Office of the
Government Chief Information Officer

GUIDELINES FOR APPLICATION SOFTWARE TESTING

[G20]

Version: 1.10

May 2018

Amendment History				
Change Number	Revision Description	Pages Affected	Revision Number	Date
1	Update the document style in order to conform with the Document Style Manual Version 2.0.	Whole document	1.1	Dec 1999
2	Revise the outdated contents to reflect current practices.	2-1, 5-2, 6-2, 7-2, 7-4, 7-5, 7-7, 8-9, 10-1, 11-1	1.1	Dec 1999
3	The personnel to take up the quality assurance role is revised.	5-2, 6-4, 8-1, 9-2, 9-3	1.1	Dec 1999
4	Remove the banner 'Standards & Methods Document'.	Front Cover Page	1.2	Jun 2003
5	Remove page number of Amendment History and Tables of Contents	Amendment History and Tables of Contents	1.2	Jun 2003
6	Remove the Distribution and Release Page	Distribution and Release Page	1.2	Jun 2003
7	Merging of ITSD into the Commerce, Industry and Technology Bureau on 1 July 2004. Replace 'Information Technology Services Department (ITSD)' by 'The Office of the Government Chief Information Officer (OGCIO)'. Replace logo of ITSD by OGCIO	Whole Document Front Cover Page	1.3	Jul 2004
8	Review the update-ness		1.4	Sep 2005

9	Review the update-ness		1.5	Aug 2008
10	Review the update-ness		1.6	Oct 2009
11	Correct typos	7-1, 7-5, 7-8, 8-1, 9-2, D-1, F-1	1.7	Jul 2012
12	Revise the outdated contents to reflect current practices and insert a new appendix on Independent Testing Services	Whole document	1.8	Mar 2015
13	Revise to add guidelines for the essential need of involvement and effort of business users in User Acceptance Testing	7-5	1.9	Feb 2018
14	Revise to add guidelines for better communication of test plans (especially User Acceptance Testing plans) developed at different stages amongst stakeholders for timely completion of IT projects	3.2, 5.2, 7.5	1.10	May 2018

APPENDICES

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 is to give a set of guidelines for reference by application project teams on the planning and carrying out of testing activities for 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 adopt the guidelines according to individual project's actual situation.**

This document is suitable for development projects following the SDLC which is defined in the 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 in-house-developed and outsourced software development projects.

3. REFERENCES

3.1 STANDARDS

Nil

3.2 OTHER REFERENCES

(a) The Art of Software Testing, Third Edition by Glenford J. Myers, Corey Sandler and Tom Badgett
(b) A Structured Approach to Systems Testing by William E. Perry
(c) IEEE Standard for Software and System Test Documentation by IEEE
(d) NCC IT Starts Developers' Guide by National Computing Centre
(e) J Scheibmeir, J Herschmann (2018), Adopt a "Shift Left" approach to testing to accelerate and improve application development, Gartner Inc.
(f) T Murphy (2011), Agile and the rest of the organization: Requirement definition and management, Gartner Inc.
(g) M West (2017), Survey analysis: Agile now at the tipping point – Here's how to succeed, Gartner Inc.
(h) N Wilson (2013), Managing the Agile project, Gartner Inc.
(i) M Hotle, N Wilson (2017), The end of the waterfall as we know it, Gartner Inc.
(j) M West (2017), Scrum is not enough: Essential practices for Agile success, Gartner Inc.
(k) A Rodger (2014), The impact of Agile on IT Governance, Ovum Ltd.
(l) C Singh (2013), Agile Requirements Management, Ovum Ltd.
(m) C Singh (2013), Continuous delivery for the Agile organization, Ovum Ltd.
(n) S Archer, C Kaufman (2013), Accelerating outcomes with a hybrid approach within a waterfall environment, Project Management Institute.

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 conduct of Application Software Testing. The following serves as an overview of the model:

5.1 PROJECT ORGANISATION

5.1.1 Project Organisation Structure

A project organisation is set up to champion, manage and execute an IT project. Members of the project organisation include the Project Owner, Project Steering Committee (PSC), Project Assurance Team (PAT), Internal Project Manager (IPM) and Business Analyst (BA).

The PSC champions the project and is the ultimate decision-maker for the project. It provides steer and support for the IPM and endorses acceptance of project deliverables. The IPM manages the project and monitors the project implementation on a day-to-day basis for the Project Owner/the PSC.

The PAT is responsible for overseeing project progress and managing quality assurance activities, which include:

(a) recommending the test plans, test specifications and test summary report for endorsement by the PSC; and
(b) co-ordinating, monitoring and resolving priority conflicts on the testing activities to ensure smooth running of testing activities.

Please refer to the Practice Guide to Project Management for IT Projects under an Outsourced Environment (PGPM)[2] for more details of the project organisation.

5.1.2 Test Group

Testing is the process of executing a program with the intent of finding errors. Since it is such a "destructive" process, it may be more effective and successful if the testing is performed by an independent third party other than the original system analysts / 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. That group of people is called the Test Group.

A Test Group can be set up to carry out the testing activities especially for large-scale projects or projects involving a large number of users. The emphasis here is on the independent role of the Test Group, which does not necessarily mean dedicated

[2] Practice Guide to Project Management for IT Projects under an Outsourced Environment [S19] can be found on ITG InfoStation web site at http://itginfo.ccgo.hksarg/content/pgpm/index.asp.

resources. The necessity of an independent Test Group should be determined at the project initialisation stage through an assessment based on project complexity, criticality, implementation schedule and other risks. The type(s) of testing to be performed independently and the high level estimation of additional resources, if required, should be determined and planned for respectively as early as possible.

The following figure shows an example of project organisation with the formation of a Test Group and an optional Independent Testing Contractor providing independent testing service. It is noted that the independent testing may be conducted by a Test Group of in-house staff members as well as by external contractor.

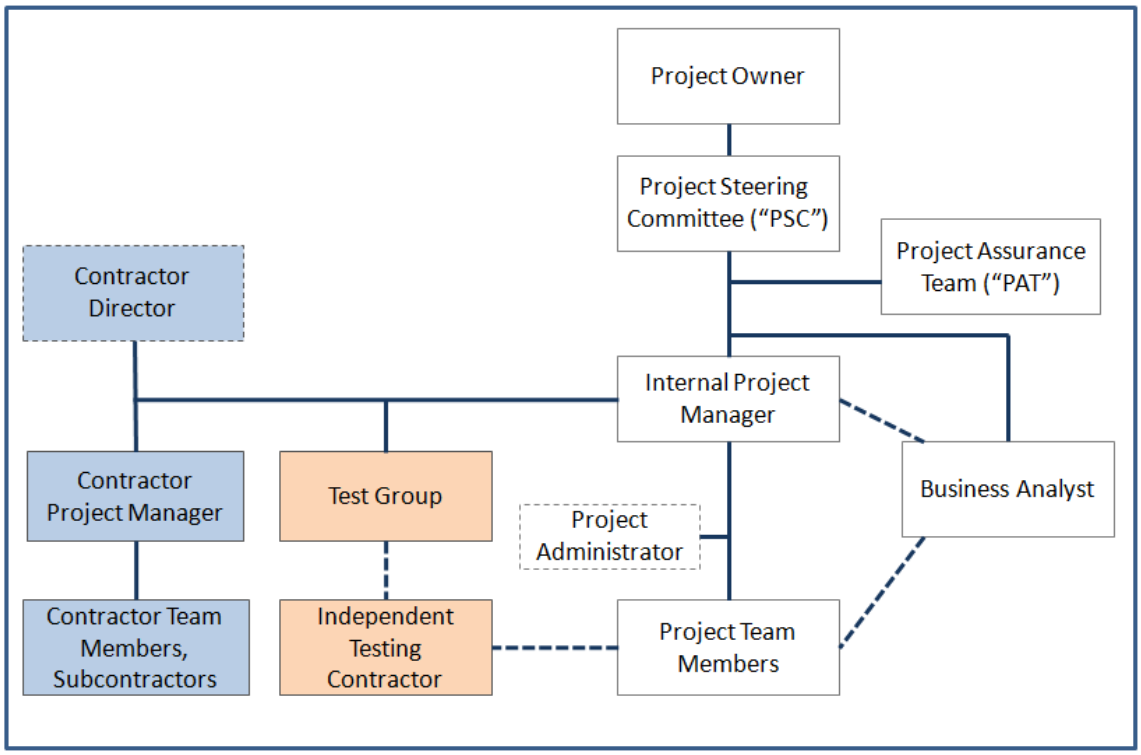


Figure 1 - Example of Project Organisation with Test Group and Independent Testing Contractor

5.2 TESTING ACTIVITIES

With reference to the Agile Software Development Method, test preparation should be started as early as possible and constant communication should be maintained with relevant stakeholders to facilitate collaboration and transparency. The following activities are suggested:

(i) The IPM should develop a high level test plan covering all major test types during project initiation with the objectives, scope of testing and the composition of Test Group including contractors, business users and internal IT staff with defined roles and responsibilities.

(ii) Contractor project manager (or IPM for in-house developed project) to enrich the test plans by engaging his/her staff to draft test cases; internal IT staff to

check all major test plans; and business users to provide different test cases to address different scenarios; and

(iii) Contractor project manager (or IPM for in-house developed project) to maintain ongoing communication and collaboration among stakeholders by distributing all major test plans and feedbacks to stakeholders regularly to keep them informed the project progress throughout the whole system development stage.

A computer system is subject to testing from the following five different perspectives:

(i) To validate individual program modules against program specifications (Unit Testing);

(ii) To validate linkages or interfaces between program modules against design specifications (Link/Integration 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 and business requirements (Acceptance Testing).

(Refer to Section 7)

5.3 TEST 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 and CONTROL

5.4.1 Progress Control

Monitor the 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 to be compiled by Test Group or Independent Testing Contractor if outsourced, cross-checked by quality assurance staff[3], and reviewed by the PAT.

5.4.3 Resource Estimation

Project teams may update the testing metrics information to a centralized database for future test planning references.

[3] Quality assurance staff should be the IPM or other delegated staff. However, those who are the members of the Test Group should not take up the quality assurance role for the project if the tests are conducted by them but not by Independent Testing Contractor.

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) showing the correct functioning of the program(s). 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 improve the quality and reliability of the application software.

6.2 TESTING APPROACH

There are two approaches to software testing, 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 agree with required results documented in the specifications.

Both approaches should be used, according to the level of testing.

6.3 LEVELS OF TESTING

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

	Testing Approach Applied
(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/Integration Testing	
- Testing of the linkages or interfaces 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) System 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 and business requirements	Black Box Test

6.4 GENERAL TESTING PRINCIPLES

The following points should be noted when conducting testing:

(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. For large-scale projects or projects which require higher quality in terms of accuracy or reliability, testing may be performed by a third-party Independent Testing Contractor.

(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 defining the expected outputs or results.

(d) Do not plan testing effort on assumption that no errors will be found.

(e) If much more errors are discovered in a particular section of a program than other sections, it is advisable to spend additional testing efforts on this error-prone section as further errors are likely to be discovered there.

(f) Testing libraries should be set up allowing regression test to 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) Success of testing relies on complete and unambiguous specification.

6.5 COMPLEMENTARY REVIEWS

The testing 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) Requirements Review

(i) to review the requirements specification with the objective to identify requirement items that are Incomplete, Incorrect, Inconsistent, or not testable.
(ii) proposed participants : e.g. business analysts, system analysts, Test Group, users and quality assurance staff (on user requirements specification).

(b) Design Review

(i) to review the design specification with the objective to identify design items

that are Incomplete, Incorrect, Inconsistent, or not testable.

(ii) proposed participants : e.g. system analysts (on system design), Test Group, computer operators (on operational procedures), business analysts (on functional specifications), quality assurance staff, users (on user interface and manual procedures), domain architects (on architecture design) and where possible domain experts from vendors (on specific domains such as storage and network infrastructure).

(c) Program Walkthrough

(i) to walkthrough, at least the most critical ones, program modules with the objective to identify errors as against the program specifications.

(ii) proposed participants : e.g. system analysts, programmers, and lead programmers.

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

6.6 INDEPENDENT TESTING

Due to the demands on improving quality of IT projects, third-party independent testing is recommended to ensure that system functions are completed in good quality. Independent testing may include various kinds of testing, e.g. unit test, functional test, regression test, integration test, load test, accessibility test, and usability test, etc.

The independent Test Group conducts testing for the system with the aim to reveal defects which cannot be found by the project team, which improves the quality of the system in fulfilling project or business needs without any biases. Independent Test Group sometimes engages professional software testers to perform testing according to internationally recognised standards and methodologies which ensure the quality of testing in conformance to requirements of the project or business.

6.6.1 Independent Testing Services by outsourced contractors

The independent Test Group may acquire external professional testing services from an Independent Testing Contractor to conduct various types of testing to help discover hidden problems in the system. The benefits of acquiring external independent testing services are:

(a) to improve the objectiveness and accuracy of test results, free from the influence of the developers or users;
(b) to meet industry or business standards, or comply with policies/regulations;
(c) to bring in expertise and skills which may not be available in the project team;
(d) to incorporate more effective and standardised quality control and failure analysis;
(e) to enhance the quality, availability and reliability of the software, reducing the costs of remediation during maintenance;

(f) to release the project team to concentrate on development and implementation activities; and
(g) to save costs in acquiring testing tools, facilities and equipment that may be used for one-off testing only.

Considerations for conducting independent testing by outsourced contractors:

(a) System requirements are fixed and readily available.
(b) Business domain knowledge is relatively easy to learn and understand.
(c) Test process requires deeper/stronger/specialised skills which are not available in-house.
(d) An independent third party's certification or proof is required on the robustness of the system.
(e) Team members have less/no skills on testing.
(f) Project is developing a mission-critical system or a large-scale system.
(g) Project does not involve sensitive data or processes.
(h) It is a high-risk project.

For example, if a project develops mission-critical system in which the quality is very important, it is helpful to employ an external Independent Testing Contractor to perform system testing to assure the quality and reduce the number of defects. If it is a high-risk project, it is necessary to acquire a professional testing contractor to help conduct function or system testing to increase the project success rate.

If a project is a short-term, small and low-risk project, it may not be cost-effective to acquire independent testing services. Moreover, if the project team has already possessed experienced team members having skills on testing or utilizing test-driven development approach, there may not be a necessity to acquire independent testing services.

6.6.2 Independent Testing by In-house Resources

Independent testing may be conducted by an in-house team other than the project development team; this can provide an impartial test result from the angle of an independent third-party, though not an external contractor.

As the in-house independent team has more business domain knowledge than that of the outsourced contractors, time and resources required for understanding the business requirements before conducting testing will be saved.

Considerations for conducting independent testing by in-house resources:

(a) Business domain knowledge is difficult to learn and understand by outsourced contractors.
(b) Technical complexity of the project is high.
(c) Project involves sensitive data or processes.
(d) Team members have sufficient technical skills and experience in testing.
(e) There are undisclosed business know-how and processes.
(f) There are unique requirements on business domain knowledge.

(g) System requirements and test cases are volatile.

Please refer to ***Appendix H*** for further details of independent testing services.

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, classes, or procedures in a program. The goal here is to find discrepancy between the programs and the program specifications prior to its integration with other units.

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 has 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 an I/O module.

(d) A unit testing will be considered as completed if the following criteria are satisfied:
   (i) all test cases are properly tested without any errors;
   (ii) there is no discrepancy found between the program unit and its specification; and
   (iii) program units of critical sections do not have any logical and functional errors.

(e) Well-documented test cases can be reused for testing at later stages.

(f) A team-testing approach may be applied in Unit Testing to improve the coverage of testing. A team is formed consisting of two or more programmers. The first programmer will write the test case for the other programmer to code. Upon completion, the first programmer will conduct unit test on the program according to the test case. This helps to improve the accuracy and reduce the defects found in the program.

(g) Please refer to Appendix A for a checklist on Unit Testing.

(h) A code review may be performed after the Unit Testing. A code review is a process of inspecting the source code of a software program line-by-line carefully by one or more reviewers with the aid of automated testing tools, aiming to find out defects in the code before proceeding to the next stage of development. It helps improve the quality and maintainability of the program.

7.2 LINK/INTEGRATION TESTING

7.2.1 Scope of Testing

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

This level of testing is often referred to as "Integration Testing", which is understood to mean that the test process would end up with the software modules in integration i.e. integrated together. Please note that the term "System Integration" means integration of the automated and manual operations of the whole system which is not the same as the Integration Test stated here.

7.2.2 Activities, Documentation and Parties Involved

(a) Test Group to prepare a Link/Integration Testing test plan.

(b) Test Group to prepare a Link/Integration Testing test specification before testing commences.

(c) Test Group, with the aid of the system analysts/programmers, to set up the testing environment.

(d) Test Group to perform Link/Integration Testing; and upon fault found issue Test Incident Reports to system analysts/programmers, who would fix up the liable errors.

(e) Test Group to report progress of Link/Integration Testing through periodic submission of the Link/Integration Testing Progress Report.

7.2.3 Practical Guidelines

(a) Both control and data interface between the programs must be tested.

(b) The following approaches can be applied during Link/Integration Testing:

(i) 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.

(ii) 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.

(iii) Sandwich integration approach: it is a mix of the top-down and bottom-up approaches. It divides the software into three layers. The bottom layer contains all the modules that are often invoked, and bottom-up approach is applied to the integrated modules in this layer. The top layer contains modules implementing major design decisions. These modules are integrated by using the top-down approach. The rest of the modules are put in the middle layer which is integrated with the top layer and bottom layer, forming a complete application software. This sandwich approach has the advantages of the top-down approach while writing dummy modules for the low-level module is not required.

(c) As a result of the testing, an integrated software should be produced.

(d) Please refer to Appendix B for a checklist on Link/Integration 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 PSC via the PAT, 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 system analysts/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 system analysts/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 Testing.

(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) The following methods are useful in designing test case in Function Testing:

(i) Equivalence Partitioning
This testing technique is to divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same. It assumes that all conditions in one partition will be treated in the same way by the integrated software. Therefore, only one test case is required to be designed to cover each partition. For example, an input field accepting integer values between -1,000 and +1,000 can be expected to be able to handle negative integers, zero and positive integers. The test data is partitioned into three equivalent acceptable set of values. In addition to numbers, this technique can also apply to any set

of data that is considered as equivalent e.g. file type. By using this technique, testing one partition is equivalent to testing all of them.

(ii) Boundary Value Analysis
Boundary value testing is a technique to find whether the software will accept the expected range of values (e.g. maximum, minimum, just-inside and typical values) and reject the out-of-range values (e.g. just-outside and error values), focusing on boundary/corner values of an input item/class. This technique will normally design test cases for the boundary values, and a test case for a value of acceptable range.

(e) Please refer to Appendix C for a checklist on Function Testing.

7.4 SYSTEM TESTING

7.4.1 Scope of Testing

System Testing is the process of testing the integrated software with regard to the operating environment of the system (e.g. 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/Integration Testing and the Function Testing.

For small scale projects which combine the Link/Integration 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 three levels of testing.

7.4.2 Activities, Documentation and Parties Involved

(a) Test Group to prepare a System Testing test plan, to be endorsed by the PSC via the PAT, 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 System analysts/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) 13 types of System Test are discussed below. It is not claimed that all 13 types will be mandatory to every application system nor are they meant to be an exhaustive list. To avoid possible overlooking, all of them should be explored when designing test cases.

(i) Volume Testing
Volume testing is to subject the system to heavy volumes of data under normal workload, 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 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. Examples on recovery testing are Disaster Recovery Testing and Backup and Restore Testing.

(v) Security Testing
Security testing is to test the adequacy of the security controls and procedures imposed in the system by attempting to violate them. Test cases are devised to subvert the imposed security checks and to search for security holes in existing programs. For example, testing should attempt to access or modify data by an individual not authorized to access or modify that data. Code review may be conducted on programs to detect any insecure codes such as hard coding of passwords.

(vi) Usability Testing
Usability testing is to test the system in respect of its ease to understand, learn and use; and its capability in achieving specified goals in a specified context of use. The specified goals can be measured in relation to effectiveness, efficiency and user satisfaction.

(vii) 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, database administrator, or terminal user, should be tested during the System test.

(viii) 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.

(ix) 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.

(x) Scalability Testing
Scalability testing is to test the ability of the system to meet future efficiency requirements that may be beyond the current performance requirement.

(xi) Installation Testing
Installation testing is to test the validity of written installation procedures for installing the system onto the target environment i.e. to verify if the installation procedures are accurate and with understandable instructions.

(xii) Compatibility (Co-existence) Testing
It focuses on testing the compatibility of the system with other co-existing software such as operating system or web browser which may affect the behavior (e.g. resources usage) and stability of the system in the same environment (e.g. on same hardware).

(xiii) Adaptability Testing
Adaptability testing is to test whether the system can function correctly in all intended target environments with various specified components such as hardware, software and operating system. These environments are tested using a selection of functional test cases according to a predefined test procedure to find out any faults that may be introduced in adapting the software to a different environment.

(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 deferred to later stages. Such deferral may be acceptable 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 System 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 acceptable to its users and meets the business requirements.

7.5.2 Activities, Documentation and Parties Involved

(a) Test Group to prepare an Acceptance Testing test plan, which is to be endorsed by the PSC via the PAT.

(b) Test Group to prepare an Acceptance Testing test specification, which is to be endorsed by the PSC via the PAT.

(c) Test Group, with the aid of the project team, to set up the testing environment;

(d) Test Group to perform testing according to the Acceptance Test Plan; and upon fault found issue test incident reports to the system analysts/programmers, who will fix up the liable error; and

(e) Test Group to report progress of the Acceptance Testing through periodic submission of Acceptance Testing Progress Report.

(f) IPM to keep the overall Test Summary Report as documentation proof.

7.5.3 Practical Guidelines

(a) In general, Acceptance Testing conducted by the business users is often called "User Acceptance Testing". The objective is to ensure that system satisfies the acceptance criteria stated on the requirement document before releasing to daily operational environment. Sufficient effort and involvement of both the project team and end users in User Acceptance Testing is of paramount importance in ensuring the quality of the system. The business impact and the cost and time for fixing a production problem after live run are higher than that during the acceptance stage. Spending more effort in User Acceptance Testing is not only a prudent approach in delivering quality services, but will also reduce the overall costs and impact at the maintenance stage.

(b) Business users' involvement is essential

User Acceptance Testing should be performed by business users to prove that a new system works according to their understanding of their business requirements. Business users have the necessary knowledge and understanding of business requirements that IT testers may not have. It is therefore essential to get the

business users involved in the testing and not rely only on IT professionals. Sufficient resources on business users shall be planned and allocated to conduct the testing.

(c) Effort of business users

User Acceptance Testing for projects, which have relatively clear requirements during development stage, normally account for about 5% to 10% of the project implementation effort. For those projects with complicated requirements that are not easy to be specified clearly during development stage, the effort required for User Acceptance Testing may be up to 20% of the project implementation effort.

The effort required by business users can be estimated based on the number and complexity of test cases required to verify the system against the business requirements. Detailed test cases may only be available at a later stage of the project, normally after System Analysis and Design. Nonetheless, the effort can be estimated based on the following information when available:

(i) the number of business processes and their level of complexity, which would determine how much time is required to go through those processes and validate the acceptance criteria with the system;
(ii) the number of functional requirements and their level of complexity (can be measured by number of data sets required for a function), which would determine how much time is required to go through those functions and validate the acceptance criteria with the system;
(iii) the number of inputs and outputs (screens, reports, interfaces, etc.) and their level of complexity (can be measured by no. of fields on screens, reports, etc, and their validation requirements), which would determine how much time is required to go through those inputs and outputs and validate the acceptance criteria with the system;
(iv) the total number of test cases, the complexity and the time required for completing these test cases based on the parameters mentioned above.

The number of rounds of the testing to be conducted shall also be considered for the total effort required. Two to three rounds of testing are normally considered the minimum. More rounds of testing shall be considered if the business requirements are complicated or the quality of the development team is less assured.

(d) Following the User Acceptance Testing, an "Operational Acceptance Testing" is conducted by computer operators to ensure that the system meets the operational requirement and provide confidence that the system will work properly before deployment.

(e) For large-scale application systems, or systems involving new business or involving a large number of new users, it may be better if additional user testing is performed prior to the User Acceptance Testing. They are generally named as follows:
(i) Alpha Testing
The testing is taken place at the development environment by some end users. Developers can observe problems while users are testing the system before

deployment of the system to user testing environment for User Acceptance Testing.

(ii) Beta Testing
The testing is taken place at the user's testing environment by users. The objective of this testing is to determine the user satisfaction on the system and the fitness within the operational environment.

(f) Precautions for IPM and project team

(i) 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

(g) Precaution for users

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

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

8. TEST DOCUMENTATION

8.1 INTRODUCTION

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

For each project

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

  For each of the four levels of testing (i.e. Link/Integration, Function, System 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 and test specification as for Link/Integration Testing, Function testing, System Testing could be combined.

8.2 TEST PLAN

8.2.1 Purpose of Document

To prescribe the scope, approach, resources, and schedule of 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 test plan should provide at least the following information:

(a) Test Items
List the functional items and software features to be tested.
For Link/Integration 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 System 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) Testing 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 and 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	#	XXXXX
	-Test cases	#	XXXX
	-Test data	#	XXX
	-Testing 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:

(i) Error Reporting procedures;
(ii) Change / Version control procedures of S/W modules; and
(iii) Set-up and Wind-down procedures of the testing environment.

(b) Testing Environment
To specify at least the following items that are required in the testing progress:

(i) H/W and System S/W required;
(ii) Number of terminals/personal computers required;
(iii) Test facilities / tools required;
(iv) Test database; and
(v) 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 case, state its objective, specify also the steps and any special procedural requirements (e.g. bring up screen, input data, keys pressed etc.), the failure/pass criteria, the expected outputs (e.g. message displayed, file changes, etc.), programs involved, inter-case dependencies 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 test process which requires investigation. The report is to be issued to the system analysts/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 versions/revision levels.

(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
1. Design / Program error
2. Error related to testing environment
3. Others

(ii) Action taken
1. Design / Program changes.
2. Testing Environment Changes
3. No action taken.

Sample Test Incident Report

Test Incident Report

TIR NO. : _____ Date : _____
Reported By : _____
_____

Incident Description

_____
_____
_____
_____

Impact

_____
_____
_____
_____

Results of Investigation
Investigated By _____ Date _____

Classification	Actions taken
() design/pgm error	() design/pgm changes
() error related to testing environment	() testing environ' changes
() others	() no action taken

8.5 TEST PROGRESS REPORT

8.5.1 Purpose of Document

In order that progress of the test process is controlled properly, a periodic test progress report should be prepared by the Test Group, and submitted to the PAT / IPM.

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

8.5.2 Terminology

No. of test cases specified (#1): the total number of test cases that have been specified.

No. of test cases tried at least once (#2): the number of the specified cases that have been put into test execution at least once.

The percentage of #2 over #1 gives the percentage of the specified test cases that have been executed at least once. More importantly, the complement of this percentage gives the percentage of the specified test cases against which no test runs have ever been put so far.

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

8.5.3 Outline of Document

(a) Sample 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 and 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 provide information for future test 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/Integration 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 seven activities in the sequence indicated:

1. Identification and inclusion of the testing activities in the project plan;

2. Setting up of the Test Group;

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 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; and

7. Set up of the testing environment.

9.2 TEST CONTROL

Test control is an activity that runs continuously during the test process to ensure that all planned tests are complete and test results are able to meet the test objectives, strategies and mission. Test control generally includes the comparison between the planned progress and the actual progress, application of appropriate corrective actions and updating of the test planning activities as necessary.

Some test control actions that may be performed during the test process are:

(i) decision making based on information gathered and reported in test activities;
(ii) resetting priority of tests when identified risk occurs e.g. late delivery of programs to be tested; and
(iii) rescheduling of test activity because of late availability of the test environment.

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.

10.3 STATIC ANALYSERS

These are the tools to check about a program's structure, complexity and maintainability.

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.

11. SUMMARY

11.1 TEST 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 and procedures	Program Specification	Correct data then with flawed data	Programmer	Lead Programmer + System Analyst
Link/ Integration Testing	Linkages/ interfaces between program modules	Program Specification + System Specification	Control and data interface, returns/calls	Test Group	PSC recommended by PAT
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	Test Group	
Acceptance Testing	Entire system simulated in production environment	User requirements/ acceptance criteria	Live data where possible	Users	

11.2 TESTING ACTIVITIES IN SYSTEM DEVELOPMENT LIFE CYCLE

	System Analyst	Programmer	Test Group	User	Computer Operator	PAT
Feasibility Study						
System Analysis	- Perform Requirement Review		- Prepare Project Plan			
System Design	- Include test guidelines in Design and Program Specification - Perform Design Review		- Prepare Link/Integration Test Plan - Prepare Function Test Plan - Prepare System Test Plan - Prepare Test Specification for each of the Link/Integration, Function and System Testing - Set up testing environment	- Prepare Acceptance Test Plan (Project team to assist user to prepare the Acceptance Test Plan) - Prepare Acceptance Test Specification		- Recommend endorsement of the Link/ Integration, Function, System and Acceptance Test Plan - Recommend endorsement of the Test Specification
Program Development		- Perform Program Walkthrough - Include extra test cases - Perform Unit Testing				
System Integration and Tests			- Perform Link/Integration 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		

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 or updated 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 or control scripts 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/INTEGRATION 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 SYSTEM 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 transactions.

Stress Testing

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

Performance Testing

1. The response times are 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 an 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 is powered off when an update transaction 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 OUTSOURCED SOFTWARE DEVELOPMENT

1. Tailor and suitably incorporate the following in the tender specification or work assignment brief as appropriate.

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. Review and accept the different types of test plan, test specifications and test results.

4. Wherever possible, ask if there are any tools (ref. Section 10) to help demonstrate the completeness and test the coverage of the software developed.

5. Perform sample program walkthrough.

6. Ask for a periodic test progress report. (ref. Section 8.5)

7. Ask for the contractor's contribution in preparing the Acceptance Test process.

8. Ask for a Test Summary Report (ref. section 8.6) at the end of the project.

9. If third party independent testing service is required for a particular type of testing (e.g. system testing), define the scope of the testing service needed and state the requirements of the testing in a separate assignment brief or service specification for procurement of the independent testing services. Such service should be separately acquired from the procurement of software development. Besides, the service requirement for support and coordination with the Independent Testing Contractor should be added to the software development tender or work assignment brief.

APPENDIX G LIST OF SOFTWARE TESTING CERTIFICATIONS

1. Software Certifications by the Quality Assurance Institute (QAI) Global Institute[4]

   QAI was established in 1980 in Orlando, Florida in U.S.A. It provides educational and training programs for development of IT professionals in different aspects including software quality assurance and testing. QAI issues the following certifications that qualify professional software testers and test managers:
   (i) Certified Associate in Software Testing (CAST)
   (ii) Certified Software Tester (CSTE)
   (iii) Certified Manager of Software Testing (CMST)

2. International Software Testing Qualifications Board (ISTQB) [5]

   Founded in 2002, ISTQB is a not-for-profit association legally registered in Belgium. It aims to create a complete set of concepts on software testing which allows testers to get certification. It issues the following certifications:
   (i) Certified Tester Foundation Level
   (ii) Certified Tester Advanced Level – Test Manager
   (iii) Certified Tester Advanced Level – Test Analyst
   (iv) Certified Tester Advanced Level – Technical Test Analyst
   (v) Certified Tester Expert Level – Test Management

3. BCS, The Chartered Institute for IT (formerly named as British Computer Society) [6]

   It was founded in 1957 aiming to promote the study and practice of information technology for benefit the public. In March 2010, the ISTQB UK Testing Board and BCS, The Chartered Institute for IT became partners in providing software testing examinations at Foundation and Advanced level. It also provides other certificates such as the Intermediate certificate for software testing.

4. National Education Examinations Authority of the People's Republic of China[7]

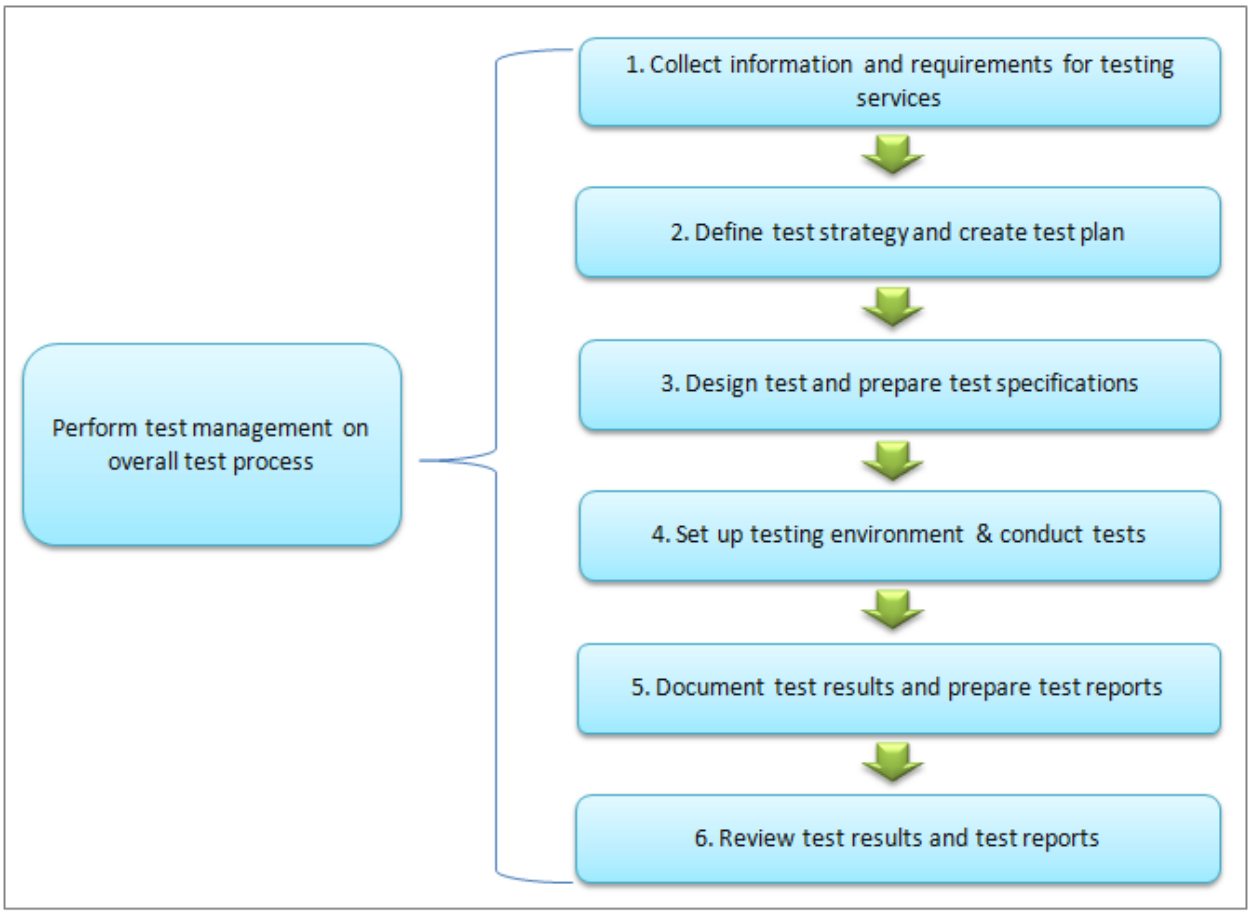
   National Computer Rank Examination (NCRE, 全國計算機等級考試), is authorized by the Ministry of Education and held by the National Education Examinations Authority. It is used to examine the knowledge and skill about computer application according to the national computer rank system. It has provided the following software testing examinations:
   (i) 3rd Level – Software Testing Skill (軟件測試技術)
   (ii) 4th Level – Software Testing Engineer (軟件測試工程師)

[4] Reference website: http://www.qaiusa.com
[5] Reference website: http://www.istqb.org/about-istqb.html,
[6] Reference website: http://certifications.bcs.org/category/15582
[7] Reference website: http://sk.neea.edu.cn/

APPENDIX H – INDEPENDENT TESTING SERVICES

Independent Testing Services Activities

Independent testing services generally include the following testing activities:



Perform test management on overall test process:

(i) Manage test plan and test procedures;
(ii) Schedule and allocate resources to support testing services;
(iii) Organise regular meetings with project team and users to report the testing progress, discuss and resolve issues/problems, present test results and reports; and
(iv) Coordinate project team with users for test items and activities throughout the test process.

1. Collect information and requirements for testing services:
   (i) Collect relevant information such as user requirements, security requirements, system specifications and architecture design to understand the system and user needs;
   (ii) Conduct workshops / interviews with project team and users to identify and collect the detailed user requirements on testing services according to the service requirements specified in the assignment brief or service specification; and
   (iii) Identify the test items i.e. which features are to be tested, and what are going to be validated.

2. Define test strategy and create test plan:
   (i) Define the scope, goals and objectives of testing and test strategy based on requirements, project schedule, budget, resources, identified risks, etc;
   (ii) Define overall test process, approach to be taken and deliverables to be produced;
   (iii) Estimate the test effort and resources required; and
   (iv) Create a high-level master test plan and detailed test plans as necessary.

3. Design test and prepare test specifications:
   (i) Define what needs to be tested in detail by examining the requirements or items to be tested, and identify any constraint for the test and the pass/fail/ending criteria for the test;
   (ii) Design test procedures, test cases with different scenarios, test data covering all test scenarios and testing environment;
   (iii) Prepare test specifications to document the details of test; and
   (iv) Review and align with project team and/or users on test specifications especially for the test cases.

4. Set up testing environment and conduct tests:
   (i) Set up the testing environment;
   (ii) Schedule the tests;
   (iii) Create sample test data and load test data to the testing environment;
   (iv) Conduct tests according to the schedule and test specifications;
   (v) Complete the tests, record the test results and create test incident reports if any; and
   (vi) Re-run the tests on failed cases after rectification by project team until all cases have passed the testing.

5. Document test results and prepare test reports:
   (i) Analyse test results, prepare test progress reports and test summary reports.

6. Review test results and test reports:
   (i) Review test results and reports to ensure completeness of tests, covering all defined test cases; and
   (ii) Ensure the validity of the test results.

Common Types of Independent Testing Services

(a) Unit and Link/Integration Testing
   (i) Black-box / White-box Testing
   Test the functionality of an application and application's internal structure.
   (ii) Code review
   Check and verify the quality of program codes to ensure that the codes are written properly and securely.
   (iii) Coverage Testing
   Ensure the percentage of test coverage is high enough to reduce the likelihood of occurrence of bugs / errors containing in the program.

(b) System Testing
   (i) Installation Testing

Ensure the system is installed and set up properly.

(ii) Operations Testing
Evaluate that the software application operates according to operational procedures.

(iii) Performance Testing
Conduct various tests such as stress test and load test to ensure that the system runs properly under the normal and abnormal loading, and meet the required performance level.

(iv) Regression Testing
Ensure that the original code is not affected by the changed code, and the system still works properly after the new changes are deployed.

(iv) Disaster Recovery Testing
Ensure that the data could be recovered in the disaster recovery site due to the hardware failure in production platform.

(v) Security Testing
Ensure that the security controls and measures imposed on the system can protect system data and programs against unauthorized access or security attacks by both internal and external parties.

(c) Acceptance Testing

(i) User Acceptance Testing
Develop user acceptance test plan, manage test process and conduct tests or assist users to conduct tests to ensure the end-to-end business processes are valid and capable to fulfill all business requirements.

Considerations for Selecting Parts of Testing to Outsourced Contractor

The project team should consider which parts of testing to outsource, including test levels, test types and test activities.

(a) Test Level
Programmers generally perform unit testing and link/integration testing by themselves during coding process. Therefore, it is sometimes not suitable for such testing to be outsourced to an external independent testing contractor especially if the development work is outsourced. This may lead to more time and effort for coordination and communication, and may affect the development progress.

User Acceptance Testing is required to be conducted in a production-like environment by users who are familiar with the business and know what they really want. Therefore, it is not suitable for the User Acceptance Testing to be conducted by external independent testing contractor who may not be familiar with the business and may not be capable to take up the user role to accept that the system satisfies all user requirements.

On the other hand, system testing is comparatively more suitable to be performed by an independent testing contractor. It is because system testing does not involve users and has less impact on development work. Besides, a separate testing environment can be set up and tests can be scheduled and conducted according to a well-defined test plan.

(b) Test Type
In general, both functional and non-functional system tests can be outsourced to an

independent testing contractor. Among these, test types requiring regular repetition and test automation is more suitable for outsourcing in order to save costs. Examples are function test, regression test and compatibility test. Test types such as performance test, load test and security test may also be outsourced to professional testing contractors who have specialized tools, techniques and expertise to perform the tests, on a case-by-case basis.

(c) Test Activities

Test activities within the test process such as test planning, test design, test execution and test reporting can all or partially be outsourced depending on the level of control to be released and knowledge to be given away. In general, all test activities within a test process will be outsourced for easier control and management of the testing contractor's services.

Considerations for Selecting Testing Contractors

(a) Experience in Software Testing

Service provider that has experience in providing same or similar type of testing services to clients for project of related business and/or similar project nature is preferred.

(b) Specialised Skill Set in Software Testing

Service provider whose core business is software testing is more preferred as the supplier is more able to provide specialized and professional staff and resources.

(c) Secured Environment for Protection of Test Deliverables

Service provider should be committed to high security protection as regards the physical storage of documents and compliance with non-disclosure requirements on sensitive test documentation and deliverables.

(d) Good Communication Skills of Testers

It is important that the testers of the service provider should possess good communication skills to effectively liaise between the project team and contractor throughout the test process, collect requirements and report testing progress and test results.

(e) Well Design Infrastructure for Testing

The technical infrastructure used by the service provider for software testing must be well designed to support comprehensive testing for the software products. It should consist of adequate hardware, software, network and other equipment such as PC, mobile devices, wireless devices, operation systems and different types of web browsers, etc.

(f) Standard Documentation

Service provider should have some testing guidelines and documentation established for use and control of the testing purpose.

(g) Qualifications of Testers

Staff of the service provider should preferably possess at least one valid and relevant qualified software testing certificate. Please refer to ***Appendix G*** for some examples of testing certifications issued by different organisations.

Roles and Responsibilities of Testing Contractors

The following shows an example of the roles and responsibilities of testers for independent test services.

Roles	Responsibilities
Test Manager / Test Specialist	(i) Schedule and assign duties to subordinates (ii) Plan and manage the test process (iii) Resolve technical and non-technical issues and disputes related to the test process (iv) Establish procedures and/or automated performance measurement capability to monitor the progress of testing (v) Liaise with project team and developer's team on day-to-day testing work (vi) Develop project management plans and quality control parameters
Test Coordinator	(i) Schedule and assign testing tasks to team members (ii) Assist in setting up the testing environment (iii) Co-ordinate with all working parties in projects (iv) Define the approach, methodology and tools used in testing (v) Perform quality control and quality assurance in test process (vi) Prepare test scenarios and produce documentation (vii) Provide support and troubleshoot problems in test process. (viii) Provide status of progress and defects to Test Manager (ix) Assure conformance to standards and test procedures
Test Lead	(i) Supervise and lead the work of testing in the same team (ii) Assign testing tasks to testers and assist in test execution when required (iii) Produce and maintain testing status documentation
Tester	(i) Conduct testing according to test procedures (ii) Produce and maintain test documentation