The Journal of the Quality Assurance Institute, 12(4), October 1998, 36-48.

A STRUCTURED APPROACH TO DEVELOPING TEST PLANS

by

Eldon Y. Li

College of Business
California Polytechnic State University
San Luis Obispo, California 93407
U.S.A.

Phone: 1-805-756-2964
Fax: 1-805-756-1473
E-Mail: eli@tuba.calpoly.edu
http://www.calpoly.edu/~eli

***Keywords*:** **Software testing, test plans, testing activities, structured process, systems development life cycle, project management.**

March 4, 1997

The Journal of the Quality Assurance Institute, 12(4), October 1998, 36-48.

A STRUCTURED APPROACH TO DEVELOPING TEST PLANS

ABSTRACT

This paper describes a two-stage structured approach to developing test plans during the system development life cycle (SDLC). The first stage is to identify the test plans required at the different phases of the SDLC and include them in the project test plan. The second stage is to develop the contents of these test plans based on 8 W's questions. The project test plan developed using this approach is relatively complete and comprehensive. It can serve as a reference book for the future test activities. Moreover, the format of the project test plan as proposed by this paper can be tailored to different project environments and organizational needs.

INTRODUCTION

Software testing is a critical process in a system development project. It is a necessary means for detecting the number of errors made during the analysis, design, and implementation phases. To have an effective software testing process, we must have a carefully prepared test plan for the process. The purpose of a test plan is to provide guidelines for a project team to plan for the test sequence, test criteria, test schedule, and required supporting resources such as personnel, hardware, software, test methods, test tools, test room, test facilities, and available budget. It is necessary for a project manager to use this plan along with the project (development) plan to allocate project resources and to manage and control the project activities.

Many system professionals (e.g., Aron [1983], Beizer [1984], Glass [1979], Larson [1975], Li [1985], Myers [1976, 1979], Perry [1983], Pressman [1982], and Wooldridge [1977], among others) had suggested various formats for a test plan. However, none of their formats can be adopted directly as a test plan for the structured testing process [Li, 1989]. Nor do they provide any guidelines for developing a test plan.

A structured testing process is a step-by-step procedure for assuring the quality of test plans, designs, execution, controls, and documentation throughout the entire system development process [Li, 1989]. A proper test plan in this process must address to all the questions of the 8 W's: what, who, which, when, why, where, how, and how much? To be more specific, it must address to the following questions: Where are we now? Where do we want to go? Why should we get there? How do we get there? When should we get there? Who will do it? How much does it cost? and How much should we spend? Table 1 enumerates the questions or activities related to testing under each category of "W." These questions are the essential components of an effective test plan. The purpose of this paper is to propose a structured approach to developing a comprehensive test plan in the system development life cycle (SDLC) which answers these 8 W's questions.

——— INSERT **Table 1** ABOUT HERE ———

THE STRUCTURED APPROACH

The structured approach to developing test plans for a system development project as proposed by this paper begins with a time schedule when different test plans should be ready, followed by developing the contents of the test plans.

The Time Schedule of Developing Test Plans

The first stage of our structured approach is to develop an outline of the project test plan and identify what specific test plans need to be included in this plan during the SDLC. The project test plan is the test plan for the entire system project. It consists of an overall test plan and several specific-level test plans, each prescribing guidelines for different levels of testing activities such as module testing, integration testing, etc. The former should be developed before beginning the development of any specific-level test plan. As the project progresses, the project test plan will be revised, expanded, and updated. More and more details will be added into the test plan. That is, one should start developing the project test plan early in the SDLC; as soon as the project plan has been established, usually at the end of the project viability analysis (PVA) phase. During the rest of SDLC, other test plans and test case specifications will be developed, and different levels and types of testing will be conducted. The resulting documents will be added to the project test plan. The description of the sequence of developing test plans during the SDLC follows. Figure 1 shows a Gantt chart summarizing the flow of various test plans and testing activities.

——— INSERT **Figure 1** ABOUT HERE ———

1. **System Requirements Definition (SRD) Phase:** During the SRD phase, the questions common to all levels or types of testing such as the testing goals, objectives, strategies, accountabilities, procedures, productivity tools, schedule, resource needs, etc., should be added to the project test plan. The dates when each specific level of testing starts and ends are roughly estimated; so are the percentage of resources allocated to each of them. As the project progresses, the schedules and resource allocations of the completed test activities will be updated with the actual time and resource spending. This will help us in revising the allocations to the unfinished test activities. Therefore, there will be several versions of test milestones and the personnel requirements included in the project test plan. In summary, the project test plan will be periodically revised, expanded, and updated throughout the entire SDLC. Each update of the project test plan will increase its details by adding another test plan for a different level of testing or more information about testing activities.

2. **System Design Alternatives (SDA) Phase:** An initial schedule for various specific levels of testing activities is added to the project test plan. Resource needs are updated during this phase as well. At the end of the SDA phase, an initial project test plan containing an overall test plan is completed.

3. **System External Specifications (SES) Phase:** At the end of the SES phase, a system test plan including the system acceptance criteria should be in place.

4. **System Internal Specifications (SIS) Phase:** Following the system test plan, test plans for the acceptance testing (including both software acceptance testing and final acceptance testing), the integration testing, and the module testing should be established during the SIS phase.

5. **Program Development (PD) Phase:** Initially in each test plan, test cases and a test-case summary should be developed. The summary should stay relatively stable, i.e., no changes would be made unless a major error in design is detected. The test case specifications for the module testing and the integration testing will be developed during the PD phase. They will be included in the corresponding test plans. These two types of testing would be conducted during this phase. Note that each test case specification may include multiple sets of test items and data. Toward the end of the PD phase, test case specifications for the system testing and user procedures for running the system should be developed and ready for use.

6. **Testing Phase:** As soon as the system testing begins, test case specifications for the software acceptance testing will be prepared. Meanwhile user training program is designed and set up. Test plans for the conversion testing and the installation testing should be developed in sequence after the integration testing is completed and the system testing is in progress. Following the system testing, the design of test cases and data, and the preparation of testing site, one can begin the software acceptance testing. Portion of the system testing and the entire acceptance testing can be performed using the parallel processing approach.

7. **Conversion Phase:** For the conversion testing, test case specifications will be prepared as soon as the conversion program design logic has been defined. User training should begin so as to prepare users for the final acceptance testing.

8. **Installation Phase:** Furthermore, test case specifications for the installation testing will be derived once the conversion testing has been completed. As for the final acceptance testing, test case specifications will be derived along with the installation testing activities. Note that any test case specifications created during the course of testing should be included in their corresponding test plans. Once the installation phase is completed, the project test plan should contain all the test-related documents. It is a complete and comprehensive reference book for future testing activities. A proper index should be created for quick reference purpose.

THE CONTENTS OF TEST PLANS

Following the time schedule of developing test plans, the second stage of the structured approach is to develop the contents of each of these test plans by answering the aforementioned 8 W's questions. The description of this process follows.

1. **Where are we?** To answer this question, project test plan begins with a "Background" section which introduces to the reader the existing corporate guidelines and standards for project testing. This section describes method of testing, method of evaluation, available test materials, and supporting documents needed by the project. The supporting documents

include not only the related corporate guidelines and standards, but also the references and project documents related to the test activities to be conducted. These guidelines and standards should be applied and tailored to all levels of testing activities during the entire SDLC. Similarly, a test plan for a specific level of testing should begin with an background section which describes the specific related corporate guidelines and standards that will be used in each testing activity.

2. **Where do we want to go?** Following the background section, the objectives of the project test plan will be described in the overall test plan section. A brief description of the system to be tested follows. Similarly in the specific-level test plan, an objectives section and a software system section should be provided.

3. **Why should we get there?** In the statement of an objective, the reason or importance of this objective should be briefly described.

4. **How do we get there?** How do we conduct the tests? How do we know they are acceptable? To answer these questions, we should have a section specifying the method(s) of testing and evaluation to be used in the entire project, and similarly, in each specific level of testing. Furthermore, the criteria for beginning and ending a test common to all the test activities in the entire project should be described in the overall test plan. Likewise, in the specific-level test plan, there should be a criteria section describing the criteria of starting and stopping a test which are unique to that particular level of testing activities. Finally, a section describing the procedure of controlling the test activities should be included in the project test plan.

5. **When should we get there?** In the overall test plan, the start and end date of each level of testing in the entire project should be estimated and plotted on a schedule network or a Gantt chart. The location where each testing activity is conducted should be specified. The quality control checkpoints for the testing activities should be identified in terms of time, date, and objectives of the checkpoint. Similarly, in the specific-level test plan, milestone should be specified for each testing activity.

6. **Who will do it?** This question relates to the personnel requirements for all the testing activities during the entire project life cycle. The skill types and the number of personnel required by each testing level should be identified and acquired. The available schedules and the accountabilities of users and project team members should be specified in the overall test plan. Similarly, the personnel requirements for each specific-level testing activity should be specified in each specific-level test plan.

7. **How much does it cost?** The resource requirements other than personnel should also be specified in the overall test plan and the specific-level test plans. These include hardware, software, and data or files needed for each testing level and each testing activity. At the end of each testing activity, one must answer "How much did it actually cost?" Any testing activity which consumed the time or resources more or less than the control limits should be specially reported in the project review meetings. Corrective actions should be taken to improve estimating the required resources or to improve the testing process.

8. **How much should we spend?** The resource requirement section in project test plan and each specific test plan should include the availability and constraints of the resources.

Following the above process, a project test plan could be properly derived. The table of contents of one such possible project test plan is shown in Table 2. This table is adapted from the works of J. D. Aron's (1983) and E.Y. Li (1985, 1990). Its format is designed to serve as a framework for all varieties of test plans. It is relatively complete and comprehensive. As a project manager, one should tailor this table to one's own particular project environment.

——— INSERT **Table 2** ABOUT HERE ———

CONCLUSION

Software testing is an important process within a quality assurance program. Since it is not a constructive process like design and programming, it should be kept at the minimum level. To do so, one needs proper test plans to effectively guide the testing activities throughout the entire SDLC. The structured approach proposed in this paper identifies all the test plans required in the SDLC and develops the contents of the test plans based on a set of structured questions: the 8 W's. This approach is well structured and relatively easy to implement. The project test plan developed using this approach is relatively complete and comprehensive. It can serve as a reference book for future testing activities. It is recommended that a project manager should tailor the format of the project test plan proposed in this paper to his or her own particular project environment and organizational needs.

REFERENCES

[1] Aron, J.D. *The program Development Process: The Programming Team, Part II*, Reading, MA.: Addison-Wesley, 1983.

[2] Beizer, B. *System Testing and Quality Assurance*, New York: Van Nostrand and Reinhold, 1984.

[3] Glass, R.L. *Software Reliability Guidebook*, Englewood Cliffs, N.J.: Prentice-Hall, 1979.

[4] Larson, R.R. "Test Plan and Test Case Inspection Specifications," TR-21.586, IBM System Development Division, Kingston, N.Y., 1975.

[5] Li, E.Y. "Structured Software Testing: An Introduction," Working Paper Series, Center for Business and Economic Research, California Polytechnic State University, San Luis Obispo, California, 1985.

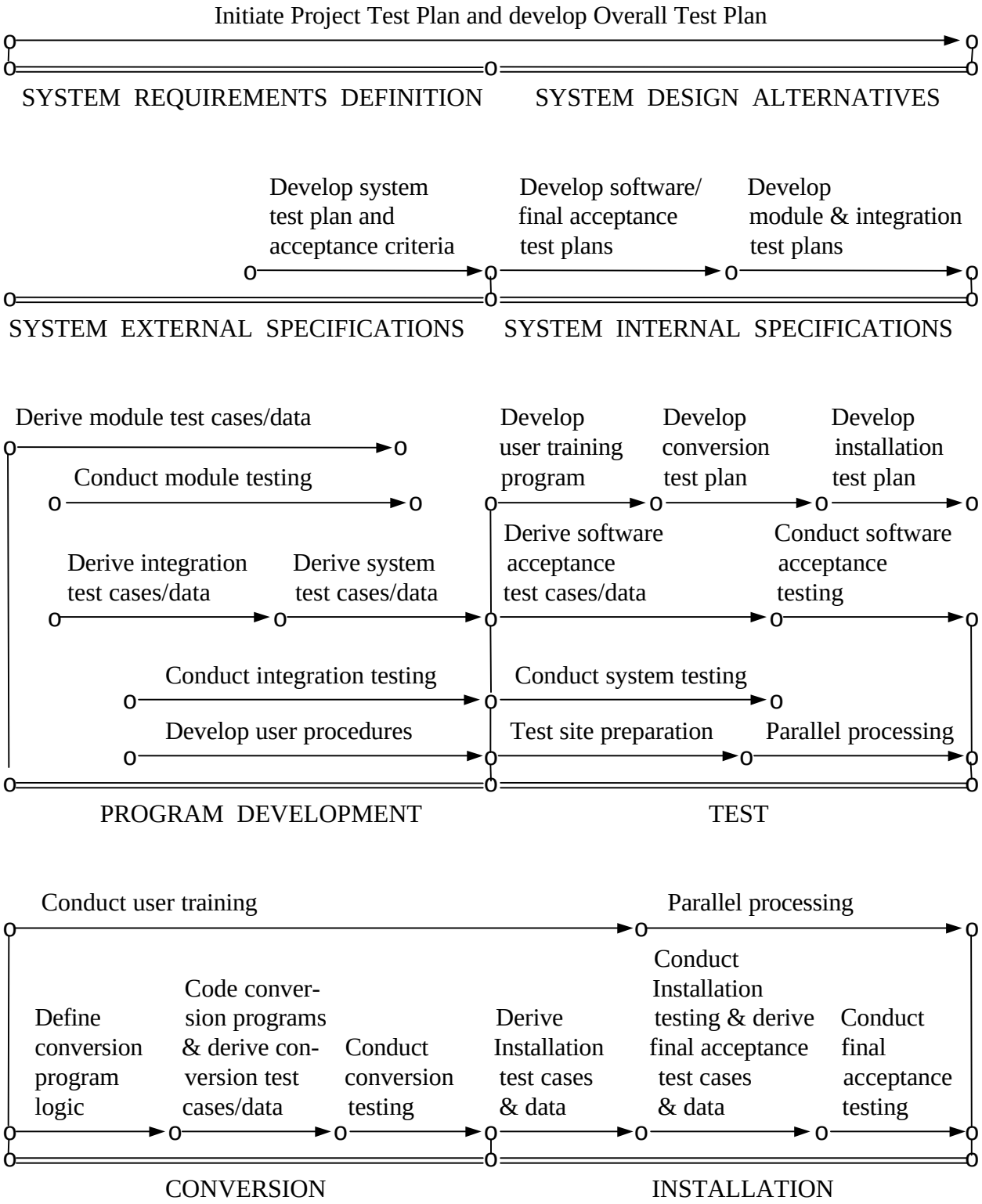
[6] Li, E.Y. "Structured Testing in the System Development Life Cycle," *Quality Data Processing*, Vol. 3, No. 3, July 1989, pp. 11-19.

[7] Li, E.Y. "Software Testing in a System Development Process: A Life Cycle Perspective," *Journal of Systems Management*, Vol. 41, No. 8, August 1990, pp. 23-31.

[8] Myers, G.J. *Software Reliability: Principles and Practices*, New York: Wiley-Interscience, 1976.

[9] Myers, G.J. *The Art of Software Testing*, New York, N.Y.: Wiley-Interscience, 1979.

[10] Perry, W.E. A *Structured Approach to Systems Testing*, Wellesley, MA.: QED Information Sciences, 1983.
[11] Pressman, R.S. *Software Engineering: A Practitioner's Approach*, New York: McGraw-Hill, 1982.
[12] Wooldridge, S. *Systems and Programming Standards*, New York: Petrocelli/Charter, 1977.



Note: Walkthrough/inspection and regression testing are not shown in this diagram. While the former is human-based and being conducted throughout the entire development life cycle, the latter is computer-based and being conducted only after the module testing has begun.

Source: Adapted from Li [1990].

Figure 1: The Flow of Test Plans and Test Activities

Table 1. The Questions Addressed by a Test Plan

CATEGORY	QUESTIONS/ACTIVITIES
1. What?	1.1. What will be tested? 1.2. What will not be tested? 1.3. What are the assumptions? 1.4. What is the test objective? 1.5. What test cases do we need? 1.6. What are the required test data/files? 1.7. What are the required hardware/software resources? 1.8. What is the testing environment? 1.9. What is the test procedure? 1.10. What documents can we keep?
2. Which?	2.1. Which test should go first? 2.2. Which document should we keep?
3. Who?	3.1. Who is responsible for conducting a test? 3.2. Who is responsible for designing test cases? 3.3. Who is responsible for providing test data? 3.4. Who is the test coordinator or supervisor? 3.5. Who will record test results? 3.6. Who will compile test run binder? 3.7. Who will assess test results?
4. When?	4.1. When will each test start and end? 4.2. When will each test level start and end? 4.3. When will each test task start and end? 4.4. When will each test be performed?
5. Why?	5.1. Why is the test performed? Is it for: 5.1.1. Functional (black-box) testing: ● auditability test — Does the system provide audit trails and meet other auditing standards? ● correctness test — Does the system have correct logic to deliver accurate output information? ● error-handing test — Can errors be prevented or detected, and then corrected? ● integrity test — Can the system uphold data and file integrity? ● inter-systems test — Can data be passed correctly from system to system? ● manual support test — Does the people-computer collaboration work? ● parallel comparison test — Are there unplanned differences between the old and

	the new systems? ● regression test — Do the unchanged system components still perform correctly with the changed component? ● requirements test — Does the system perform as specified? ● system controls test — Do the controls reduce system risk to an acceptable level? ● timeliness test — Can the system deliver desirable information in a timely manner? ● usability test — Are the system and the documentation user-friendly? 5.1.1. Structural (white-box) testing: ● compliance test — Is the system developed in accordance with standards and procedures? ● configuration test — Does the system work under minimum and maximum configurations? ● documentation test — Is the user and system documentation complete and accurate? ● maintainability test — Is the system easy to maintain and well documented? ● operations test — Can the system be executed in a normal operational status? ● performance test — Does the system achieve desired level of performance or efficiency under certain workload and configuration conditions? ● portability test — Is the system compatible, installable and maintainable? ● recovery test — Can the system be returned to its previous operational status after a failure? ● reliability test — Does the system meet the reliability objective? ● security test — Is the system protected in accordance with it level of importance to organization? ● storage test — Does the system have enough main and secondary storages? ● stress test — Can the system perform with peak load over a short span of time? ● volume test — Can the system perform with high volume of I/O and communications over a long span?
6. Where?	6. 1. Where are we now? (or what is the project status?) 6. 2. Where are we from? (or what are the completed testing activities?) 6. 3. Where are we going to? (or what are the remaining testing activities? 6. 4. Where is the testing location? 6. 5. Where are the checkpoints?
7. How?	7. 1. How will the test be performed? Discuss: ● structural and functional testing strategy. ● integration and regression testing strategy.

	● test progression and schedule. ● test tools to be used. ● test case design methods. ● job stream control. ● problem reporting and resolution. 7. 2. How will the test be evaluated? Is it to: ● use system acceptance criteria? ● inspect test materials? ● review test materials? ● walkthrough test materials? ● use exit criteria?
8. How much?	8. 1. How much testing do we need? (or what are the exit criteria?) 8. 2. How much documentation do we keep? 8. 3. How much manual support do we have? 8. 4. How much testing effort (time and resources) should we spend? 8. 5. How much testing effort (time and resources) did we actually spend?

Table 2: Recommended Table of Contents of a Test Plan.

PROJECT TEST PLAN

TABLE OF CONTENTS*

Section 1. Background

1.1. Method of Testing
1.1.1. General procedure for application software testing
1.1.2. Levels of tests --- module, integration, system, etc.
1.1.3. Structural and functional testing strategy
1.1.4. Integration and regression testing strategy
1.1.5. Test case development, validation, and verification
1.1.6. Job stream and scenario control that isolates problems to specific test case
1.1.7. Problem reporting and resolution
1.1.8. Test-case design --- white-box and black-box techniques.
1.1.9. Test methods --- inspection, review, walkthrough, desk checking, machine tests, etc.
1.1.10. Test tools --- profilers, file comparers, simulators, test harnesses, auto-runner, etc.

1.2. Method of Evaluation
1.2.1. Types of structural test to be evaluated --- compliance, configuration, documentation, maintainability, operations, performance, portability, recovery, reliability, security, storage, stress, volume
1.2.2. Types of functional test to be evaluated --- accuracy, auditability, timeliness, usability, system controls, error-handing, inter-systems, manual support, parallel comparison, regression, requirements
1.2.3. Evaluation criteria to be used

1.3. Test Materials --- sample documents should be provided
1.3.1. Test case binder

- Test cases summary
- Complexity-based coverage worksheet
- Equivalence partitioning worksheet
- Final test case/data worksheet
- Test case/data specifications
- Test scenarios/scripts
- Test case/function matrix

- Calling/called modules matrix

1.3.2. Test run binder

- Test results summary
- Test activity log
- Discrepancy/fix reports
- Discrepancy summary
- Test summary report

1.3.3. Test control binder

- Walkthrough/Inspection checklists
- Program test cases checklist
- Screen test cases checklist
- Improvement suggestion forms
- Change control forms

1.3.4. Operator procedures and training outlines

1.3.5. User procedures and training outlines

1.3.6. Other desirable documents

1.4. Supporting Documents

1.4.1. References

1.4.2. Attachments --- related corporate guidelines/standards, related project documents

Section 2. Overall Test Plan

2.1. Objectives (of the project test plan)

2.1.1. Things that will be tested

2.1.2. Things that will not be tested as part of this plan --- third-party software, date entries, installation, readiness, unique user characteristics, etc.

2.2. Software System Description --- a brief description of the system to be tested

2.2.1. System components --- DFD, HIPO, structure charts, etc.

2.2.2. Other components relevant to the tests

2.2.3. General characteristics of the environment in which the system will be used

2.3. Methods --- describe the methods to be used.

2.3.1. Method of testing (see Section1.1)

2.3.2. Method of evaluation (see Section1.2)

2.4. Milestone --- test network/schedule, sites, and checkpoints

2.5. Common Criteria
2.5.1. Entry criteria --- clean compiled, desk checked, etc.
2.5.2. Exit criteria --- error-to-statement ratio, path coverage, budget usage, time schedule, etc.

Section 3. Testing Requirements

3.1. Personnel
3.1.1. Constraints --- available personnel
3.1.2. Skill types and number of personnel
3.1.3. Available schedules
3.1.4. Accountabilities
3.1.5. Other special requirements

3.2. Hardware
3.2.1. Constraints --- available schedule
3.2.2. Configuration required for developing test cases
3.2.3. Configuration required for simulating test environments: host, terminals, communication links, load generator
3.2.4. Configuration required for user acceptance tests
3.2.5. Usage schedule: batch, real-time, switched/dedicated

3.3. Software
3.3.1. Constraints --- available configurations and software
3.3.1. Various configurations under which the system will be tested
3.3.2. Operating system
3.3.3. Communication subsystems
3.3.4. Network load simulation software
3.3.5. Environment simulation software
3.3.6. Measurement software
3.3.7. Test-aid software
3.3.8. Schedule for availability of each software system's generation

3.4. Data/Files --- types and availability, including old test cases and test data

Section 4. Procedure Control

4.1. Test initiation

4.2. Test execution

4.3. Test completion

4.4. Test failure

4.5. Access control

4.6. Change control

4.7. Document control

4.8. Backup and recovery

(*Note that the overall test plan ends here. The following part of table of contents is for specific test plans such as module test plan, integration test plan, etc..)*

* Sections 1 through 4 are directed to the overall project testing. Section 5 is the test plan for a specific level of testing which include module testing, integration testing, system testing, conversion testing, installation testing, and acceptance testing. Each level of testing forms a new section of its own (i.e., Sections 5, 6, and so forth).

Section 5. Specific-Level Test Plan**

5.1. Background

5.2. Objectives

5.3. Software System to be Tested

5.3.1. Description of system (using tables, flowcharts, HIPO, DFD, etc.)

5.3.2. List of functions/modules to be tested

5.4. Methods --- describe the methods to be used
5.4.1. Method of testing (see Section 1.1.)
5.4.2. Method of evaluation (see Section 1.2.)

5.5. Milestone --- test network/schedule, progression, sites, and checkpoints.

5.6. Criteria --- list the entry and exit criteria in addition to the ones stated in the overall test plan.
5.6.1. Entry criteria
5.6.2. Exit criteria

5.7. Resource Requirements --- including constraints, estimates, and actual usage.
5.7.1. Personnel
5.7.2. Hardware
5.7.3. Software
5.7.4. Data/files

5.8. Resulting Test Materials --- as shown in Section 1.3 and including post-test documents

5.9. Execution Control
5.9.1. Are we ready to start the test?
5.9.2. How to carry out the scripted activities?
5.9.3. What to do when something goes wrong: test failure; test environment failure; security; recovery
5.9.4. Can problem be fixed without authorization?
5.9.5. What document needs to be kept and who does it?

5.10. Attachments --- corporate or project documents which are related to but not directly generated by the test process

** This section should be repeated for each level of testing , i.e., module testing, integration testing, system testing, software acceptance testing, conversion testing, installation testing, and final acceptance testing.