CHAPTER 12: SYSTEM TEST PLANNING AND AUTOMATION

STRUCTURE OF A SYSTEM TEST PLAN

- Test planning is essential in order to complete system testing and ship quality product to the market on schedule.
- Planning for system testing is part of overall planning for a software project.
- It provides the framework, scope, resources, schedule, and budget for the system testing part of the project.
- Test efficiency can be monitored and improved, and unnecessary delay can be avoided with a good test plan.

STRUCTURE OF A SYSTEM TEST PLAN

- The purpose of a system test plan is summarized as follows:
  - It provides guidance for the executive management to support the test project, thereby allowing them to release the necessary resources to perform the test activity.
  - It establishes the foundation of the system testing part of the overall software project.
  - It provides assurance of test coverage by creating a requirement traceability matrix.
  - It outlines an orderly schedule of events and test milestones that are tracked.
  - It specifies the personnel, financial, equipment, and facility resources required to support the system testing part of a software project.

STRUCTURE OF A SYSTEM TEST PLAN

- The activity of planning for system testing combines two tasks: *research and estimation.*
- Research allows us to define the scope of the test effort and resources already available in-house.
- Each major functional test suite consisting of test objectives can be described in a bounded fashion using the system requirements and functional specification as references.

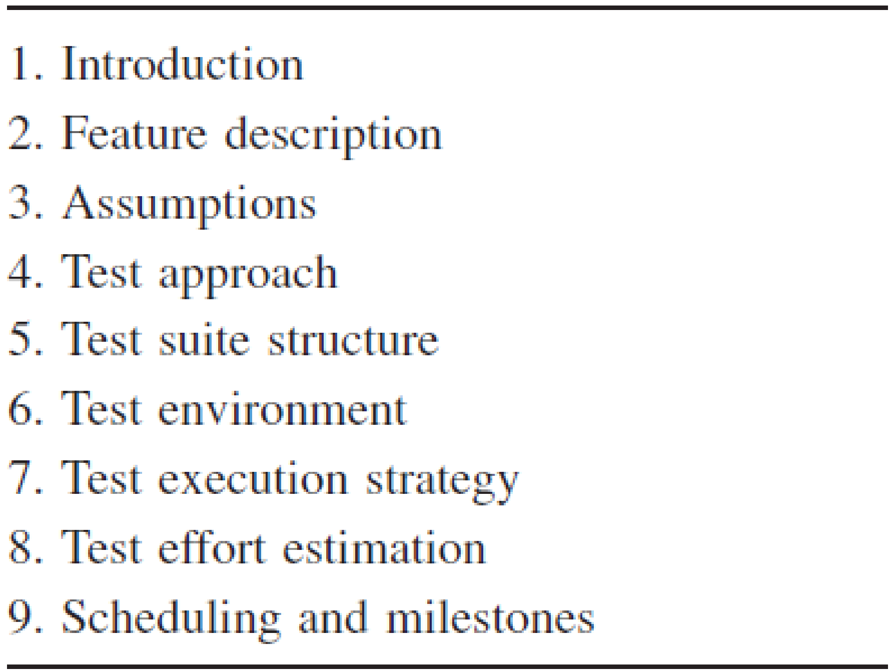


TABLE 12.1 Outline of System Test Plan

1. Introduction
2. Feature description
3. Assumptions
4. Test approach
5. Test suite structure
6. Test environment
7. Test execution strategy
8. Test effort estimation
9. Scheduling and milestones

INTRODUCTION AND FEATURE DESCRIPTION

- The introduction section of the system test plan describes the structure and the objective of the test plan.
- This section includes:

  (i) test project name,

  (ii) Revision history,

  (iii) terminology and definitions,

  (iv)names of the approvers and the date of approval,

  (v) references,

  (vi) summary of the rest of the test plan.
- The feature description section summarizes the system features that will be tested during the execution of this test plan.

ASSUMPTIONS

- The assumptions section describes the areas for which test cases will not be designed in this plan due to several reasons.
- First, the necessary equipment to carry out scalability testing may not be available.
- Second, it may not be possible to procure third-party equipment in time to conduct interoperability testing.
- Finally, it may not be possible to conduct compliance tests for regulatory bodies and environment tests in the laboratory.

TEST APPROACH

- The overall test approach is an important aspect of a testing project which consists of the following:
  - Lessons learned from past test projects are useful in focusing on problematic areas in the testing process. Issues discovered by customers that were not caught during system testing in past projects are discussed. For example, if a customer encountered a memory leak in the system in the past project, action must be taken to flush out any memory leak early in system test execution. An appropriate response may be needed to develop effective test cases by using sophisticated software tools such as memory leak detectors from the market.
  - If there are any outstanding issues that need to be tested differently, for example, requiring specific hardware and software configuration, then these issues need to be discussed here.

TEST APPROACH

- The overall test approach is an important aspect of a testing project which consists of the following:
  - A test automation strategy for writing scripts is a topic of discussion.
  - Test cases should be identified in the test factory that can be reused in this test plan.
  - Outline should be prepared of the tools, formats, and organizing scheme, such as a traceability matrix, that will be used and followed during the test project.

TEST SUITE STRUCTURE

- Detail test groups and subgroups are outlined in the test suite structure section based on the test categories identified in the test approach section.
- Test objectives are created for each test group and subgroup based on the system requirements and functional specification.
- A traceability matrix is generated to make an association between requirements and test objectives to provide the highest degree of confidence.
- At this stage only test suites along with the test objectives are identified but not the detail test cases.

TEST ENVIRONMENT

- It is necessary to plan for and design the test environment, also called a *test bed or* a *test laboratory*, to make the execution of system-level test cases effective.
- The objective here is to achieve effective testing whereby most of the defects in the system are revealed by utilizing a limited quantity of resources.
- Efforts must be made to create a deployment environment by using simulators, emulators, and third-party traffic generation tools.
- Such tools are found to be useful in conducting scalability, performance, load, and stress testing.

TEST ENVIRONMENT

- Multiple test environments are constructed in practice for the following reasons:
  - To run scalability tests, we need more resources than needed to run functional tests.
  - Multiple test beds are required to reduce the length of system testing time.
- Preparing for a test environment is a great challenge in test planning.
- This is especially true in testing distributed systems and computer networks where a variety of equipment are connected through communication protocols.

TEST ENVIRONMENT

- For example, such equipment includes user computers, servers, routers, base stations in a wireless network, authentication servers, and billing servers.
- It may take several months to set up an effective test bed for large, complex systems.
- A separate, dedicated system test laboratory, different from the ones used in unit and integration testing, is essential for the following reasons:
  - Test engineers need to have the ability to reconfigure a test environment.
  - Test activities need not interfere with development activities or live operations.
  - Increased productivity is achieved by having a dedicated test laboratory.

TEST ENVIRONMENT

- The following preparation activities are conducted to support the development of a system test bed:
  - Obtain information about the customer deployment architecture, including hardware, software, and their manufacturers. For example, the real deployment network diagram along with the software configuration is useful in designing a scaled-down version of the system in the laboratory. The manufacturer names will be handy in procuring the exact equipment for interoperability and compatibility testing.
  - Obtain a list of third-party products to be integrated with the SUT. Identification of the external products is important because of the need for performing interoperability testing.
  - List third-party test tools to be used to monitor, simulate, and/or generate real traffic. This traffic will be used as input to the SUT.

TEST ENVIRONMENT

- Identify the third-party software tools to be used under licenses.
- Identify the hardware equipment necessary to support special features specified in the requirement/test objectives, such as high availability and backup/recovery exercises within the test environment.
- List the number of hardware copies to carry out system testing if the project involves new hardware.
- Analyze the functional, performance, stress, load, and scalability test objectives to identify elements of the test environment that will be required to support those tests.

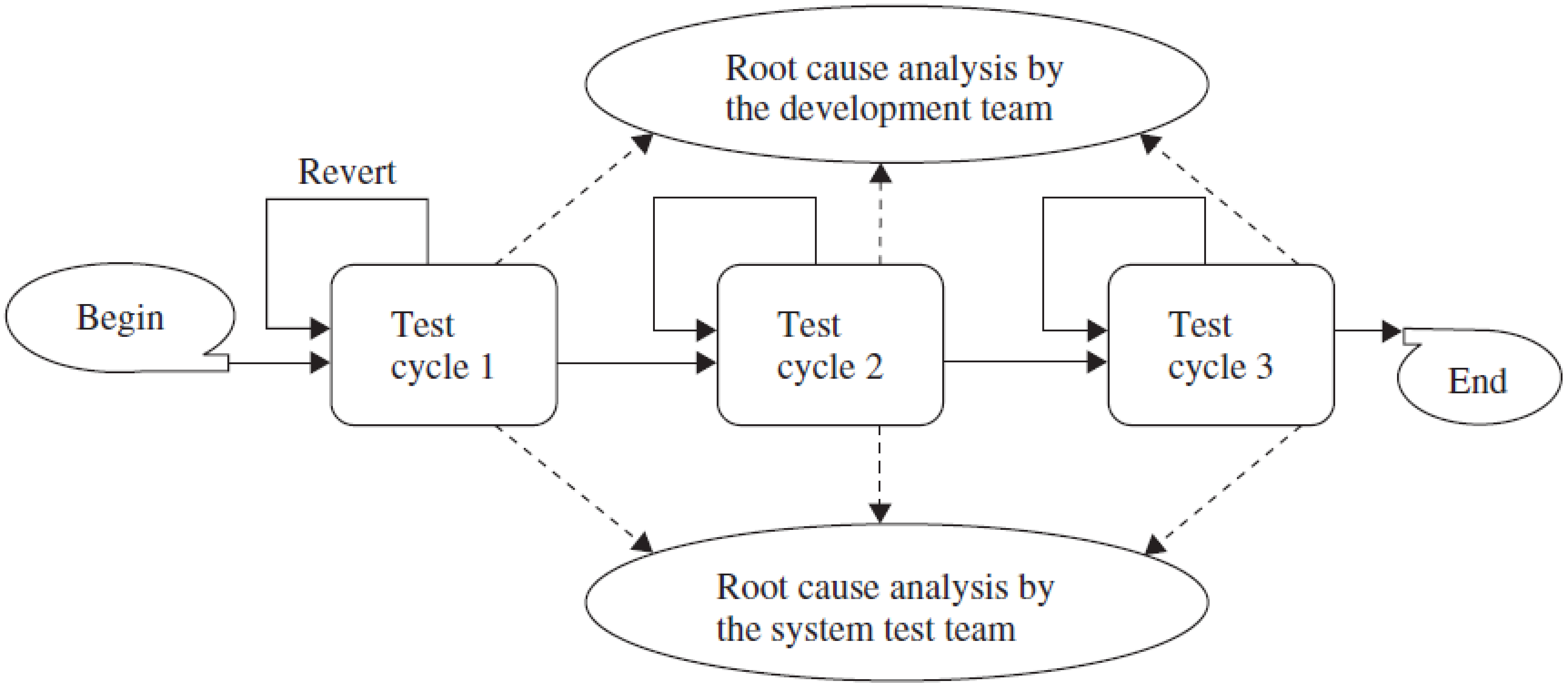
TEST ENVIRONMENT

- Identify the security requirements for the test environment. Ensure that the security test cases can be executed using the test environment and an intruder cannot disrupt the stress and stability tests that may be running overnight or over the weekends.
- List the small, but necessary networking gears that may be required to set up the test laboratory, such as switches, terminal servers, hubs, attenuators, splitters, personal computers, servers, and different kinds and sizes of cables to interconnect these gears.
- List any other accessories required to facilitate system testing, such as racks, vehicles, and special shielding to prevent radiation.

TEST EXECUTION STRATEGY

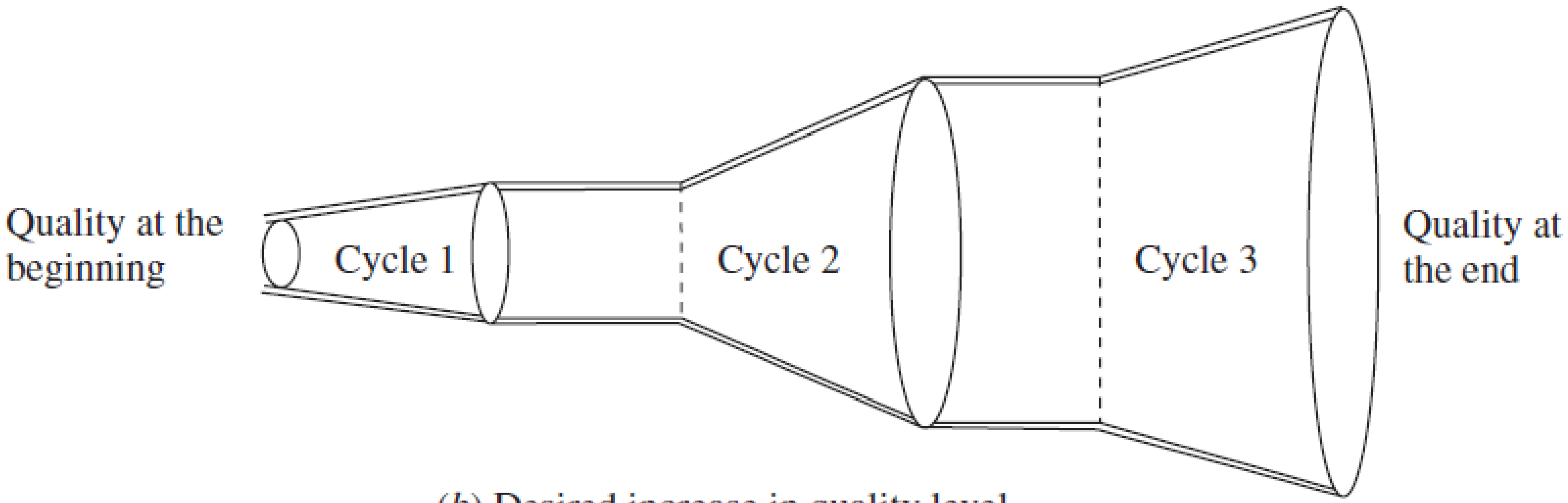
- The processes of test execution, defect detection, and fixing defects are intricately intertwined. The key characteristics of those processes are as follows:
- Some test cases cannot be executed unless certain defects are detected and fixed.
- A programmer may introduce new defects while fixing one defect, which may not be successful.
- The development team releases a new build for system testing by working on a subset of the reported defects, rather than all the defects.
- It is a waste of resources to run the entire test set T on a build if too many test cases fail.
- As the number of reported defects significantly reduces over a few iterations of testing, it is not necessary to run the entire test set T for regression testing—a carefully chosen subset of T *is expected to be adequate.*

Multicycle System Test Strategy


Root cause analysis by the development team
Revert
Begin
Test cycle 1
Test cycle 2
Test cycle 3
End
Root cause analysis by the system test team

(*a*) Progress of system testing in terms of test cycles

Multicycle System Test Strategy



(*b*) Desired increase in quality level

Characterization of Test Cycles

- Each test cycle is characterized by a set of six parameters: goals, assumptions, test execution, actions, revert and extension criteria, and exit criteria:
  1. Goals - A system test team sets its own goals to be achieved in each test cycle.
  2. Assumptions - In the assumption part, the system test group records its own assumption about when to select builds for system testing. Appropriate assumptions must be made in order to achieve the goals.
  3. Test Execution - Test cases are simultaneously executed in multiple test environments by using the concept of test prioritization. Prioritization of test execution changes between test cycles.

Characterization of Test Cycles

4. Revert and Extension Criteria - Each test cycle must be completed within a stipulated time. However, the duration of a test cycle may be extended under certain circumstances, and in the worst case, the test cycle must be restarted from the beginning.
5. Actions - Two unexpected events may occur while a test cycle is in progress:
   (i) too many test cases fail and
   (ii) the system test team has to design a large number of new test cases during the test cycle.
6. Exit Criteria We know when a test cycle has completed by applying an exit criterion. Mere execution of all the test cases in a test suite, or a subset thereof, does not mean that a cycle has completed.

Prioritization of Test Cases

- Prioritization of test cases means ordering the execution of test cases according to certain test objectives.

- ***Test Prioritization in Test Cycle 1***
- Principle. Prioritize the test cases to allow the maximum number of test cases to completely execute without being blocked.
- Test engineers execute their assigned test cases in different test environments.
- Each engineer prioritizes the execution of their subset of test cases as follows:
  - A high priority is assigned to the test cases in the basic and functionality test groups.
  - A medium priority is assigned to the robustness and interoperability test groups.
  - A low priority is assigned to the test cases in the following groups: documentation, performance, stress, scalability, and load and stability tests.

Prioritization of Test Cases

- ***Test Prioritization in Test Cycle 2***
- **Principle.** Test cases which failed in the previous test cycle are executed early in the test cycle.
- In the second test cycle, the test cases are reassigned to the test engineers based on their interest and expertise.
- Each test engineer prioritizes the execution of test cases in their subset as follows:
  - A high priority is assigned to the test cases in the red bin.
  - A medium priority is assigned to the test cases in the yellow bin.
  - A low priority is assigned to the test cases in the green bin.

Prioritization of Test Cases

- ***Test Prioritization in Test Cycle 3***
- **Principle.** Test prioritization is similar to that in the second test cycle, but it is applied to a selected subset of the test cases chosen for regression testing.
- Then each test engineer prioritizes the execution of test cases in their assigned subset as follows:
  - A high priority is assigned to the test cases in the red bin.
  - A low priority is assigned to the test cases in the yellow bin.

TEST EFFORT ESTIMATION

- The system test group needs to estimate testing effort to produce a schedule of test execution.
- Intuitively, testing effort defines the amount of work that needs to be done.
- In concrete terms, this work has two major components:
  - The number of test cases created by one person in one day
  - The number of test case executed by one person in one day
- In the planning stage it is natural to estimate the cost of the test and the time to complete the test.
- Together the two parameters *cost and time are called test effort* .

TEST EFFORT ESTIMATION

- The three key factors in estimating test effort are as follows:
  - Number of test cases to be designed for the software project
  - Effort required to create a detailed test case
  - Effort required to execute a test case and analyze the result of execution
- Some other factors affecting the test effort are as follows:
  - Effort needed to create test environments
  - Effort needed to train test engineers on the details of the project
  - Availability of test engineers for system testing as they are needed

Number of Test Cases

- Several ways to estimate the number of test cases:
- ***Estimation Based on Test Group Category***
  - It is straightforward to estimate the number of test cases after the test suite structure and the test objectives are created.
  - The number of test cases used by the end of the system testing phase is more than its initial estimation.
  - A more accurate estimation of the number of test cases for each group is obtained by adding a “fudge factor” of 10–15% to the total number of test objectives identified for the group.

Number of Test Cases

- ***Estimation Based on Test Group Category (cont.)***
  - For pragmatic reasons, for any moderate-size, moderate-risk testing project, a reasonable factor of safety, in the form of a fudge factor, needs to be included in the estimate in order to provide a contingency for uncertainties.
  - The test team leader can generate a table consisting of test (sub)groups and for each (sub)group the estimated number of test cases.
  - An additional column may be included for time required to create and time required to execute the test cases

Number of Test Cases

- ***Estimation Based on Function Points***
  - Given a functional view of a system in the form of the number of user inputs, the number of user outputs, the number of user on-line queries, the number of logical files, and the number of external interfaces, one can estimate the project size in number of lines of code required to implement the system and the number of test cases required to test the system.
  - Given a functional view of a system in the form of the number of user inputs, the number of user outputs, the number of user on-line queries, the number of logical files, and the number of external interfaces, one can estimate the project size in number of lines of code required to implement the system and the number of test cases required to test the system.

Number of Test Cases

- ***Estimation Based on Function Points***
- The function point metric can be utilized to estimate the number of test cases in the following ways:
  - **Indirect Method:** Estimate the code size from the function points and then estimate the number of test cases from the code size.
  - **Direct Method:** Estimate the number of test cases directly from function points.

Number of Test Cases

- ***Indirect Method :***
  - The function points of a software system are computed by examining the details of the requirements of the system from the requirement database.
  - At the time of such a computation, the programming language in which the system will be implemented may not be known.
  - Implementation of the system in different programming languages will produce different measures of line of code (LOC). Therefore, the choice of a programming language has a direct impact on the LOC metric.

Number of Test Cases

- Direct relationship between function points and the total number of test cases created as:

  Total number of test cases = $(\text{function points})^{1.2}$

- The number of test cases estimated above encompasses all forms of testing done on a software system, such as unit testing, integration testing, and system testing.
- It does not distinguish system test size from total testing effort.

SCHEDULING AND TEST MILESTONES

- Scheduling system testing is a portion of the overall scheduling of a software development project.
- It is merged with the overall software project plan after a schedule for system testing is produced.
- It is essential to understand and consider the following steps in order to schedule a test project effectively:

SCHEDULING AND TEST MILESTONES STEPS

1. Develop a detailed list of tasks, such as:
   - Procurement of equipment to set up the test environments
   - Setting up the test environments
   - Creation of detail test cases for each group and subgroup in the test factory
2. List all the major milestones needed to be achieved during the test project, such as:
   - Review of the test plan with cross-functional team members
   - Completion of the approved version of the test plan
   - Preparation of test environments
   - Date of entry criteria readiness review meeting
   - Official release of the software image to system test team

SCHEDULING AND TEST MILESTONES

3. Identify the interdependencies among the test tasks and any software milestones that may influence the flow of work, such as official release of the software image to the system test group. In addition, identify the tasks that can be performed concurrently. Some examples of concurrent implementations of test tasks are as follows:
   - Test beds can be set up concurrently with the creation of test cases.
   - Creation of test cases in different groups can be done concurrently.
   - Test cases from different groups can be concurrently executed on different test beds.

SCHEDULING AND TEST MILESTONES

4. Identify the different kinds of resources and expertise needed to fulfill each task on the list.
5. Estimate the quantity of resources needed for each task, such as equipment and human resources.
6. Identify the types and quantities of resources available for this testing project, such as:
   - Human resources: persons available and their dates of availability, part or full-time availability of each individual human resource, and area of expertise of each individual human resource.
   - Hardware/ software resources: availability of all the hardware/software to build the test environment

SCHEDULING AND TEST MILESTONES

7. Assume that the rest of the resources will be available by certain dates.
8. Allocate the resources to each task. This allocation is done task by task in sequence, starting backward from the end date of the test project.
9. Schedule the start and end dates of each task. For example, the test execution task can be based on the estimates. Allow delays for unforeseen events, such as illness, vacation, training, and meetings.
10. At this point insert the rest of the milestones into the overall schedule. You may have to adjust the resource allocation and the schedule dates.
11. Determine the earliest and the latest possible start date and end date for each task by taking into account any interdependency between tasks identified earlier.

SCHEDULING AND TEST MILESTONES

12. Review the schedule for reasonableness of the assumptions. At this stage, it is a good idea to ask "what if" questions. Analyze and change the schedule through iterations as the assumptions change.
13. Identify the conditions required to be satisfied for the schedule.
14. Document the assumptions, such as (i) new test engineer must be hired by a certain date, (ii) new equipment must be available by a certain date, and (iii) space availability to set up the test environment.
15. Review the schedule with the test team and get their feedback. The schedule may not be met without their active participation.

SYSTEM TEST AUTOMATION

- The organization must assess and address a number of considerations before test automation can proceed. The following prerequisites need to be considered for an assessment of whether or not the organization is ready for test automation:
  - The system is stable and its functionalities are well defined.
  - The test cases to be automated are unambiguous.
  - The test tools and infrastructure are in place.
  - The test automation professionals have prior successful experience with automation.
  - Adequate budget has been allocated for the procurement of software tools.
- The system must be stable enough for automation to be meaningful.
- If the system is constantly changing or frequently crashing, the maintenance cost of the automated test suite will be rather high to keep the test cases up to date with the system.

SYSTEM TEST AUTOMATION

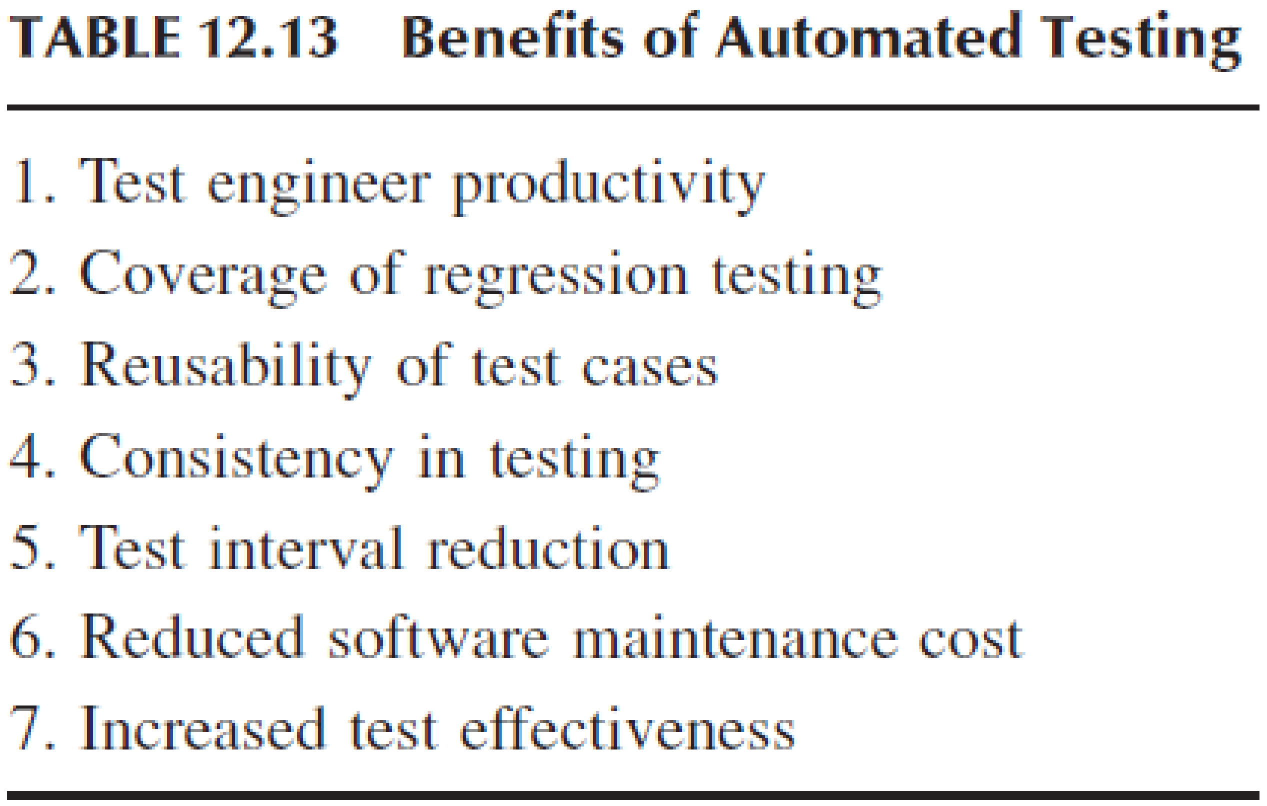


TABLE 12.13 Benefits of Automated Testing

1. Test engineer productivity
2. Coverage of regression testing
3. Reusability of test cases
4. Consistency in testing
5. Test interval reduction
6. Reduced software maintenance cost
7. Increased test effectiveness

EVALUATION AND SELECTION OF TEST AUTOMATION TOOLS

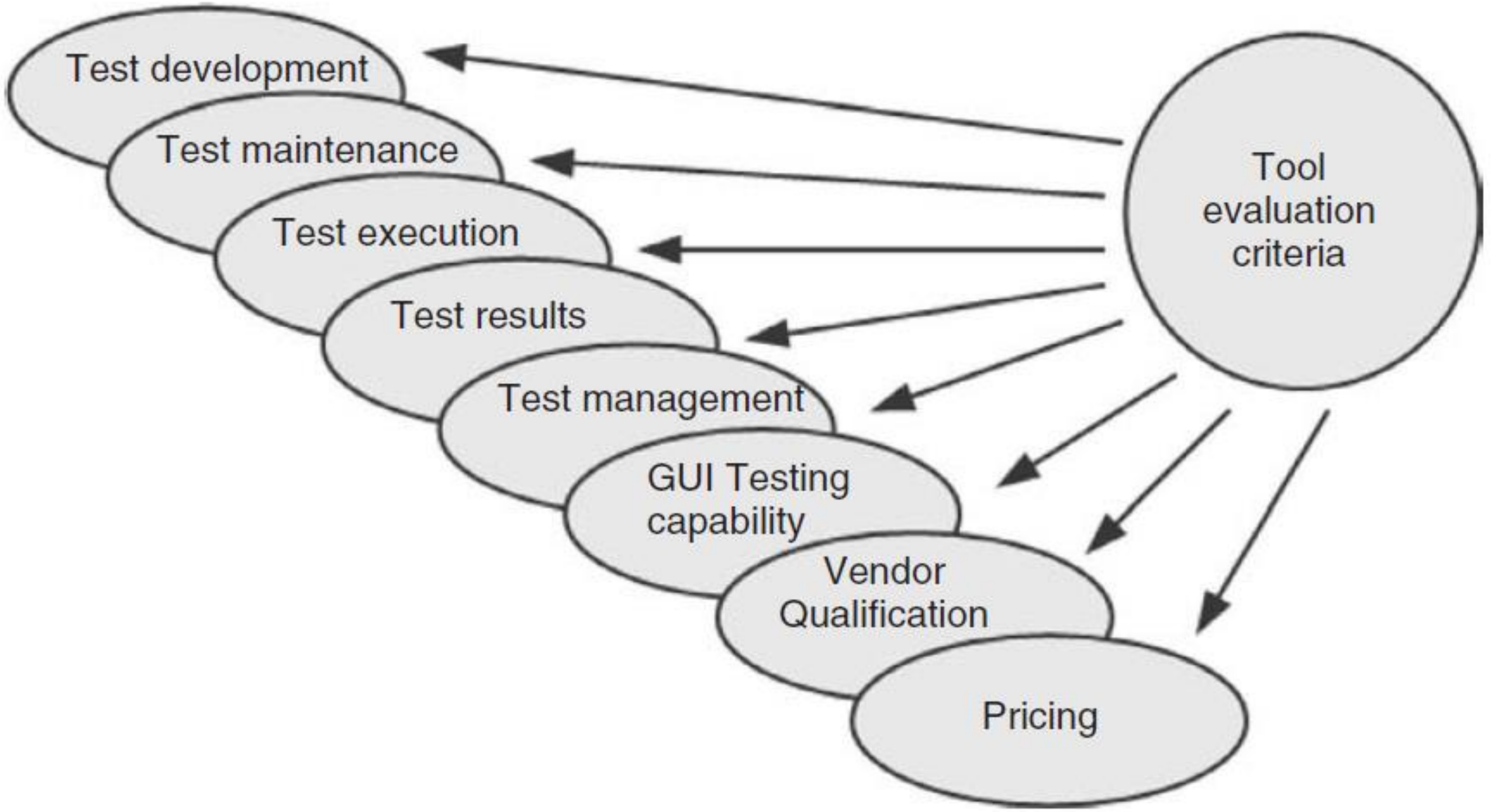


Figure 12.3 Broad criteria of test automation tool evaluation.

TEST SELECTION GUIDELINES FOR AUTOMATION

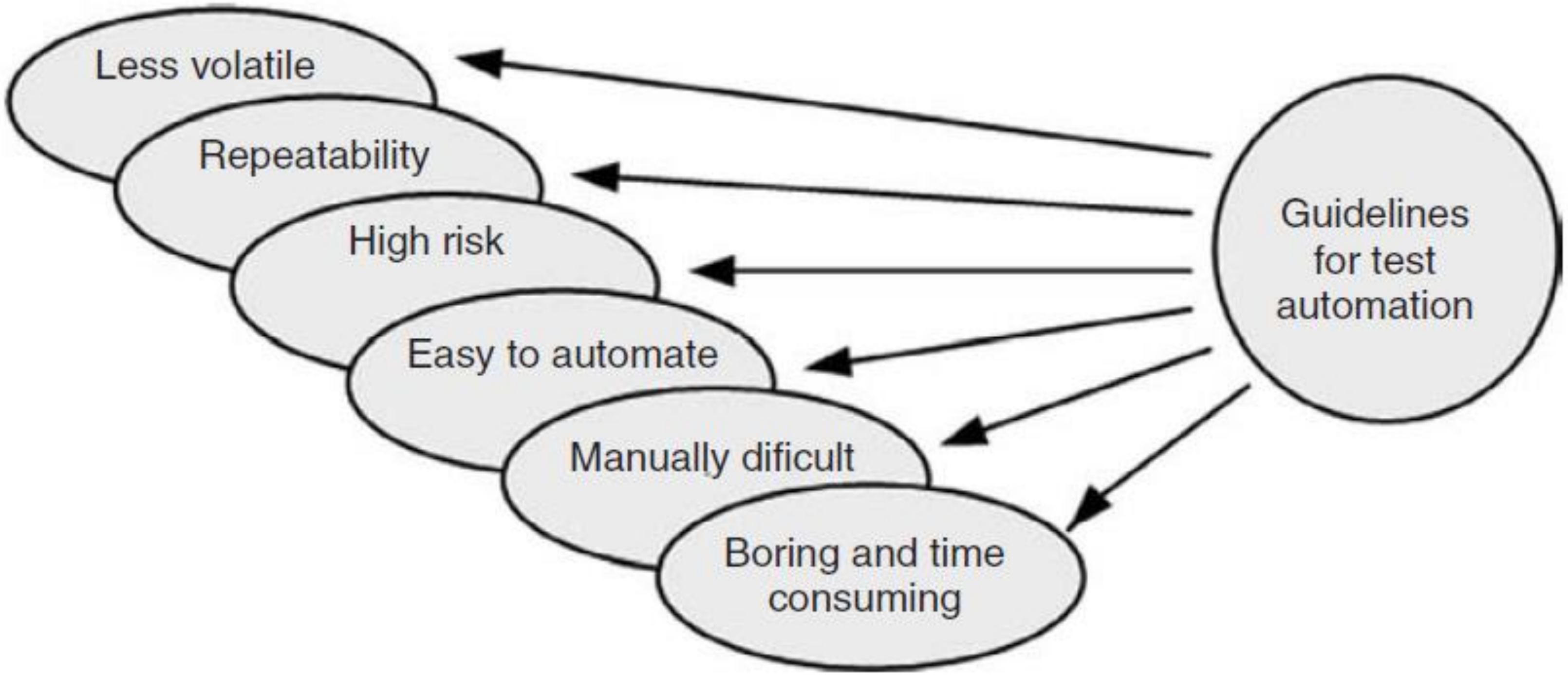


Figure 12.4 Test selection guideline for automation.

CHARACTERISTICS OF AUTOMATED TEST CASES

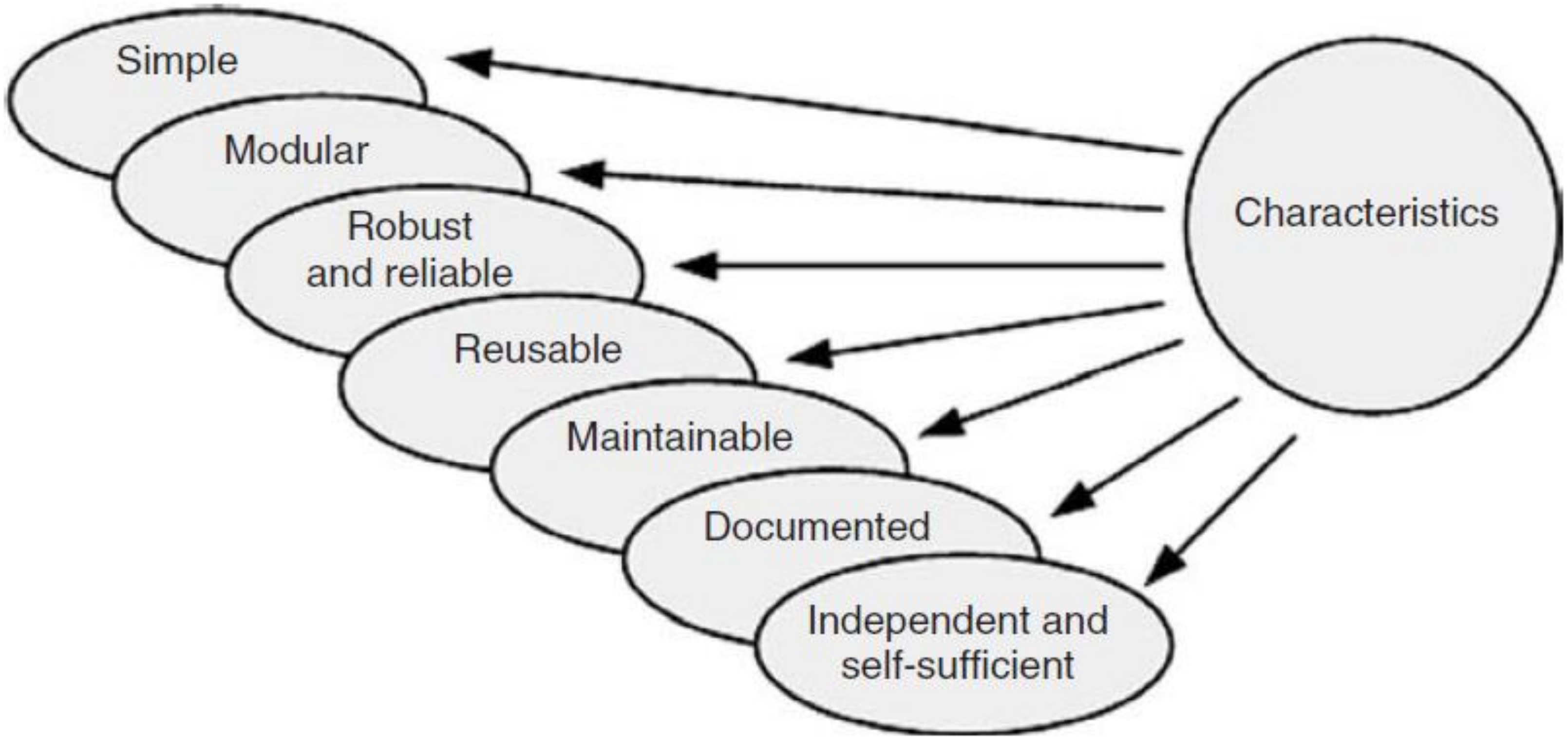


Figure 12.5 Characteristics of automated test cases.

STRUCTURE OF AN AUTOMATED TEST CASE

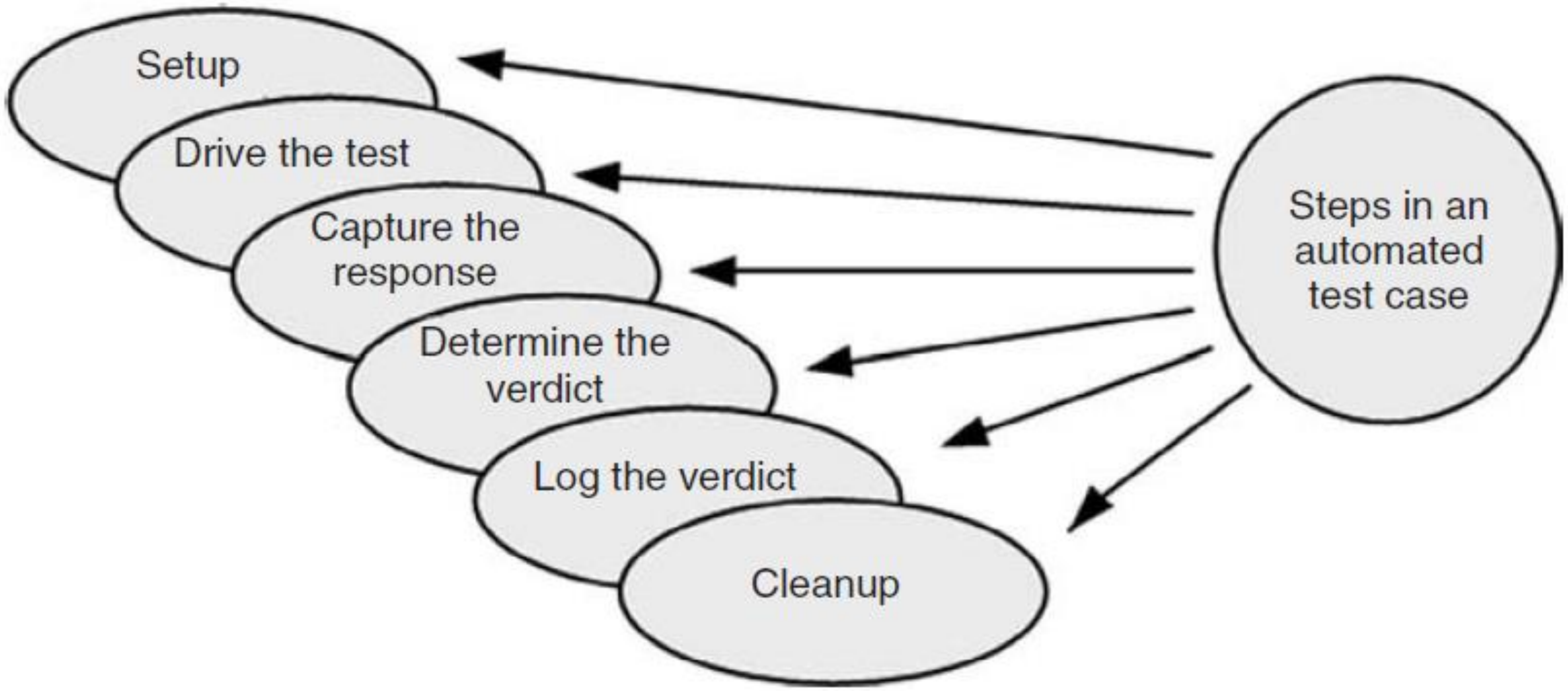


Figure 12.6 Six major steps in automated test case.

TEST AUTOMATION INFRASTRUCTURE

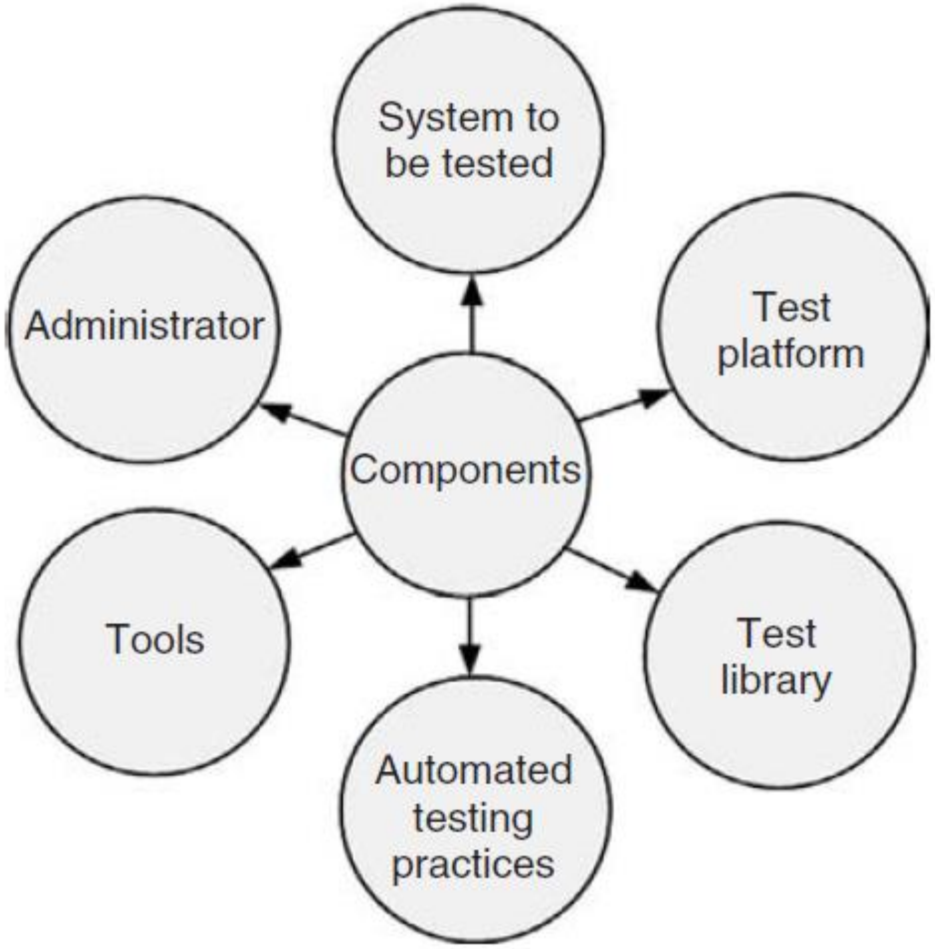


Figure 12.7 Components of automation infrastructure.