	_____ % (#2) (of #1)
No. of test cases completed	_____ % (of #1)	_____ % (of #1)
No. of Incident reports issued	_____	_____ (#3)
No. of Incidents reports cleared	_____ % (of #3)	_____ % (of #3)
Test group testing effort (in man-hrs)	_____	_____
Designers / Programmers fault clearing effort (in man-hrs)	_____	_____
User testing effort (in man-hrs)	_____	_____

Notes: Testing effort should include ALL the effort directly related to the testing activities, but excluding the administration overhead.

(b) Highlights of Outstanding Items & Reasons

To bring to the management attentions the problems encountered / foreseen and where possible, the planned way of solving them.

(c) Test Cases Results (Optional)

Refer to Section 8.3.2 (d).

8.6 TEST SUMMARY REPORT

8.6.1 Purpose of Document

To summarize the results of the testing activities for documentation purpose and to provide information for future testing planning references.

8.6.2 Outline of Document

(a) Test cases Results

Refer to section 8.3.2(d)

(b) Remarks

9. TEST PLANNING AND CONTROL

9.1 TEST PLANNING

As general practices, Link Testing, Function Testing, System Testing and Acceptance Testing for the 3GL-based projects normally account to about 40% to 50% of the project implementation effort. (A higher percentage is expected for the 4GL-based projects).

The objective of test planning is to prepare the blueprint used by the project personnel and users to ensure that the application system achieves the level of correctness and reliability desired by the user.

Test planning normally is accomplished by performing the following 7 activities in the sequence indicated:

- 1) Identification and inclusion of the testing activities in the project plan;
- 2) Setting up of the Test Group & the Test Control Sub-Committee;

For each level, with the exception of Unit Testing, of testing,

- 3) Preparation of a test plan, which should define the scope of the testing and include criteria for ending testing;
- 4) Decide on what test method, strategies, tools and resources to be used;
- 5) Preparation of test specification;
- 6) Purchase or develop test tools prior to the time when they will be needed;
- 7) Set up of the testing environment.

9.2 TEST GROUP

By definition, testing is the process of executing a program with the intent of finding errors, since it is a “destructive” process, it is more effective and successful if performed by another party other than the original designers / programmers.

As far as possible, testing should be performed by a group of people different from those performing design and coding of the same system, and those people called the Test Group.

9.3 INDEPENDENT TEST CONTROL SUB-COMMITTEE

9.3.1 Introduction

It is recommend that a separate Test Control Sub-Committee to coordinate, monitor and resolve priority conflict on the testing activities be set up under the project committee. In case of formation of such Sub-Committee is not justified, as e.g. for the small-sized projects, the co-ordination and monitoring activities would still be required but through discussion meetings between OGCIO project team and user representatives.

9.3.2 Terms of reference of the Sub-Committee

- (a) To recommend the test plans, test specifications & test summary report be endorsed by the project level committee.
- (b) To ensure smooth running of testing activities.

9.3.3 Proposed membership of the sub-committee

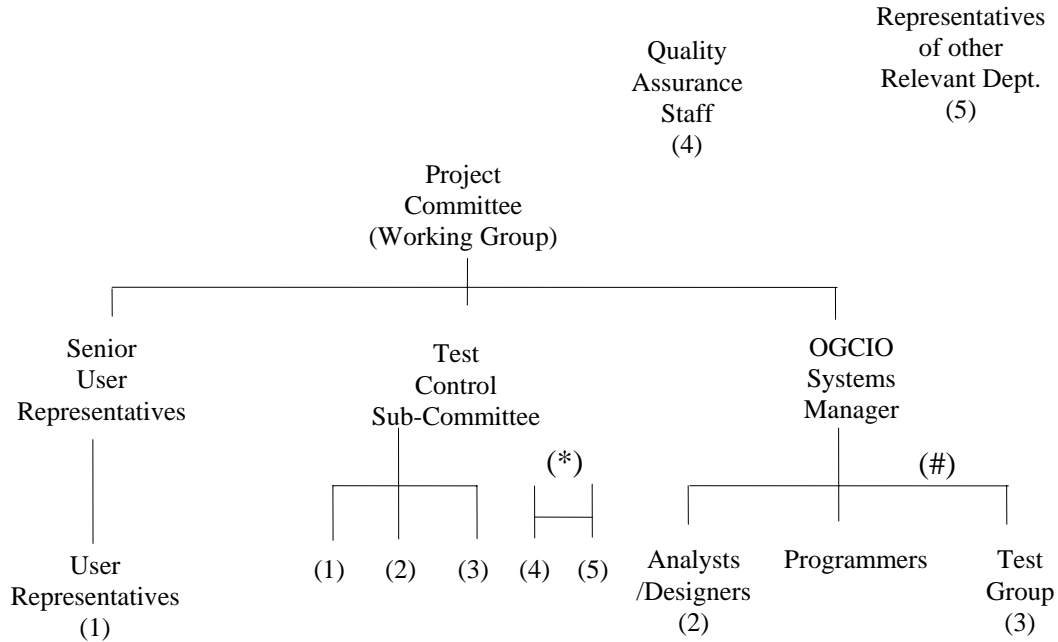
The ideal members of the Test Control Sub-Committee are:

- User representatives
- Key Analysts/Designers
- Test Group
- Quality Assurance Staff
- Representatives of relevant Dept. (e.g. Audit Dept.)

As regards the chairmanship, the choice would be made between User representatives and Test Group depending on the factors as whether the user side is ready / skillful enough to take up the responsibility, whether there are more than 1 user department involved and other environmental constraints.

Once again, for turnkey and contract-out projects, user representatives would mean OGCIO + User Department.

9.3.4 Project Structure



(*) Optional membership

(#) Please note that these job responsibilities should be considered as Roles rather than as actual persons. A person may assume more than 1 role in the project.

10. AUTOMATED TOOLS FOR TESTING

10.1 INTRODUCTION

Because software testing often accounts for as much as 50% of all effort expended on a software implementation project, tools that can reduce test time (but without reducing thoroughness) are very valuable. For that purpose, use of the following types of automated tools would be most desirable.

10.2 TEST DATA GENERATOR

These are the tools to generate typical input data for programs that are undergoing testing. System Managers are advised to acquire one from vendors.

e.g. Data-xpert

Please liaise with the support team if there is a need for such tools.

10.3 STATIC ANALYSERS

These are the tools to check about a program's structure, complexity and maintainability. Research is being conducted to identify such tools for general use.

Please liaise with the support team if there is a need for such tools.

10.4 DYNAMIC ANALYSERS

It is also known as Test Coverage Verifiers. These are the tools to measure internal test coverage of a program as to a given set of test data. Research is being conducted to identify such tools for general use.

Please liaise with the support team if there is a need for such tools.

11. SUMMARY

11.1 TESTING DOCUMENTATION AND PARTIES INVOLVED

Type of Testing	What is being tested	Testing against	Test data	Done by	Who does sign-off
Unit Testing	Program units subprograms job control & procedures	Program Specification	Correct data then with flawed data	Programmer	Lead Programmer + analyst
Link Testing	Linkages between program modules	Program Spec. + System Spec.	Control and data interface, returns/calls	Test Group	Project Committee recommended by Test Control Sub-committee
Function Testing	Integrated software on a function by function basis	Function Specification	Functions of the integrated software	Test Group	
System Testing	Integrated software	User Objectives + System Specification	User supplied tests data -pilot -parallel	Test Group	
Acceptance Testing	Entire system simulated in production environment	User requirements/ acceptance criteria	'life' data where possible	Users	

11.2 TESTING ACTIVITIES IN SYSTEM DEVELOPMENT LIFE CYCLE

	Analyst/ designers	Programmer	Test Group	User	Computer Operator	Test Control Sub Committee
Feasibility Study						
System Analysis	- Perform Requirement review		-Prepare Project Plan			
System Design	- Include test guidelines in design & program Specification - Perform design review		-Prepare Link test plan -Prepare Function test plan -Prepare System test plan -Prepare test specification for each of the Link, Function & Systems Testing -Set up testing environment	-Prepare Acceptance Test Plan (OGCIO officers to assist user to prepare the Acceptance Test Plan) -Prepare Acceptance Test specification		-Recommend endorsement of the Link, Function System & Acceptance test plan -Recommend endorsement of the test specification
Program Development		- Perform program walkthrough -Include extra test cases -Perform unit Testing				
System Integration & Tests			-Perform Link Testing -Perform Function Testing -Perform System Testing -Perform Acceptance Testing rehearsal		-Participate in system test on operation procedure -Accept operation procedure	-Ensure smooth running of testing activities -Recommend endorsement of test results
User Acceptance				-Perform Acceptance Test -Accept the system		

** For turnkey and contracted out development projects, the Acceptance test plan and test specification would be prepared by the joint effort of user department representatives and OGCIO officers.

APPENDIX A CHECKLIST ON UNIT TESTING

(This checklist to suggest areas for the definition of test cases is for information purpose only; and in no way is it meant to be an exhaustive list. Please also note that a negative tone that matches with section 6.1 suggestions has been used)

Input

1. Validation rules of data fields do not match with the program/data specification.
2. Valid data fields are rejected.
3. Data fields of invalid class, range and format are accepted.
4. Invalid fields cause abnormal program end.

Output

1. Output messages are shown with misspelling, or incorrect meaning, or not uniform.
2. Output messages are shown while they are supposed not to be; or they are not shown while they are supposed to be.
3. Reports/Screens do not conform to the specified layout with misspelled data labels/titles, mismatched data label and information content, and/or incorrect data sizes.
4. Reports/Screens page numbering is out of sequence.
5. Reports/Screens breaks do not happen or happen at the wrong places.
6. Reports/Screens control totals do not tally with individual items.
7. Screen video attributes are not set/reset as they should be.

File Access

1. Data fields are not updated as input.
2. “No-file” cases cause program abnormal end.
3. “Empty-file” cases cause program abnormal end.
4. Program data storage areas do not match with the file layout.
5. The last input record (in a batch of transactions) is not updated.
6. The last record in a file is not read while it should be.
7. Deadlock occurs when the same record/file is accessed by more than 1 user.

Internal Logic

1. Counters are not initialized as they should be.
2. Mathematical accuracy and rounding does not conform to the prescribed rules.

Job Control Procedures

1. A wrong program is invoked and/or the wrong library/files are referenced.
2. Program execution sequence does not follow the JCL condition codes setting.
3. Run time parameters are not validated before use.

Program Documentation

1. Documentation is not consistent with the program behavior.

Program Structure (through program walkthrough)

1. Coding structure does not follow installation standards.

Performance

1. The program runs longer than the specified response time.

Sample Test Cases

1. Screen labels checks.
2. Screen videos checks with test data set 1.
3. Creation of record with valid data set 2.
4. Rejection of record with invalid data set 3.
5. Error handling upon empty file 1.
6. Batch program run with test data set 4.

APPENDIX B CHECKLIST ON LINK TESTING

(This checklist to suggest areas for the definition of test cases is for information purpose only; and in no way is it meant to be an exhaustive list. Please also note that a negative tone that matches with section 6.1 suggestions has been used)

Global Data(e.g. Linkage Section)

1. Global variables have different definition and/or attributes in the programs that referenced them.

Program Interfaces

1. The called programs are not invoked while they are supposed to be.
2. Any two interfaced programs have different number of parameters, and/or the attributes of these parameters are defined differently in the two programs.
3. Passing parameters are modified by the called program while they are not supposed to be.
4. Called programs behaved differently when the calling program calls twice with the same set of input data.
5. File pointers held in the calling program are destroyed after another program is called.

Consistency among programs

1. The same error is treated differently (e.g. with different messages, with different termination status etc.) in different programs.

Sample Test Cases

1. Interface test between programs xyz, abc & jkl.
2. Global (memory) data file 1 test with data set 1.

APPENDIX C CHECKLIST ON FUNCTION TESTING

(This checklist to suggest areas for the definition of test cases is for information purpose only; and in no way is it meant to be an exhaustive list. Please also note that a negative tone that matches with section 6.1 suggestions has been used)

Comprehensiveness

1. Agreed business function is not implemented by any transaction/report.

Correctness

1. The developed transaction/report does not achieve the said business function.

Sample Test cases

1. Creation of records under user normal environment.
2. Enquiry of the same record from 2 terminals.
3. Printing of records when the printer is in normal condition.
4. Printing of records when the printer is off-line or paper out.
5. Unsolicited message sent to console/supervisory terminal when a certain time limit is reached.

APPENDIX D CHECKLIST ON SYSTEMS TESTING

(This checklist to suggest areas for the definition of test cases is for information purpose only; and in no way is it meant to be an exhaustive list. Please also note that a negative tone that matches with section 6.1 suggestions has been used)

Volume Testing

1. The system cannot handle a pre-defined number of transaction.

Stress Testing

1. The system cannot handle a pre-defined number of transaction over a short span of time.

Performance Testing

1. The response times is excessive over a pre-defined time limit under certain workloads.

Recovery Testing

1. Database cannot be recovered in event of system failure.
2. The system cannot be restarted after a system crash.

Security Testing

1. The system can be accessed by a unauthorized person.
2. The system does not log out automatically in event of a terminal failure.

Procedure Testing

1. The system is inconsistent with manual operation procedures.

Regression Testing

1. The sub-system / system being installed affect the normal operation of the other systems / sub-systems already installed.

Operation Testing

1. The information inside the operation manual is not clear and concise with the application system.
2. The operational manual does not cover all the operation procedures of the system.

Sample Test Cases

1. System performance test with workload mix 1.
2. Terminal power off when a update tranx is processed.
3. Security breakthrough - pressing different key combinations onto the logon screen.
4. Reload from backup tape.

APPENDIX E CHECKLIST ON ACCEPTANCE TESTING

(This checklist to suggest areas for the definition of test cases is for information purpose only; and in no way is it meant to be an exhaustive list. Please also note that a negative tone that matches with section 6.1 suggestions has been used)

Comprehensiveness

1. Agreed business function is not implemented by any transaction/report.

Correctness

1. The developed transaction/report does not achieve the said business function.

Sample Test Cases

(Similar to Function Testing)

**APPENDIX F CHECKLIST FOR CONTRACTED-OUT SOFTWARE
DEVELOPMENT**

1. Tailor and refer this guidelines in the tender spec.
2. Check for the inclusion of an overall test plan in the tender proposal or accept it as the first deliverable from the contractor.
3. Set up a Test Control Sub-committee to monitor the testing progress.
4. Review and accept the different types of test plan, test specifications and test results.
5. Wherever possible, ask if there are any tools (ref. Section 10) to demonstrate the structureness and test coverage of the software developed.
6. Perform sample program walkthrough.
7. Ask for a periodic test progress report. (ref. Section 8.5)
8. Ask for the contractor contribution in preparing for the Acceptance Testing process.
9. Ask for a Test Summary Report (ref. section 8.6) at the end of the project.