**4. TEST ACTIVITIES, MANAGEMENT, AND**

**AUTOMATI0N**

**1. TEST PLANNING AND PREPARATION**

Test planning and preparation is the most important activity in the generic testing process for systematic testing based on formal models. Most of the key decisions about testing are made during this stage. In this section, we first examine what key questions need to be answered in the high-level test planning, and then examine individual low-level activities related to test preparation. Test planning and test preparation are sometimes treated as separate groups of activities (Black, 2004).

**1.1 Test planning: Goals, strategies, and techniques**

The high-level task for test planning is to set goals and to determine a general testing strategy. This high-level decision should be based on answers to several key questions, particularly the objectives or goals of testing under a specific environment. The answers to these questions not only determine the general types of testing to perform, but also determine the test termination or exit criteria. Overall environment needs to be considered because the environmental constraints imposed on testing also affect the choice of testing strategies.

Most of the testing we cover in this book focuses on the correctness aspect of quality. If the software is complete or nearly complete, then the above correctness-centered quality goals can be directly translated into reliability goals, which, in turn, requires us to use usage-based statistical testing. Sometimes, these quality goals can be translated indirectly into coverage goals, which can be achieved by black-box testing for the whole system. However, if only individual units and pieces are available, we might choose to focus on the implementation details and perform coverage-based white-box testing.

Therefore, we set an overall testing strategy by making the following decisions:

* *Overall objectives and goals,* which can be refined into specific goals for specific testing. Some specific goals include *reliability* for usage-based statistical testing or *coverage* for various traditional testing techniques.
* *Objects to be tested and the specific focus:* Functional testing views the software product as a black-box and focuses on testing the external functional behavior; while structural testing views the software product or component as a (transparent) white box and focuses on testing the internal implementation details.

Once the overall testing strategy has be selected, we can plan to allocate resources and staff to implement it. The available staff and resources also affect the specific models and techniques that can be used to implement the strategy. For example, simple models based on checklists and partitions generally require less resources and prior knowledge by the testing staff, while more complex formal models and related testing techniques may require more resources and expertise. Different models and techniques are also associated with different effectiveness levels or different applicability to different types of problems and situations. Consequently, appropriate testing models and related techniques can be selected to optimize some form of cost-benefit measure.

Sometimes, existing models or test suites can be used with some minor modifications or adaptations, which would require minimal additional effort in test planning and preparation. Nevertheless, the above high-level activities still need to be carried out to arrive at this decision, because indiscriminately using exiting testing strategies, techniques, models, and test suites may not fulfill the need for the new situation and end up merely wasting valuable time and resources. In what follows, we focus on the situation where new models, procedures, and test cases need to be considered in testing planning and preparation. The situation of minor adaptations in connection with regression testing as a specialized type or testing.

**1.2 Testing models and test cases**

Different models are closely linked to different testing techniques, and the modeling details can only be described together with their corresponding techniques, as we will do in Chapters 8 through 11. However, some generic steps and activities are involved in test model construction. as follows:

1. *Information source identification and data collection:* The information and data are generally affected by both what is required by specific models and what is available in the project environment. For example, in usage-based statistical testing, information about actual in-field or anticipated usage by target customers needs to be gathered to construct operational profiles as the basis of testing; while in white-box unit testing, the tested unit provides the information source which can be analyzed to construct our testing models.

*2. Analysis and initial model construction:* The information and data collected above are analyzed to construct testing models. Expertise and familiarity with the specific testing techniques and models are required for people who perform this task. This step is typically the hardest to automate because of the human intelligence and expertise required.

*3. Model validation and incremental improvement:* This is an important step, particularly for large objects or for functions or usages associated with external customers. Iterative procedure might be necessary to fix inaccuracies and other problems found in the initial model or early versions of the candidate models.

Once the testing models have been constructed and validated, they can be used to generate test cases, which can then be executed by following some planned test procedure. First, we need to define and distinguish the static test cases and the dynamic test runs, as follows:

* **A** *test case* is a collection of entities and related information that allows a test to be executed or a test run to be performed.
* **A** *test* ***run,*** is a dynamic unit of specific test activities in the overall testing sequence on a selected testing object.

Each time a static test case is invoked, we have an individual dynamic test run. Therefore, each test case can correspond to multiple test runs. In some literature and organizations, each test run is also called an *attempt.*

The information included for a test case must enable the related test run to start, continue, and finish. For most of the testing situations, the starting and finishing points correspond to the initiation and termination of the operations for the whole software system, such as the compilation of a program when the compiler is tested. But there are exceptions, such as in operating systems and telecommunication systems, where continuous operation without stopping is the expected norm. In these cases, because the specific test is an activity associated with finite time for practical purposes, the starting and finishing points need to be artificially inserted, resulting in a subsection of the system execution as a test run.

Essential among the test case information is the specific input to the software object in operation, which includes both the initial input at the start of the test run and the input to allow it to continue and to finish. In addition, the test case often includes information about the expected output throughout the test run, which, together with the specific input and timing information, defines the program behavior under this test run. Such input, output, and timing information can be captured by the set of input variables, the set of output variables, and their values over time. With the above understanding, we can view the construction of a specific test case as assigning its input values over a planned test run, which is referred to as test *sensitization* in testing literature. This assignment is typically derived from testing models we constructed in the previous step of test planning and preparation. Different criteria and steps may be involved in test sensitization when different testing techniques are used.

**1.3 Test suite preparation and management**

The collection of individual test cases that will be run in a test sequence until some stopping criteria are satisfied is called a *test suite.* Test suite preparation involves the construction and allocation of individual test cases in some systematic way based on the specific testing techniques used. For example, when usage-based statistical testing is planned, the test case allocation will be determined by the operational profiles (OPs) constructed as the testing models, in proportion to individual usage probabilities. Similarly, when coverage-based testing is planned, the specific coverage criteria would dictate the allocation of test cases. For example, in control flow testing not involving loops, the exact number of test cases is determined by the number of paths for all-path coverage.

Another way to obtain a test suite is through reuse of test cases for earlier versions of the same product. This kind of testing is commonly referred to as *regression testing.* It ensures that common functionalities are still supported satisfactorily in addition to satisfactory performance of new functionalities. Special types of formal models are typically used to make the selection from existing test cases, in connection to regression testing.

In general, all the test cases should form an integrated suite, regardless of their origins, how they are derived, and what models are used to derive them. Sometimes, the test suite may evolve over time and its formation may overlap with the actual testing. In fact, in some testing techniques, test cases can be constructed dynamically, or “on-the-fly”, during test execution. But even for such testing, some planning of the test cases and test suite is still necessary, at least to determine the specific method for dynamic test case construction and the precise stopping criteria. For most of the testing techniques we cover in this book, a significant part of test preparation must be done before actual testing starts.

In general, test cases cost time, effort, and expertise to be obtained, and are too valuable to be thrown away. It is worthwhile to spend some addition effort and resource to save them, organize them, and manage them as a test suite for easy reuse in the future. Test suite management includes managing the collection of both the existing test cases and the newly constructed ones. At a minimum, some consistent database for the test suite needs to be kept and shared by people who are working on similar areas. Some personnel information can also be kept in the test suite, such as the testers who designed specific test cases, to better supported future use of this test suite. The information contained in the test suite constitutes an indexed database with important information about individual test cases in the test suite, as well as pointers to actual test cases. The actual test cases, in turn, contains more detailed information about the exact scenario, test input, expected output and behavior, etc.

There are many ways to organize the test suite or test suites. The most common way is to organize them by sub-phases, because of the different objects, objectives, concerns, perspectives, priorities, and the testing techniques used. Various attributes can be used to describe, classify, and organize individual test cases in the suite. One concrete example is the use of the following attributes for an IBM product in its system testing phase (Tian, 1998):

* sc - scenario class
* *sn* - scenario number
* *vn* - variation number with a particular scenario

The scenario class sc corresponds to high-level functional areas or groups of functions. Within each sc, the scenario number *sn,* and the variation number *vn* within each ***sn,*** form a three-layer hierarchical organization of test cases in the suite. In addition, *sn* and ***vn*** are generally ordered in rough correspondence to the expected execution order, ranging from **1** to 99, with consecutive numbers used up to a point and then skipping to 99 to indicate some ad hoc test cases - those do not fall into some systematic sequence. Therefore, less than 99 scenarios or variations within scenarios are allowed, which was more than adequate for the product tested.

**1.4 Preparation of test procedure**

In addition to preparation of individual test cases and the overall test suite, the test procedure also needs to be prepared for effective testing. The basic question is the sequencing of the individual test cases and the switch-over from one test run to another. Several concerns affect the specific test procedure to be used, including:

* ***Dependencies*** among individual test cases. Some test cases can only be run after others because one is used to set up another. This is particularly true for systems that operate continuously, where the later test run may need to start at a state set up by the earlier one.
* ***Defect detection*** related sequencing. Many problems can only be effectively detected after others have been discovered and fixed. For example, integration of several components and related testing typically focus on interface and interaction problems, which can be masked by problems in individual components. Therefore, these components need to be individually tested before integration testing starts.
* **Sequences to avoid *accidents****.* For some systems, possibly severe problems and damages may incur during testing if certain areas were not checked through related test runs prior to the current test run. For example, in embedded software for safety critical systems, one does not want to start testing safety features before testing other related functions first. This can be considered as a special case of the problem or defect related sequencing where there is a very strong economical incentive for preferring certain sequencing to others.
* ***Problem diagnosis*** related sequencing. Some execution problems observed during testing may involve complicated scenarios and many possible sources of problems. Under this situation, related test runs focused on a single aspect or limited areas can be used to help with the problem diagnosis. Better yet, if such complicated problems are expected, we should run related simpler test cases first to eliminate certain possibilities and narrow down the problem areas. Therefore, one natural sequence for test case execution commonly used in practical testing procedures is to progress from simple and easy ones to complicated and difficult ones. The same idea has been used in defining coverage hierarchies.
* ***Natural grouping*** of test cases, such as by functional and structural areas or by usage frequencies, can also be used for test sequencing and to manage parallel testing. However, among areas where no such order exists, or when the incentive for following a certain order is not strong, we can carry out testing for them in parallel to speed up the testing process. In fact, this is what people do all the time for large-scale software testing, where parallelism and interleaving are common.

The key to test run transition in the test procedure preparation is to make sure that the next test run can start right after the current one is finished for each software installment. This consideration may place some additional requirements on individual test cases, either requiring them to leave the system in the same initial condition or in some specified final condition. In fact, the initial and final states of specific test cases can also be used to group individual test cases in the test suite. This is similar to the grouping of test cases when system configuration and environmental setup are considered in defining the operational mode in usage-based testing using Musa’s operational profiles (Musa, 1998).

When test cases are derived dynamically, test procedure would naturally involve much more dynamic elements. However, the above considerations for test procedure preparation should still be incorporated in the corresponding test procedure. In this case, not only the execution but also the generation of dynamic test cases is affected by the dependency, effectiveness and efficiency concerns.

A related topic to test procedure preparation is the assignment of people to perform certain tests. Their roles and responsibilities need to be clearly specified. In addition, allocation of time and other resources also needs to be planned ahead of time before test execution starts, in accordance with test case grouping and allocation within a test suite. One specific type of resources is the test automation tools which could significantly reduce the time, staffing, and other resources required for test execution.