NEC
Empowered by Innovation

University of Waterloo

Software Testing and Quality Assurance

Theory and Practice

Chapter 12

System Test Planning and Automation

Outline of the Chapter

- Structure of a System Test Plan
- Introduction and Feature Description
- Assumptions
- Test Approach
- Test Suite Structure
- Test Environment
- Test Execution Strategy
  - A Multi-Cycle System Test Strategy
  - Characterization of Test Cycles
  - Preparing for the First Test Cycle
  - Selecting Test Cases for the Final Test Cycle
  - Prioritization of Test Cases
  - Details of Three Test Cycles
- Test Effort Estimation
  - Function Points
  - Computation of Function Point
  - Test Case Creation Effort
  - Test Case Execution Effort
- Scheduling and Milestones
  - Gantt Chart
- System Test Automation
- Evaluation and Selection of Test Tools
- Test Selection Guidelines for Automation
- Characteristics of Automation Test Cases
- Structure of an Automation Test Case
- Test Automation Infrastructure

Structure of a System Test Plan

The purpose of a system test plan is as follows:

- It provides guidance for the executive management to support the test project
- 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

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

12.1: An outline of a system test plan.

Introduction and Feature Description

- The *introduction* section of the system test plan includes:
  - Test project name
  - Revision history
  - Terminology and definitions
  - Name of the approvers and the date of approval
  - References
  - Summary of the rest of the test plan

- The *feature description* section summarizes the high level description of the functionalities of the system

Assumptions

- The *assumptions* section describes the areas for which test cases will not be designed
  - The necessary equipments to carry out scalability testing may not be available
  - It may not be possible to procure third-party equipments in time to conduct interoperability testing
  - It may not be possible to conduct compliance test for regulatory bodies and environment tests in the laboratory

Test Approach

- The *test approach* section describes the following aspect of the testing project
  - Issues discovered by customers that were not caught during system testing in the past project are discussed and the preventive action that are being taken in this test project
  - If there are any outstanding issues that need to be tested differently need to be discussed here
  - A test automation strategy for writing scripts is a topic of discussion
  - Identify test cases from the database that can be re-used in this test plan
  - Give an outline of the tools, formats, and organizing scheme, such as traceability matrix that will be used and folloed during the test project
  - Finally, the first level of test categories as discussed in Chapter 8 are identified

Test Suite Structure

- Detail *test groups* and *subgroups* are outlined based on the test categories identified in the *test approach* section
- Test objectives are created for each test group and subgroups based on the system requirements and functional specification documents
- Identification of test objectives provides a clue to the total number of test cases needs to be developed
- A traceability matrix is generated to make an association between requirements and test objectives to provide the test coverage

Test Environment

- Multiple test environments are constructed in practice
  - To run scalability tests one need more resources than to run functionality tests
  - To reduce the length of testing time

- A schematic diagram of one or more test beds are presented in this section of the system test plan
  - A high-level graphic layout of the test architectures
  - A table of types of equipments, their quantities, and their descriptions to support the test architecture

Test Execution Strategy

- The processes of system 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
- The following concerns must be addressed before the start of system testing:
  - How many times are the test cases executed and when?
  - What does one do with the failed test cases?
  - What happens when too many test cases fail?
  - In what order are the test cases executed?
  - Which test cases are to be run just before ending the system testing phase?

A Multi-Cycle System Test Strategy

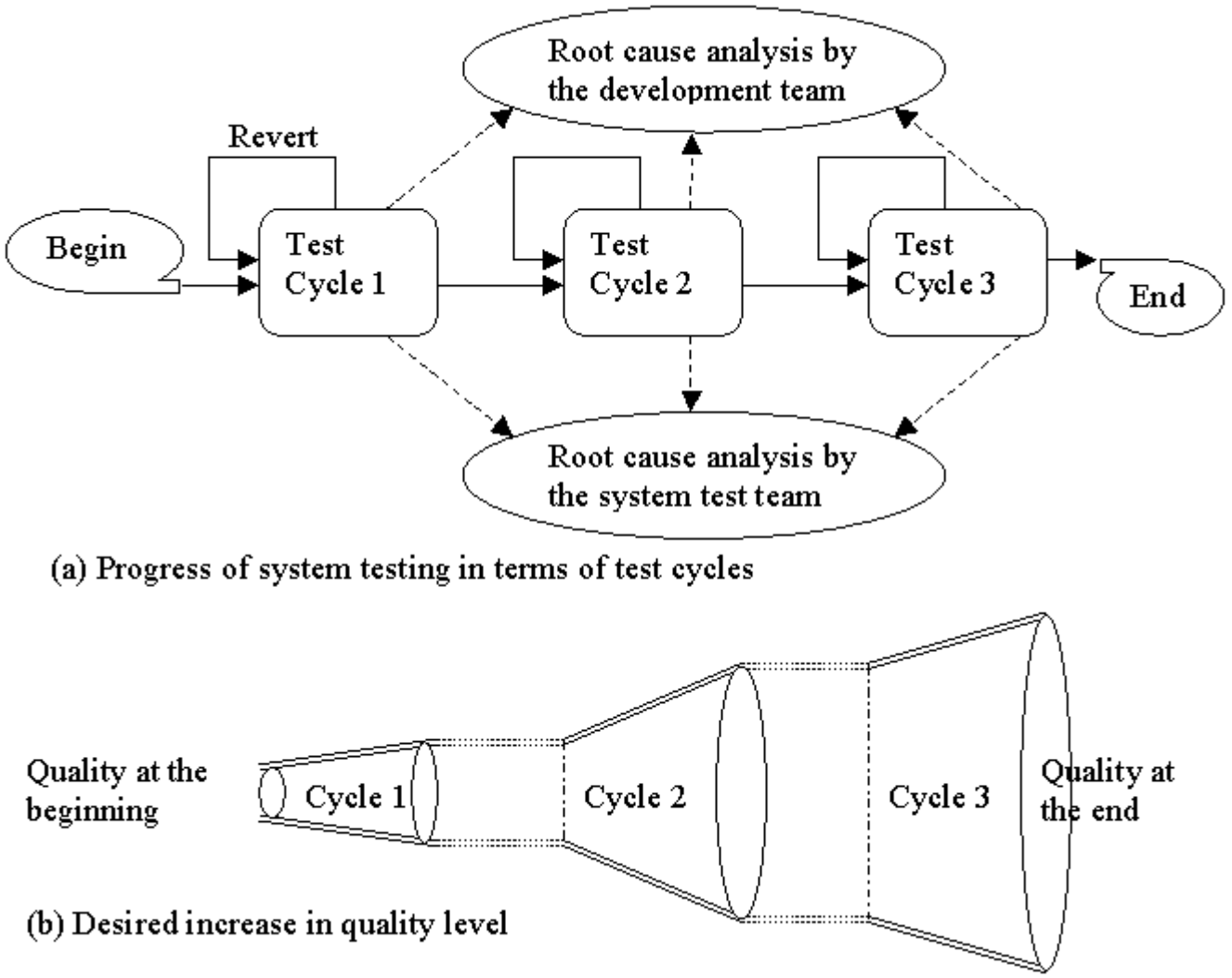

Figure 12.1: The concept of a cycle-based test execution strategy

Characterization of Test Cycles

Each test cycle is characterized by a set of six parameters:

- Goals
- Assumptions
- Test execution
- Revert and Extension criteria
- Actions
- Exit criteria

Characterization of Test Cycles

- **Goals**
  - System test team sets its own goals to be achieved in each test cycle
  - These goals are ideal in the sense these are very high standard
  - Goals are specified in terms of the number of test cases to pass in a cycle
- **Assumptions**
  - How often the builds will be selected during in a system test cycle?
  - For example "the team can accept builds from SIT group on a daily basis during a test cycle"
- **Test Execution**
  - Prioritization of test execution changes between test cycles
  - Test cases that exercise basic functionalities have higher priority than the rest in the first test cycle
  - Test cases that have failed in one test cycle have a higher priority in the following test cycle
  - Test cases in certain groups have higher priority than others

Characterization of Test Cycles

- **Revert and Extension criteria**

  - **It may not be useful to continue a test cycle if it is found that a software is of poor quality**

  - **Often a test cycle is extended due to various reasons**
    - A need to re-execute all the test cases in a particular test group because a large fraction of the test cases within the group failed
    - A significantly large number of new test cases were added while test execution was in progress

  - **The conditions for prematurely terminating a test cycle and for extending a test cycle must be precisely stated**

Characterization of Test Cycles

- **Actions**
  - **Too many test cases may fail during a test cycle**
    - **The development team is alerted**
    - **The developers take action in the form of root cause analysis (RCA)**
    - **Corrective actions are taken by updating the design specification, reviewing the code, and adding new test cases to the unit and integration test plan**
  - **The system test team has to design a large number of new test cases during a test cycle**
    - **The system test team initiates a root cause analysis (RCA)**
    - **The team studies the new test cases and categories them into different groups based on the functional requirements**
    - **The relevant requirements are studied to understand why the test team was unable to identify the objectives of these new test cases in the first place**

Characterization of Test Cycles

- **Exit Criteria**
  - **An exit criterion specifies the termination of a test cycle**
  - **One may exit a test cycle even if goals for that particular test cycle are not fully achieved**
  - **The exit criteria are defined based on few quality metrics associated with the test cycle**
  - **These quality metrics must be monitored during the test cycle**
  - **Example of exit criteria**
    - **95% of test cases passed and all the known defects are in CLOSED state**

Preparing for the First Test Cycle

- Life-cycle of a Defect
  - The life-cycle of a defect is represented by a state-transition diagram with five states: NEW, ASSIGNED, OPEN, RESOLVED, and CLOSED
  - The detail model of a defect life-cycle is discussed in Chapter 13.
- Assignment of test Cases
  - Since no single engineer is going to execute all the test cases all by themselves
  - it is desirable to assign test cases to appropriate test engineers by considering their expertise and interest
  - Assignment of test cases to test engineers may be changed from test cycle to test cycle
- Entry Criteria for the First Test Cycle
  - The entry criteria for the first test cycle, given in Table 12.3
  - It tell us when we should start executing system tests
  - It consists of five groups which depend on five cross-functional groups: **marketing, hardware, software, technical publication,** and **system testing**

Preparing for the First Test Cycle

Group	Criteria
1. Marketing	
1	Project plan and/or system requirements document is complete and updated.
2. Hardware	
2	All approved hardware versions for field deployment are available in-house. A list of these hardware versions should be provided.
3	Hardware version control process is in place and documented.
4	Hardware Design Verification Test (DVT) plan is completed and results are available.
3. Software	
5	All functional specifications (FS) are complete and have been updated to be in sync with the implementation. A list of individual FS including version number and status must be provided.
6	All design specifications (DS) are complete, and have been updated to be in sync with the implementation. A list of DS including version number and status must be provided.
7	All code complete and frozen – code changes allowed only for the defect fixing but not features.
8	A software version control is in place.
9	100% unit tests are executed and passed.
10	100% system integration tests are executed and passed.
11	Not more than two unique crashes have been observed during the last two weeks of integration testing.
4. Technical Publication	
12	A draft version of user guide is available.
5. System Testing	
13	The system test plan is in place (reviewed and approved).
14	Test execution working document is in place and complete.

Table 12.3: Entry criteria for the first system test cycle

Selecting Test Cases for the Final Test Cycle

Test cases are selected in three steps

- **Step 1:** The test suite is partitioned into red, yellow, green and white bins as follows:
  - *Red*: The *red* bin holds the following kinds of test cases that must be executed:
    - Test cases that failed at least once in the previous test cycles.
    - Test cases from those test groups for which root-cause analysis was conducted by the development team in the previous test cycles.
    - Test cases from the stress, scalability, and load and stability test groups
    - Test cases from the performance test category
  - *Yellow*: The *yellow* bin holds the test cases that are useful to execute. This bin includes those test cases whose objectives are similar to the objectives of the test cases in the red bin
  - *Green*: The *green* bin holds the test cases that will not add any value to regression testing, and, thus can be skipped
  - *White*: The test cases for which no concrete decision can be made in the first step are put in the *white* bin. This includes the rest of the test cases not falling in the *red*, *yellow*, or *green* bin

Selecting Test Cases for the Final Test Cycle

- **Step 2:** In this step, test cases from the white bin are moved to the other bins
  - The software developers identify all the software components that have been modified after the start of the first test cycle.
  - Each test case from the white bin is mapped to the identified software components.
  - This mapping is done by analyzing the objective of the test case and then checking whether the modified code of the identified software components is exercised by executing the test cases
  - The test cases that are mapped to more than one, one, or zero software components are moved to the *red*, *yellow*, or *green* bin, respectively.
- **Step 3:** In this step, test cases from the *red* and *yellow* bins are actually selected for regression testing as follows:
  - All the test cases from the red bin are selected for the final test cycle
  - Depending on the schedule, time-to-market, and customers demand, test cases from the yellow bin are selected for the final test cycle

Prioritization of Test Cases

- Prioritization of test cases means ordering the execution of test cases according to certain test objectives
- Formulating test objectives for prioritization of individual test cases is an extremely difficult task
- In system testing test cases are prioritize in terms of groups with common properties
- In a multi-cycle based test execution strategy, test cases are prioritized in a group in each test cycles for three reasons:
  - initially the quality level of the system under test is not very high
  - the quality of the system keeps improving from test cycle to test cycle
  - a variety of defects are detected as testing progresses

Prioritization of Test Cases

- 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.
    - 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 inter-operability 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
    - 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
- 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.
    - 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

Details of Three Test Cycles

- **Test Cycle 1**

Goals: To maximize the number of passed test cases (98%)

Assumptions: Test groups accepts a s/w image one every week for the first four weeks and one every weeks afterwards

Test Execution: Test cases are executed according the prioritization discussed earlier

Revert and Extension criteria:

(a) If the number of failed test cases reaches 20% of the total number of test cases to be executed the system test team abandons this test cycle

(b) If more than two unique crashes are observed during the test cycle, the system test team runs regression tests after the crash defects are fixed.

(c) If the number of failed test cases for any group of test cases reaches 20% of the number of test cases in the group during the test cycle, the test team re-execute all the test cases in that group in the test cycle after the defects are fixed

Details of Three Test Cycles

- **Test Cycle 1(Cont'd)**

Actions:

(a) The software development group initiates a root cause analysis (RCA) during the test cycle if the total number of failed test cases reaches some pre-set values as shown in Table 12.4

(b) The system test group initiates an RCA if the number of new test cases increases by 10% of the total number of test cases designed before the test cycle was started

	Test case failure count (Single week)	Test case failure count (Cumulative weeks)
Single test group	25%	20%
All test groups	20%	15%

Table 12.4: Test case failure counts to initiate root cause analysis in test cycle 1.

Exit Criteria: The test cycle is considered to have completed when the following predicates hold: (i) new test cases are designed and documented for those defects that were not detected by the existing test case - those are referred to as test case escapes, (ii) all test cases are executed at least once, (iii) 95% of test cases passed, and (iv) all the known defects are in CLOSED state

Details of Three Test Cycles

- **Test Cycle 2**

Goals: To maximize the number of passed test cases (99%)

Assumptions: Test groups accepts a s/w image one every two weeks

Test Execution: Test cases are executed according the prioritization discussed earlier

Revert and Extension criteria:

(a) If the number of failed test cases reaches 10% of the total number of test cases to be executed the system test team abandons this test cycle

(b) If more than one unique crashes are observed during the test cycle, the system test team runs regression tests after the crash defects are fixed.

(c) If the number of failed test cases for any group of test cases reaches 10% of the number of test cases in the group during the test cycle, the test team re-execute all the test cases in that group in the test cycle after the defects are fixed

Details of Three Test Cycles

- **Test Cycle 2(Cont'd)**

Actions:

(a) The software development group initiates a root cause analysis (RCA) during the test cycle if the total number of failed test cases reaches some pre-set values as shown in Table 12.5

(b) The system test group initiates an RCA if the number of new test cases increases by 5% of the total number of test cases designed before the test cycle was started

	Test case failure count (Single week)	Test case failure count (Cumulative weeks)
Single test group	15%	10%
All test groups	10%	5%

Table 12.5: Test case failure counts to initiate root cause analysis in test cycle 2.

Exit Criteria: The test cycle is considered to have completed when the following predicates hold: (i) new test cases are designed and documented for those defects that were not detected by the existing test case, (ii) all test cases are executed at least once, (iii) 98% of test cases passed, and (iv) all the known defects are in CLOSED state

Details of Three Test Cycles

- **Test Cycle 3**

Goals: A selected subset of test cases from the test suite shall be re-executed. The process of selecting test cases for this test cycle has been explained earlier. 100% of the selected test cases should pass.

Assumptions: Test groups accepts just one s/w image at the beginning of the test cycle

Test Execution: Test cases are executed according the prioritization discussed earlier

Revert and Extension criteria:

(a) If the number of failed test cases reaches 5% of the total number of test cases to be executed the system test team abandons this test cycle

(b) Testing is stopped if a single crash is observed

(c) The test cycle can be terminated and restarted again, but not extended

Details of Three Test Cycles

- **Test Cycle 3(Cont'd)**

Actions: N/A

Exit Criteria: The final test cycle is considered to have completed if all of these predicates hold:

- All the selected test cases are executed
- The results of all the tests are available
- 98% of test cases passed
- The 2% failed test cases should not be from stress, performance, scalability, and load and stability test groups
- The system does not crash in the final three weeks of testing
- The test report is completed and approved

Test Effort Estimation

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

Test Effort Estimation

- **Estimation of Number of test cases**
  - Estimation of Number of Test Cases 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
    - simply count the number of test objectives
  - Estimation of Number of Test Cases Based on Function Points
    - Total number of test cases = $(\text{Function Points})^{1.2}$

Function Points

- The central idea in the function point method is as follows:

  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

- The function point of a system is a weighted sum of the numbers of
  - inputs,
  - outputs,
  - master files
  - inquiries produced to, or generated by, the software

Computation of Function Point

- **Step 1:** Identify the following five types of components, also known as "user function types," in a software system

  - Number of external input types (NI)
  - Number of external output types (NO)
  - Number of external inquiry types (NQ)
  - Number of logical internal file types (NF)
  - Number of external interface file types (NE)

Computation of Function Point

- **Step 2:** Analyze the complexity of each of the above five types of user function and classify those to three levels of complexities, namely *simple*, *average*, or *complex*

Compute the unadjusted function point (UFP) by using the form shown in Table 12.7

$$UFP = WFNI \times NI + WFNO \times NO + WFNQ \times NQ + WFNL \times NL + WFNE \times NE$$

No.	Identifier	Complexity(Use one item in each row)			Total
		Simple	Average	Complex	
1	NI	__ × 3 = __	__ × 4 = __	__ × 6 = __	__
2	NO	__ × 4 = __	__ × 5 = __	__ × 7 = __	__
3	NQ	__ × 7 = __	__ × 10 = __	__ × 15 = __	__
4	NF	__ × 5 = __	__ × 7 = __	__ × 10 = __	__
5	NE	__ × 3 = __	__ × 4 = __	__ × 6 = __	__
				Total(UFP)	__

Table 12.7: A form for computing the unadjusted function point.

Computation of Function Point

- **Step 3:** Identify 14 factors that affect the required development effort for a project. A grade – between 0 and 5 – is assigned to each of the 14 factors for a certain project.

  The sum of these 14 grades is known as processing complexity adjustment (PCA) factor. The 14 factors have been listed in Table 12.8

In Table 12.8, the degree of influence of a factor on development effort is measured as follows:

- Not present, or no influence if present = 0
- Insignificant influence = 1
- Moderate influence = 2
- Average influence = 3
- Significant influence = 4
- Strong influence = 5

Computation of Function Point

No.	Factors affecting development effort	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5

Table 12.8: Factors affecting development time.

Computation of Function Point

- **Step 4:** Compute the function points (FP) of a system using the following empirical expression:

$$FP = UFP \times (0.65 + 0.01 \times PCA)$$

  - By multiplying the unadjusted function point, denoted by UFP, by the expression $(0.65 + 0.01 \times PCA)$, we get an opportunity to adjust the function point by +/- 35 percent.
  - Let us analyze the two extreme values of PCA, namely 0 and 70 (= 14 × 5).
    - $PCA = 0 : FP = 0.65 \times UFP$
    - $PCA = 70 : FP = UFP \times (0.65 + 0.01 \times 70) = 1.35 \times UFP$
  - The value of FP can range from $0.65 \times UFP$ to $1.35 \times UFP$
  - The value of FP is adjusted within a range of +/- 35 percent by using intermediate values of PCA.

Test Effort Estimation

- **Test Case Creation Effort**
  - **The productivity of the manual test case creation is summarized in Table 12.10**

Size of a Test Case	Average Number of Test Cases per Person-day
Small, simple test case (1 to 10 atomic test steps)	7 to 15
Large, Complex test case (10 to 50 atomic test steps)	1.5 to 3

Table 12.10: Guidelines for manual test case creation effort.

Test Case Execution Effort

Size of a Test Case	Average Number of Test Cases per Person-day
Small, simple test case (1 to 10 atomic test steps)	7 to 15
Large, complex test case (10 to 50 atomic test steps)	1.5 to 2.5

Table 12.11: Guidelines for manual test case execution effort.

Size of a Test Case	Average Number of Regression Test Cases per Person-day (did not execute the test cases earlier)	Average Number of Regression Test Cases per Person-day (test cases had been executed earlier)
Small, simple test case (1 to 10 atomic test steps)	10 to 15	10 to 20
Large, complex test case (10 to 50 atomic test steps)	1 to 2.5	2.5 to 5

Table 12.12: Guidelines for estimation of effort to manually execute regression test cases.

Scheduling and Test Milestones

- It is essential to understand and consider the following steps in order to schedule a test project effectively:
  - Develop a detailed list of tasks
  - List up all the major milestones needed to be achieved during the test project
  - Identify the interdependencies among the test tasks and any software milestones that may influence the flow of work
  - Identify the different kinds of resources and expertise needed to fulfill each task on the list
  - Estimate the quantity of resources needed for each task
  - Identify the types and quantities of resources available for this testing project
  - Assume that the rest of the resources will be available by certain dates
  - Allocate the resources to each task

Scheduling and Test Milestones

- Steps in order to schedule a test project effectively (Cont'd):
  - Schedule the start and end dates of each task
  - At this point insert rest of the milestones into the overall schedule
  - Determine the earliest and the latest possible start date and end date for each task by taking into account any inter-dependency between tasks identified earlier
  - Review the schedule for reasonableness of the assumptions
  - Identify the conditions required to be satisfied for the schedule
  - Document the assumptions
  - Review the schedule with the test team and get their feedback

Gantt Chart

- A Gantt chart is often used to represent a project schedule
- Gantt chart graphically displays the start and end points of each task, the total duration needed to complete each task
- As the project progresses, it displays the percentage of completion of each task
- It allows the planner to assess the duration of a project, identify the resources needed, and lay out the order in which tasks need to be carried out
- It is widely used for project planning and scheduling in order to:
  - Assess the time characteristics of a project
  - Show the task order
  - Show the link between scheduled tasks
  - Define resources involved
  - Monitor project completion

System Test Automation

- Benefits of system test automation
  - Test engineer productivity
  - Coverage of regression testing
  - Re-usability of test cases
  - Consistency in testing
  - Test interval reduction
  - Reduces software maintenance cost
  - Increased test effectiveness
- Pre-requisites 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 engineers have prior successful experience with automation
  - Adequate budget should have been allocated for the procurement of tools

Evaluation and Selection of Test Tools

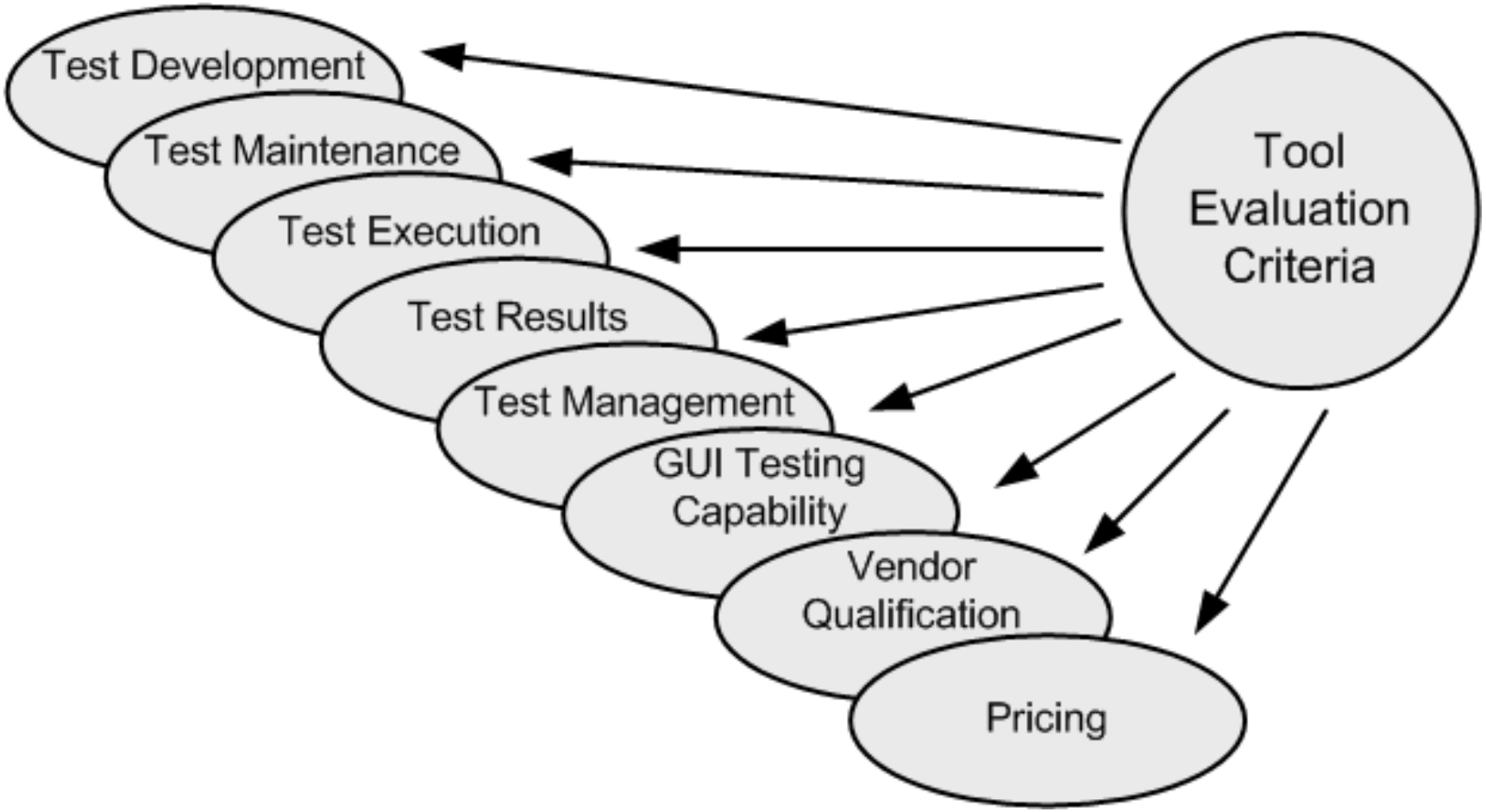

Figure 12.3: Broad criteria of test automation tool evaluation

Test Selection Guidelines for Automation

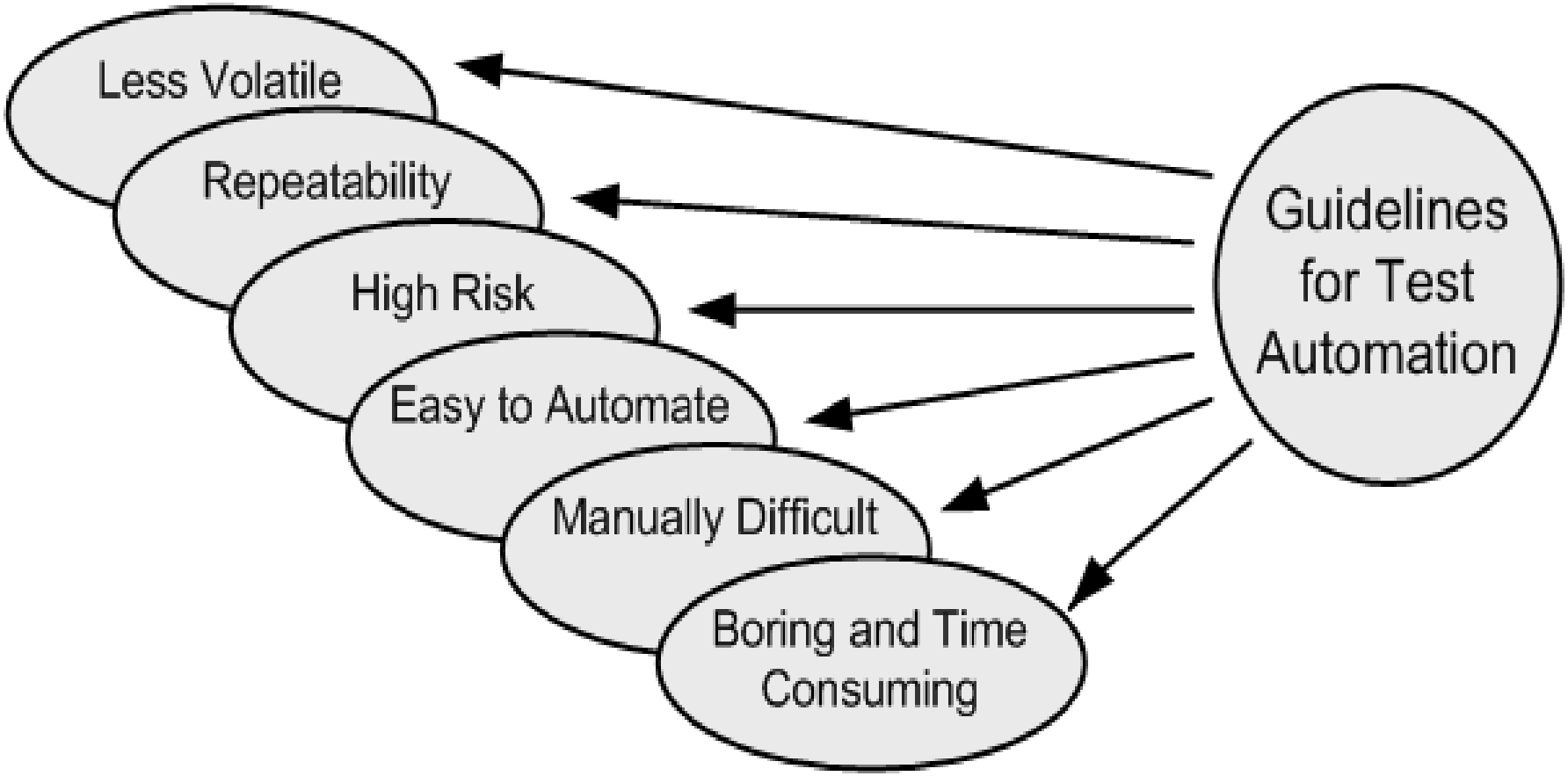

Figure 12.4: Test selection guideline for test automation

Characteristics of Automation Test Cases

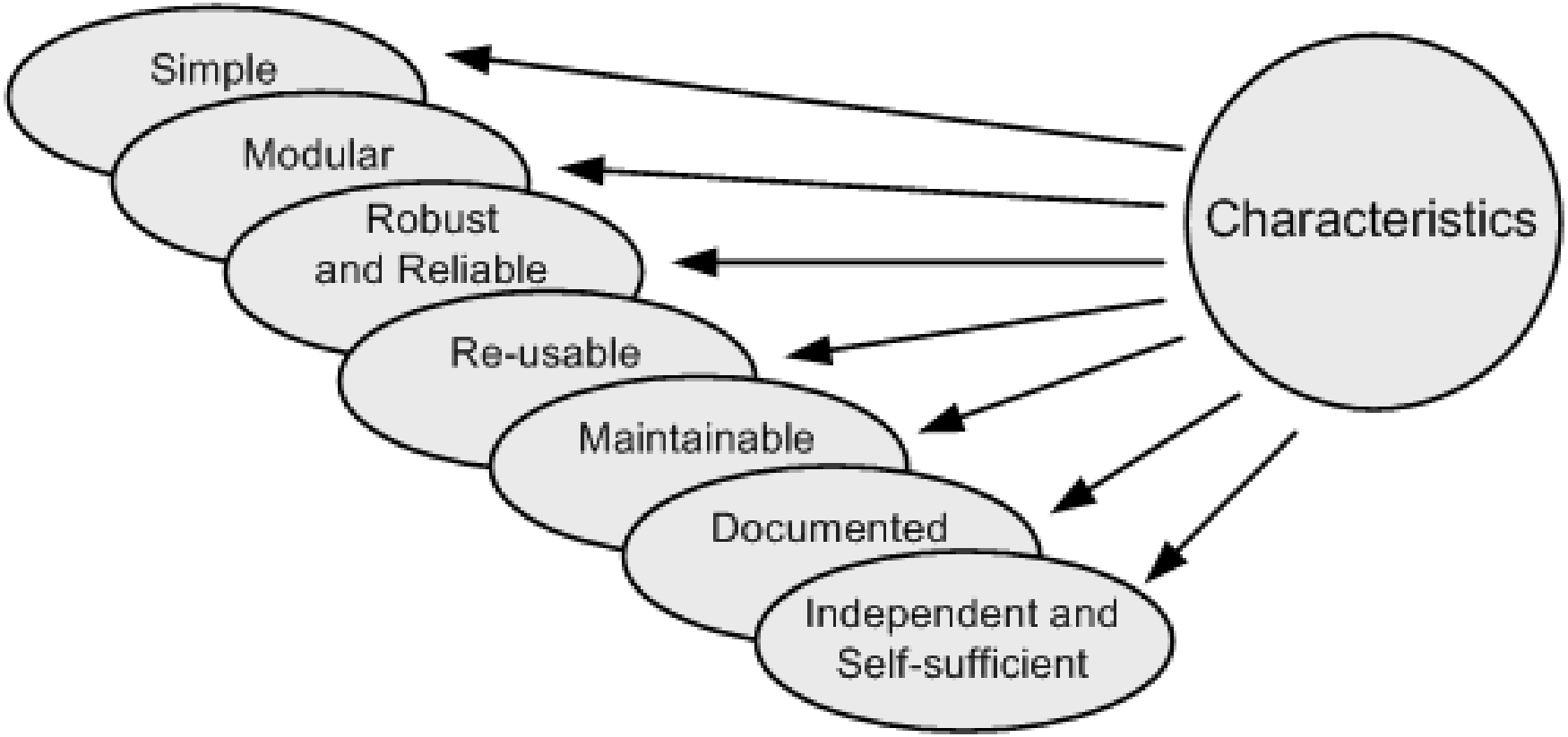

Figure 12.5: Characteristics of automated test cases

Structure of an Automation Test Case

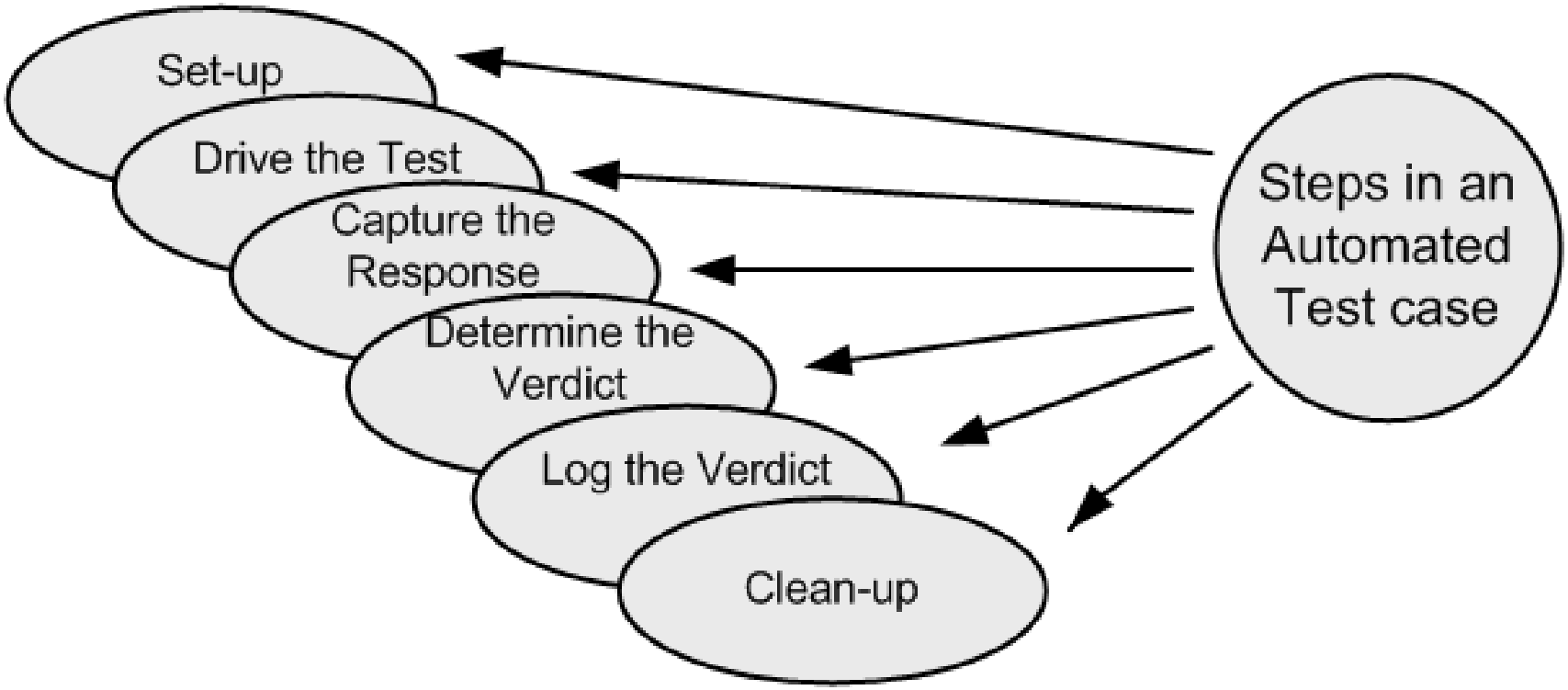

Figure 12.6: Six major steps in an automated test case

Test Automation Infrastructure

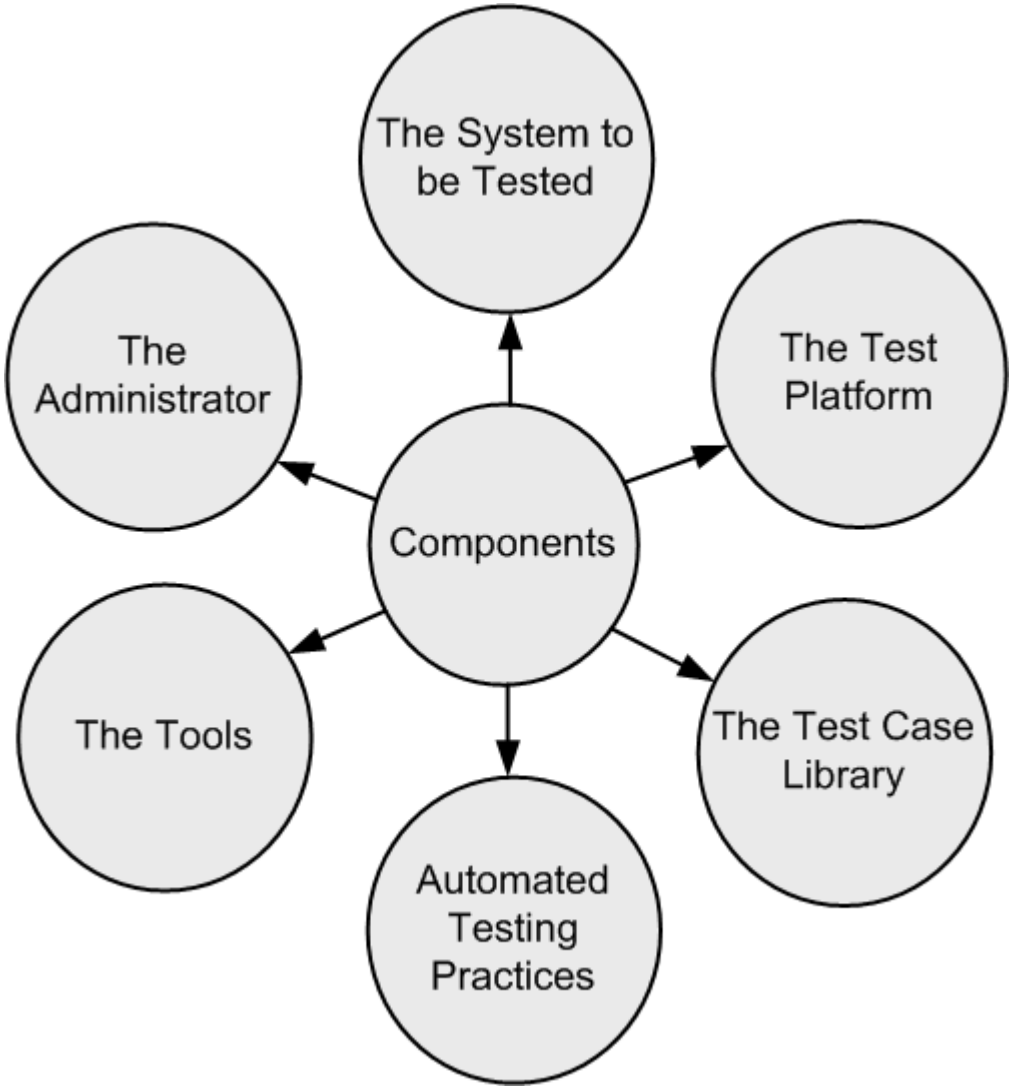

Figure 12.7: Components of an automation infrastructure