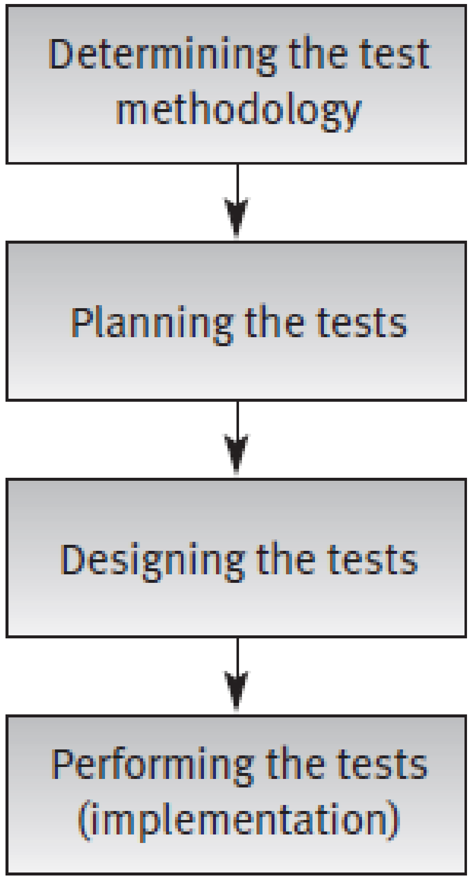
IDENTIFY BASIC TEST PROCESSES

CHAPTER 02

THE TESTING PROCESS

- Planning, design and performance of testing are carried out throughout the software development process.
- These activities are divided in phases, beginning in the design stage and ending when the software is installed at the customer's site.

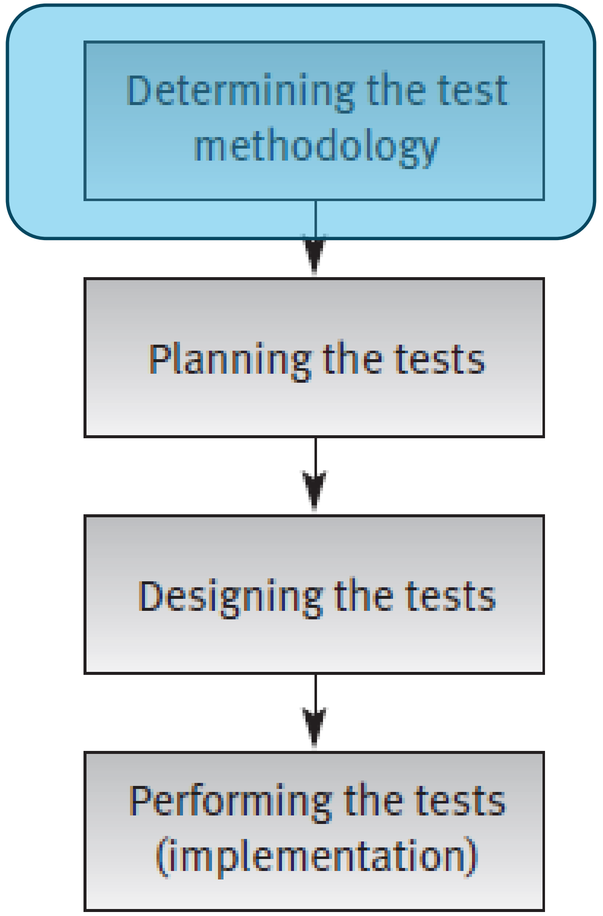

Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

1. DETERMINING THE TEST METHODOLOGY PHASE

The main issues that testing methodology has to contend with are:

1.1 The appropriate required software quality standard

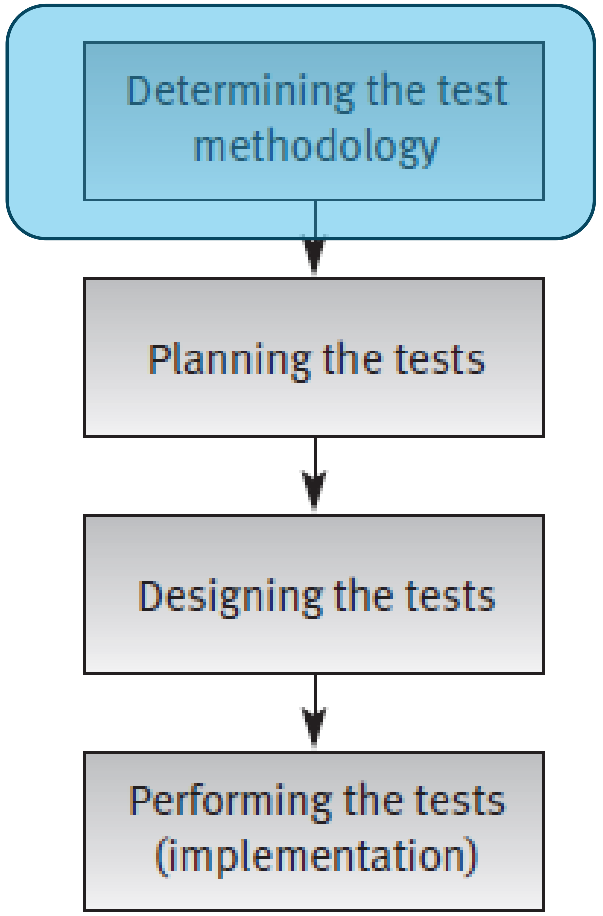
1.2 The software testing strategy. Decisions about these two issues are fundamental and must be made before planning begins.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

1.1 DETERMINING THE APPROPRIATE SOFTWARE QUALITY STANDARD

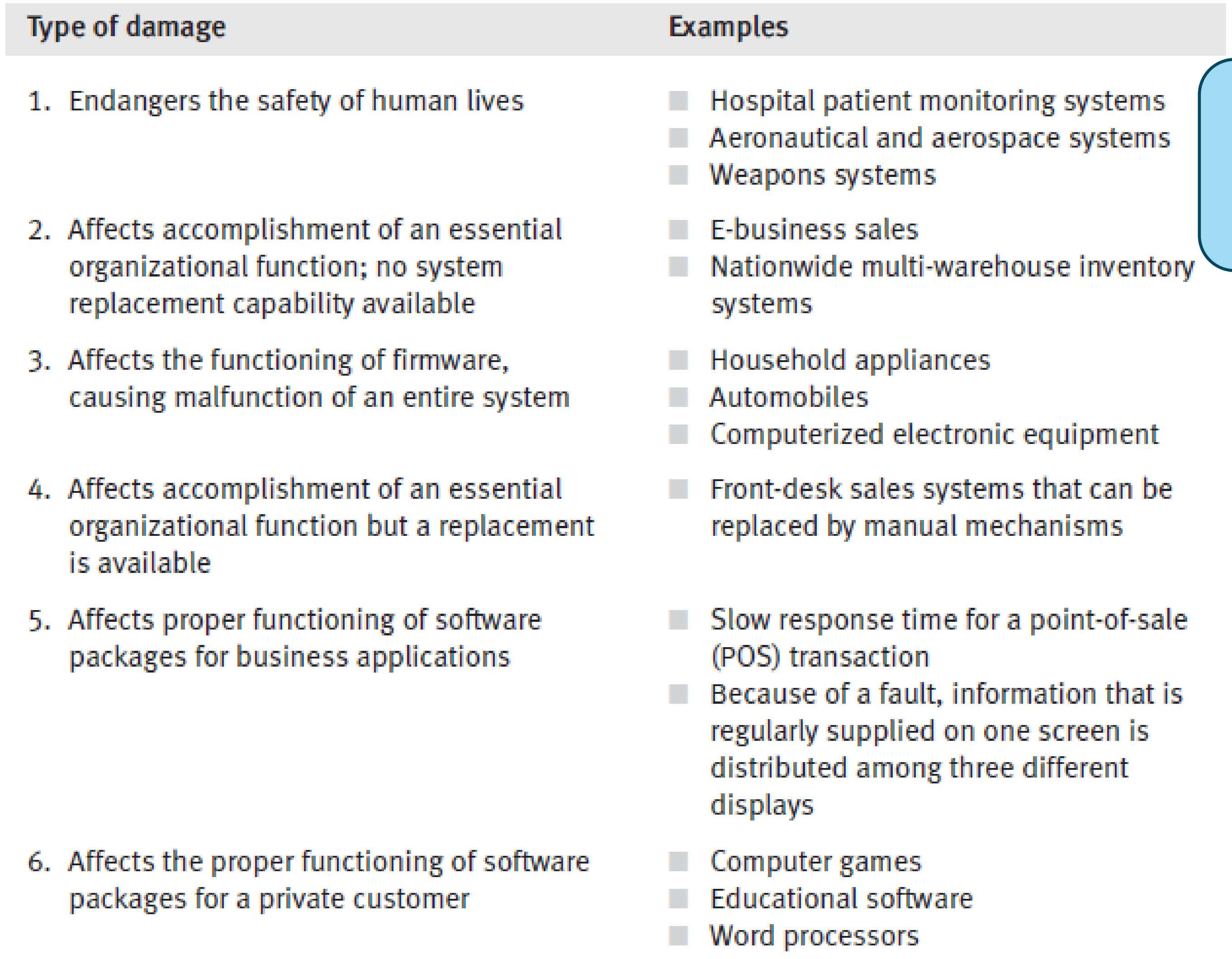
The level of quality standard selected for a project depends mainly on the characteristics of the software's application.

- Example 1: A software package for a hospital patient bed monitor requires the highest software quality standard considering the possibly severe consequences of software failure.
- Example 2: A package developed for handling feedback information for an organization's internal employee training program could make do with a medium-level software quality standard, assuming that the cost of failure is relatively low.

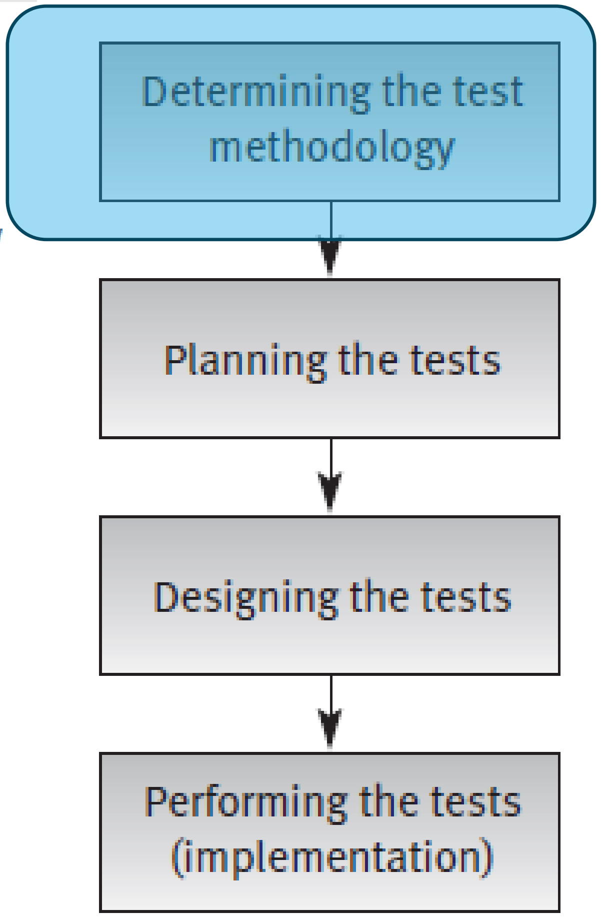

Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

DETERMINING THE APPROPRIATE SOFTWARE QUALITY STANDARD : DAMAGE TO THE END USERS

Above examples illustrate the main criterion to be applied when choosing the software quality standard: the evaluation of the nature and magnitude of
expected damages in case of system failure. These damages may affect the
customers and users on one hand, and the developer on the other

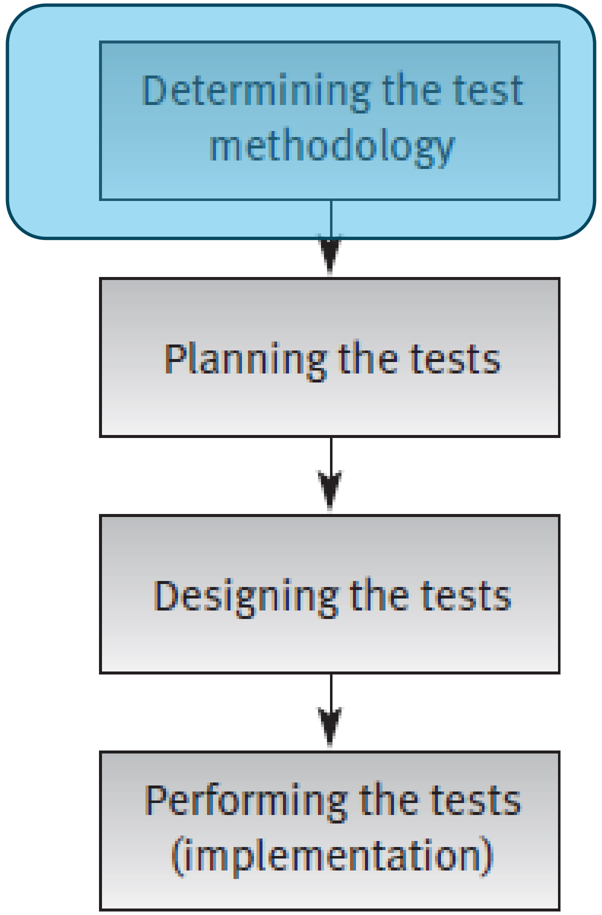


Type of damage	Examples
1. Endangers the safety of human lives	■ Hospital patient monitoring systems ■ Aeronautical and aerospace systems ■ Weapons systems
2. Affects accomplishment of an essential organizational function; no system replacement capability available	■ E-business sales ■ Nationwide multi-warehouse inventory systems
3. Affects the functioning of firmware, causing malfunction of an entire system	■ Household appliances ■ Automobiles ■ Computerized electronic equipment
4. Affects accomplishment of an essential organizational function but a replacement is available	■ Front-desk sales systems that can be replaced by manual mechanisms
5. Affects proper functioning of software packages for business applications	■ Slow response time for a point-of-sale (POS) transaction ■ Because of a fault, information that is regularly supplied on one screen is distributed among three different displays
6. Affects the proper functioning of software packages for a private customer	■ Computer games ■ Educational software ■ Word processors


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

DETERMINING THE APPROPRIATE SOFTWARE QUALITY STANDARD : DAMAGE TO THE DEVELOPER

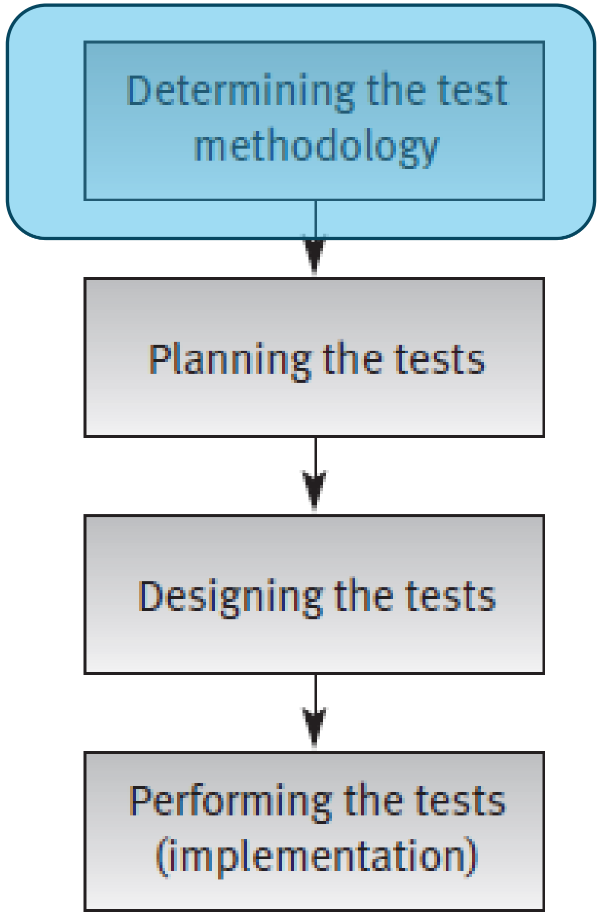
Type of damage	Examples
1. Financial losses	■ Damages paid for physical injuries ■ Damages paid to organizations for malfunctioning of software ■ Purchase cost reimbursed to customers ■ High maintenance expenses for repair of failed systems
2. Non-quantitative damages	■ Expected to affect future sales ■ Substantially reduced current sales


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

1.2 DETERMINING THE SOFTWARE TESTING STRATEGY

The issues that have to be decided include:

- The testing strategy: should a big bang or incremental testing strategy be adopted? If incremental testing is preferable, should testing be performed bottom-up or top-down?
- Which parts of the testing plan should be performed according to the white box testing model?
- Which parts of the testing plan should be performed according to the automated testing model?


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2. PLANNING THE TESTS

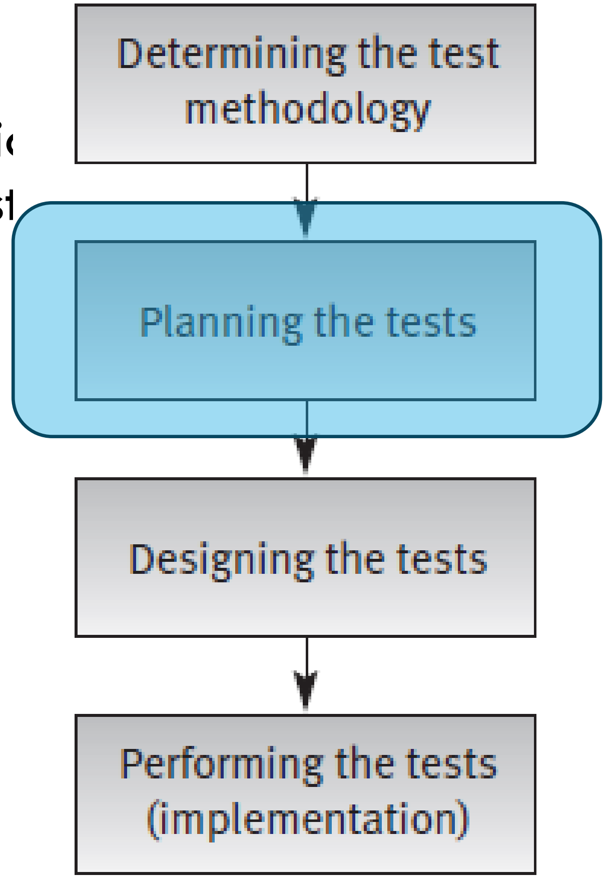
The tests to be planned include:

- ❑Unit tests
- ❑Integration tests
- ❑System tests.

While unit tests deal with small units of software or modules, integrati tests deal with several units that combine into a subsystem. System test refer to the entire software package/system.

It is incumbent upon planners to consider the following issues before initiating a specific test plan:

- 2.1 What to test?
- 2.2 Which sources to use for test cases?
- 2.3 Who is to perform the tests?
- 2.4 Where to perform the tests?
- 2.5 When to terminate the tests?


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

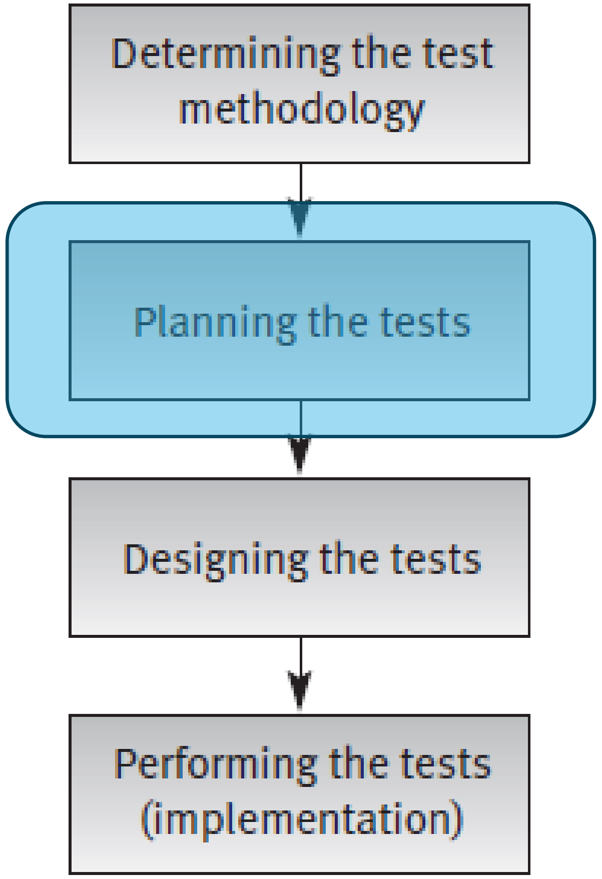
2.1 WHAT TO TEST?

Only in rare cases it is justified to test "everything". Usually, the feasibility of testing "everything" is highly limited.

- Software test plan that requires performing unit tests for all the individual units, integration tests for all the unit integrations, and a system test to test the software package as a whole. I
- Implementing this "straightforward" plan ensures top quality software but requires the investment of vast resources and an extended timetable.

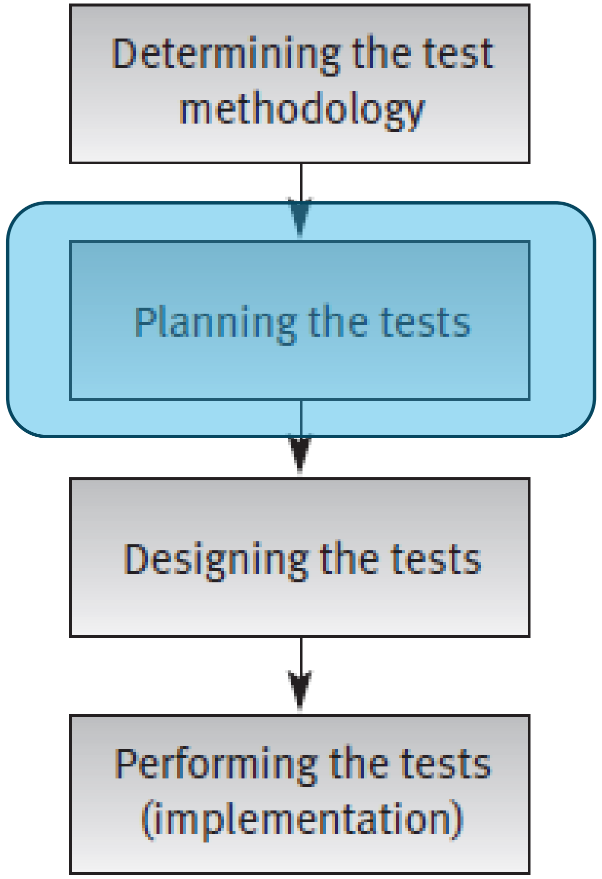
1. Is it justified to perform unit tests for a module composed of 98% reused software?

2. Is a unit test mandatory for a simple module that represents the 12th version of a basic module repeatedly applied by the development team over the last three years?


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.1 WHAT TO TEST? CONT.

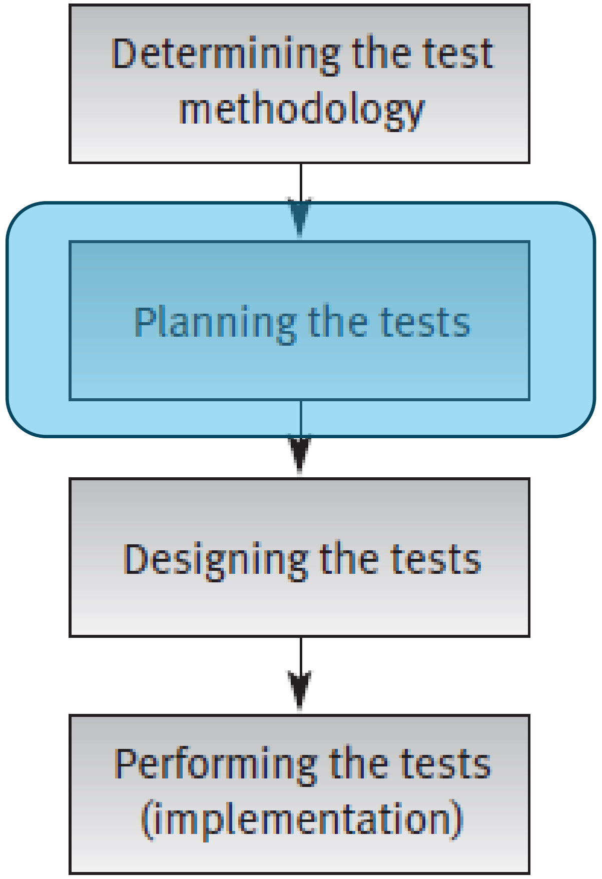
- Which modules should be unit tested ?
- Which integrations should be tested ?
- The priorities determining allocation of testing resources to the individual software system applications. As a result, low-priority applications are tested by only some types of tests or not included in the system test at all.
- In determining what is to be included and what excluded from the system tests, the unit and integration tests already planned should be considered.
- For the software quality of applications and modules not covered by the unit, integration and system tests, we rely on the code checks done by the programmer and his team leader and on code inspections and walkthroughs initiated by the development team.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

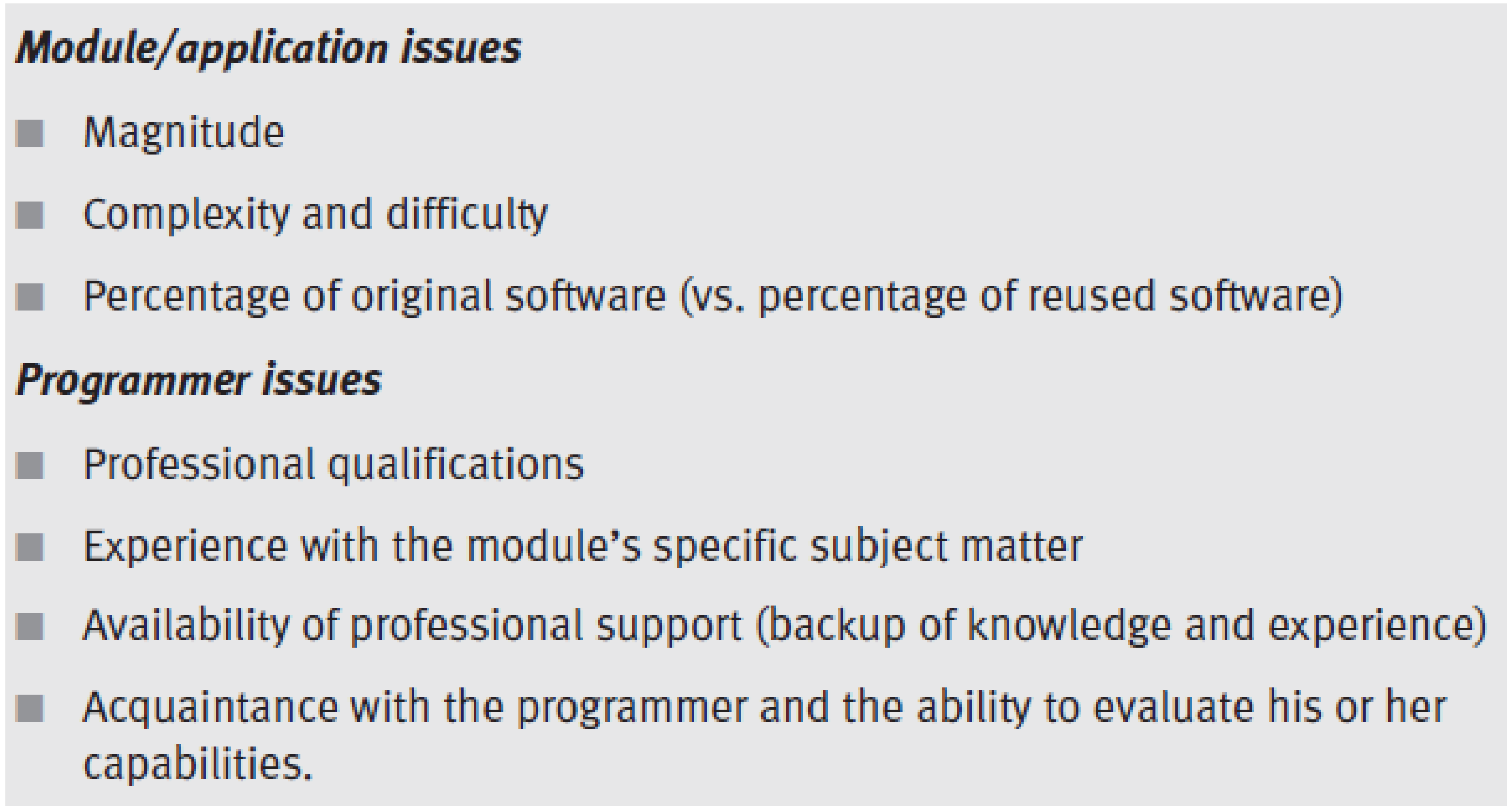
2.1 WHAT TO TEST? CONT.

Rating units, integrations and applications

- The methods for rating units (modules), integrations and applications to determine their priority in the testing plan are based on two factors:
- **Factor A:** Damage severity level. The severity of results in case the module or application fails.
- **Factor B:** Software risk level. The level of risk represents the probability of failure. In order to determine the risk level of a module, unit, integration or application, the issues affecting risk require examination. These issues can be classified as module/application issues and programmer issues.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

- **Factor B:** Software risk level. The level of risk represents the probability of failure. In order to determine the risk level of a module, unit, integration or application, the issues affecting risk require examination. These issues can be classified as module/application issues and programmer issues.

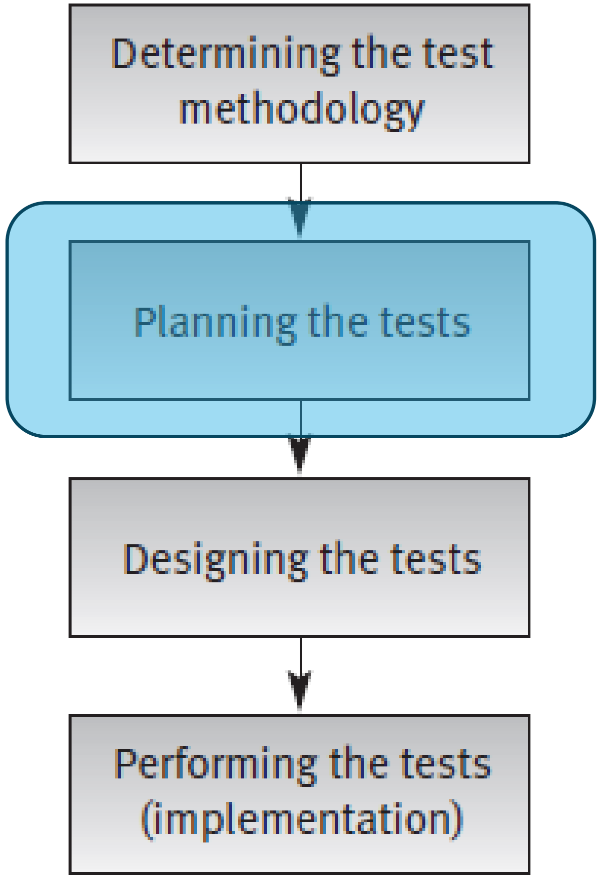


Module/application issues

- Magnitude
- Complexity and difficulty
- Percentage of original software (vs. percentage of reused software)

Programmer issues

- Professional qualifications
- Experience with the module's specific subject matter
- Availability of professional support (backup of knowledge and experience)
- Acquaintance with the programmer and the ability to evaluate his or her capabilities.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

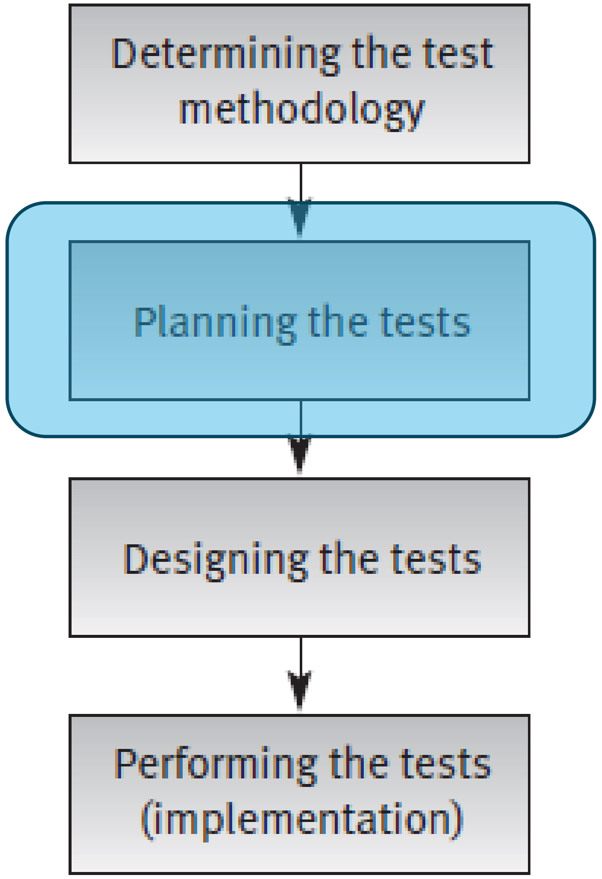
2.2 WHICH SOURCES TO USE FOR TEST CASES?

The planners should consider which of the two main sources of test cases –

1. Real-life test cases
2. Synthetic test cases – are most appropriate to their needs.

Each component of the testing plan, dealing with unit integration or the system test, requires an individual decision about the respective test cases and their source:

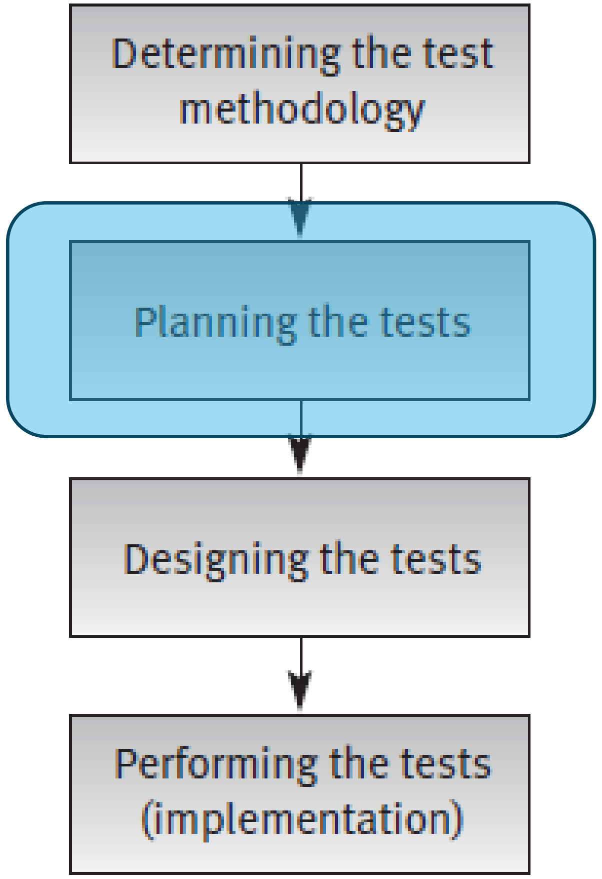
- ❑ The use of a single or combined source of test cases or both
- ❑ How many test cases from each source are to be prepared
- ❑ The characteristics of the test cases.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.3 WHO PERFORMS THE TESTS?

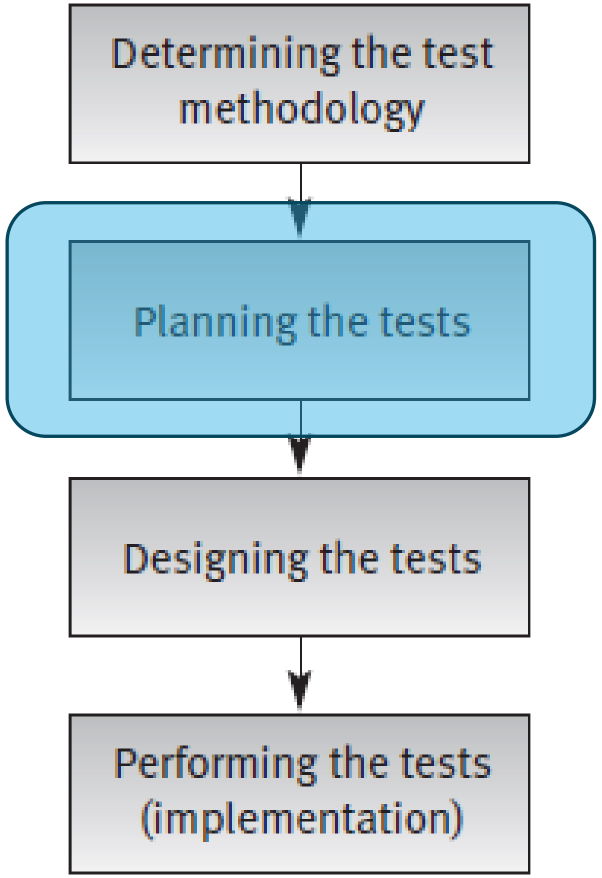
Who will perform the various tests is determined at the planning stage:

- Integration tests, but especially unit tests, are generally performed by the software development team. In some instances it is the testing unit that performs the tests.
- System tests are usually performed by an independent testing team (internal testing team or external testing consultants team).
- In cases of large software systems, more than one testing team can be employed to carry out the system tests. The prerequisite decision to be made in such cases concerns the allocation of system tests between the internal and the external testing teams. In small software development organizations, where a separate testing team does not exist, the following testing possibilities nonetheless exist:
  - Testing by another development team. Each development team will serve as the testing team for projects developed by other teams.
  - Outsourcing of testing responsibilities.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.4 WHERE TO PERFORM THE TESTS?

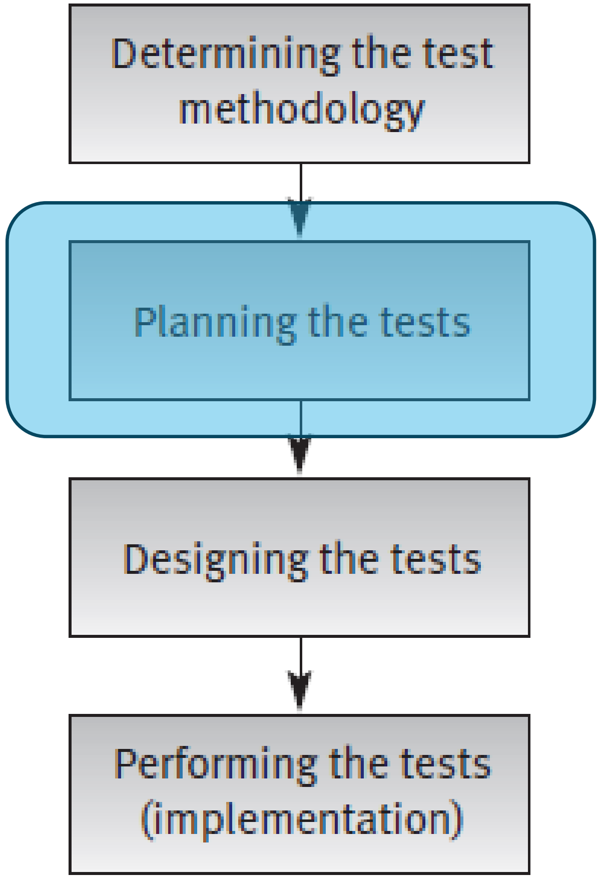
- Unit and integration testing are naturally carried out at the software developer's site.
- Location becomes important only when system tests are concerned: should they be performed at the developer's site or at the customer's site (the "target site")?
- If system testing is to be performed by external testing consultants, a third option arises: the consultant's site.
- The choice depends on the test's or system's computerized environment: as a rule, the computerized environment at the customer's site differs from that at the developer's site, despite efforts to "simulate" that environment. In such situations, apprehension regarding the occurrence of unpredicted failures once the system is installed at the customer's site is reduced as long as the customer is content with the system tests and plans no acceptance tests.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.5 WHEN ARE TESTS TERMINATED?

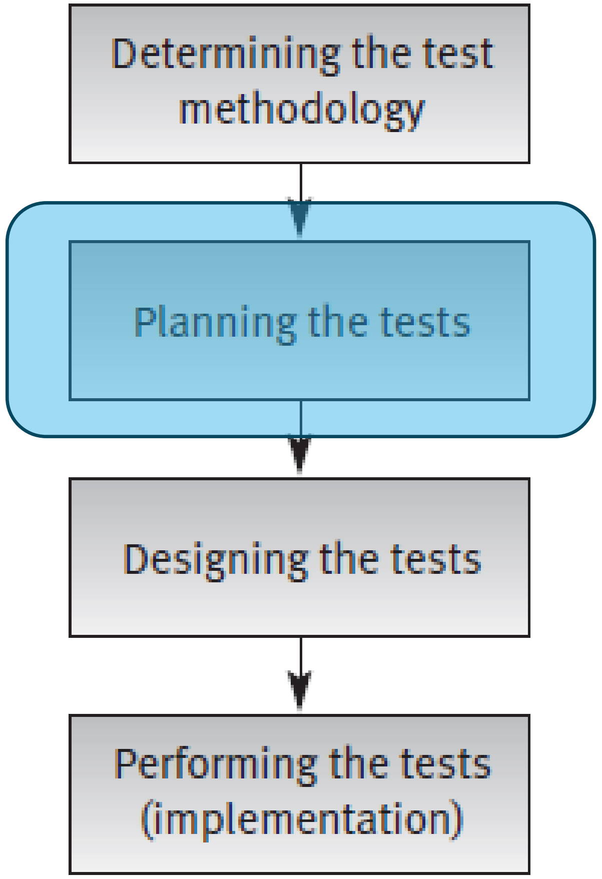
❑The decision about the stage at which software testing should be terminated is meaningful mainly with respect to system tests. The following five alternative routes are available, each chosen on the basis of different criteria.

1. **The completed implementation route.** According to this route, testing is terminated once the entire test plan has been carried out and error free ("clean") results are achieved for all the required regression tests. This alternative applies the perfection approach, which disregards budget and timetable constraints.
2. **The mathematical models application route.** When following this route, mathematical models are applied to estimate the percentage of undetected errors, based on the rate of error detection. Testing would be terminated once the error detection rate declines below the rate that corresponds to a predetermined level of undetected errors which is considered an acceptable software quality standard.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.5 WHEN ARE TESTS TERMINATED? CONT.

3. **The dual independent testing teams route.** If this route is adopted, two teams implement the testing process independently. By comparing the lists of detected errors provided by each team.
4. **Termination after resources have gone out.** Termination of this type occurs when budgets or the time allocated for testing run out. This situation, unfortunately not uncommon in the software industry
5. **The error seeding route.** According to this approach, errors of various types are seeded (hidden) in the tested software prior to the outset of testing. The underlying assumption of this route is that the percentage of discovered seeded errors will correspond to the percentage of real errors detected. Accordingly, testing will terminate once the residual percentage of undetected seeded errors reaches a predefined level considered acceptable for "passing" the system.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

2.5 WHEN ARE TESTS TERMINATED? CONT.

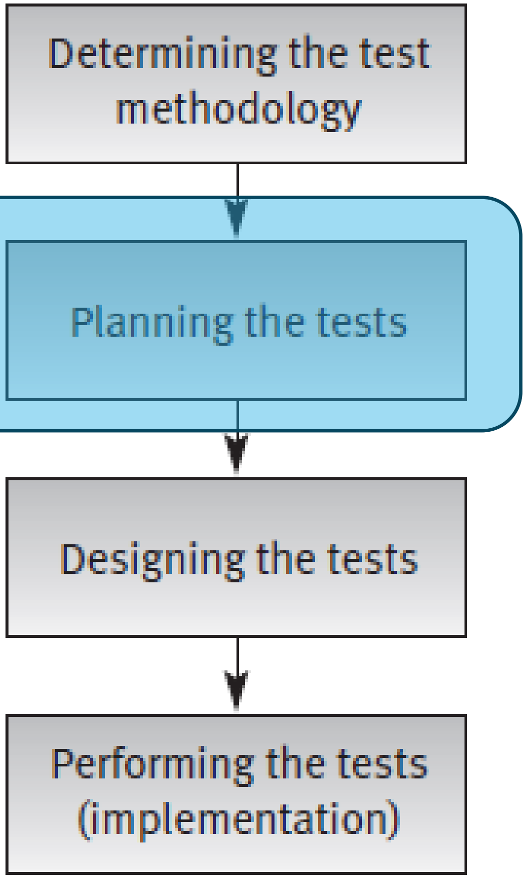
Once you consider terminating testing validation of the accuracy of the results within your organization's testing environment is of highest importance. Systematic follow-up is required for validation:

(1) Data collection. Collection of quality data on the errors detected in the project:

total number of code errors = **errors detected in the testing process** + **errors detected by the customer and the maintenance team during the first 6 or 12 months of regular software use.**

(1) Analysis of the error data. The analysis will compare estimates supplied by the models with the real figures.

(2) (3) Comparative analysis of severity of errors. Errors detected in the testing process are compared to errors detected by the customer and the maintenance team during the first 6 or 12 months of regular software use.


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

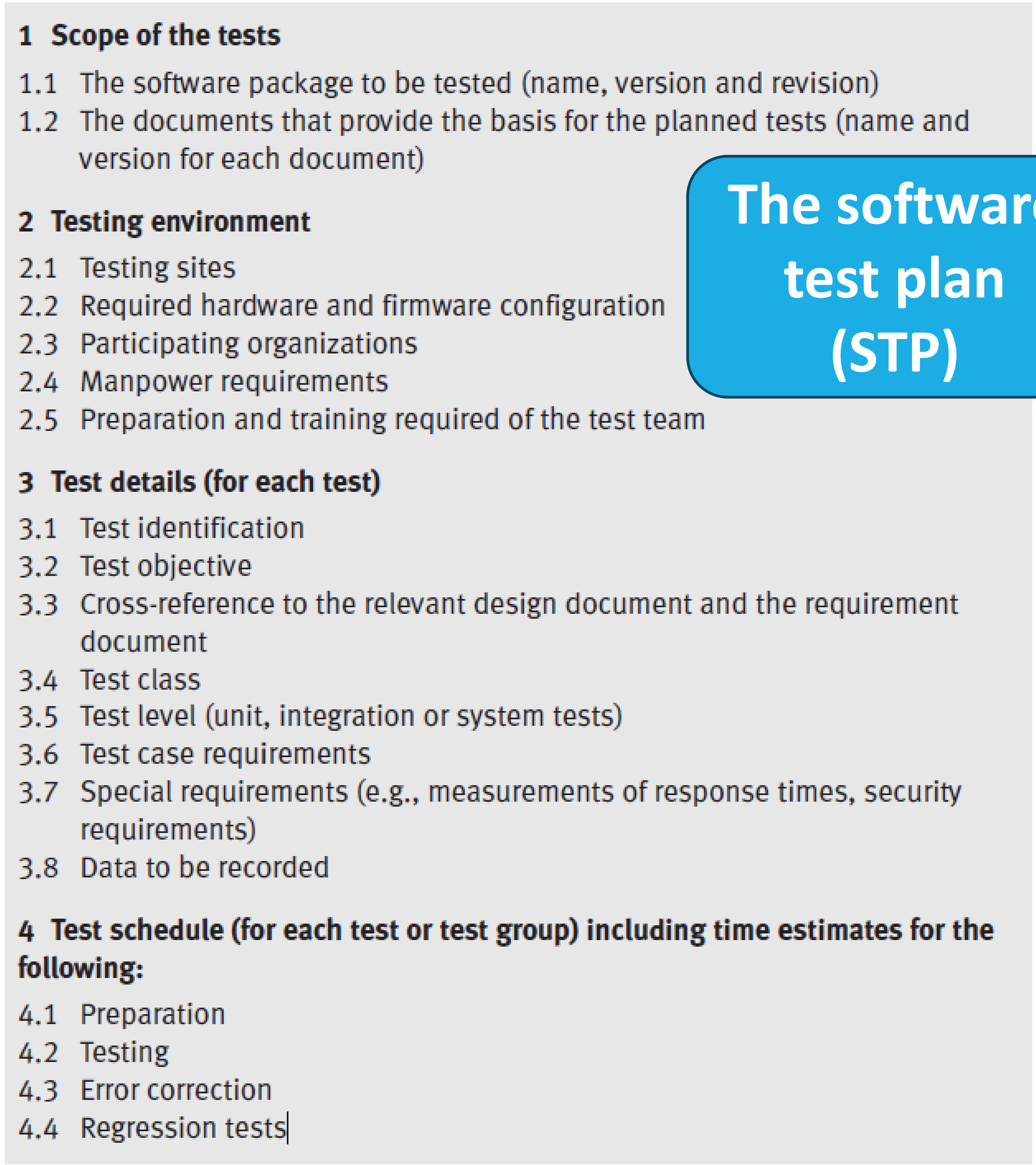
3. DESIGN THE TESTS

The products of the test design stage are:

1. Detailed design and procedures for each test

2. Test case database/file.

The software test plan (STP)



1 Scope of the tests

1.1 The software package to be tested (name, version and revision)
1.2 The documents that provide the basis for the planned tests (name and version for each document)

2 Testing environment

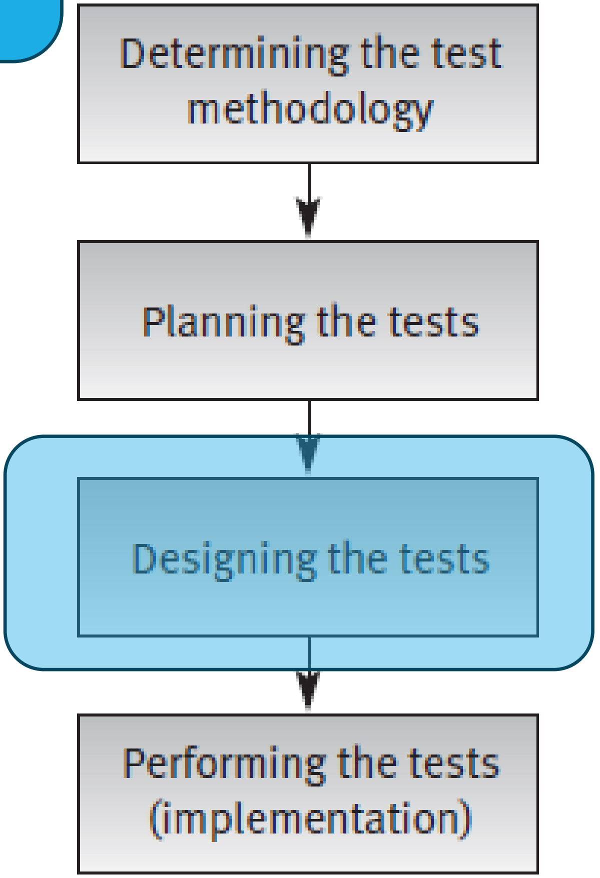
2.1 Testing sites
2.2 Required hardware and firmware configuration
2.3 Participating organizations
2.4 Manpower requirements
2.5 Preparation and training required of the test team

3 Test details (for each test)

3.1 Test identification
3.2 Test objective
3.3 Cross-reference to the relevant design document and the requirement document
3.4 Test class
3.5 Test level (unit, integration or system tests)
3.6 Test case requirements
3.7 Special requirements (e.g., measurements of response times, security requirements)
3.8 Data to be recorded

4 Test schedule (for each test or test group) including time estimates for the following:

4.1 Preparation
4.2 Testing
4.3 Error correction
4.4 Regression tests


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

3. DESIGN THE TESTS

Software test descriptions (STD)

1 Scope of the tests

1.1 The software package to be tested (name, version and revision)
1.2 The documents providing the basis for the designed tests (name and version for each document)

2 Test environment (for each test)

2.1 Test identification (the test details are documented in the STP)
2.2 Detailed description of the operating system and hardware configuration and the required switch settings for the tests
2.3 Instructions for software loading

3 Testing process

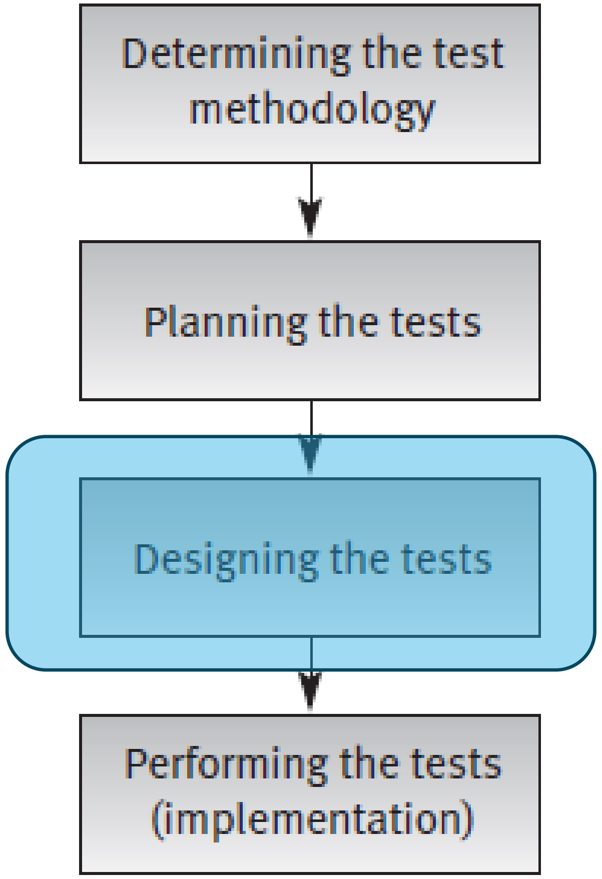
3.1 Instructions for input, detailing every step of the input process
3.2 Data to be recorded during the tests

4 Test cases (for each case)

4.1 Test case identification details
4.2 Input data and system settings
4.3 Expected intermediate results (if applicable)
4.4 Expected results (numerical, message, activation of equipment, etc.)

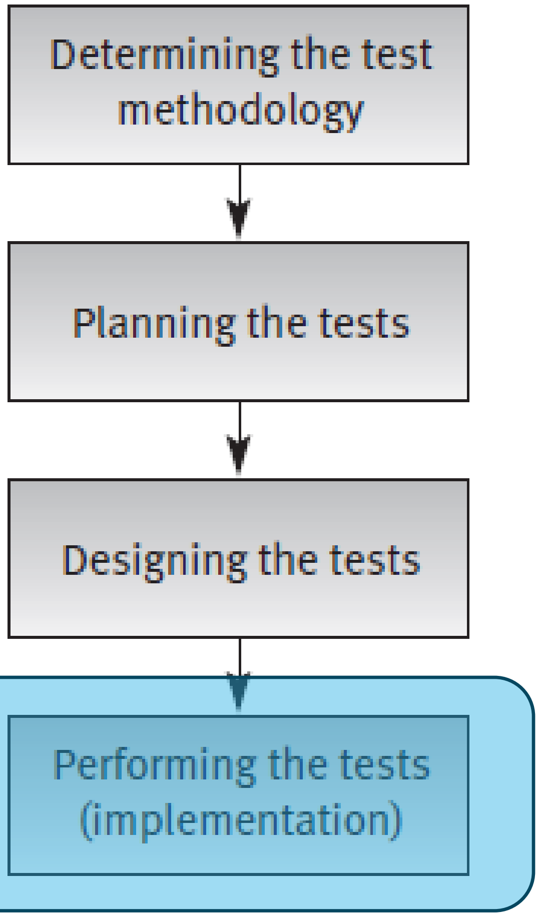
5 Actions to be taken in case of program failure/cessation

6 Procedures to be applied according to the test results summary

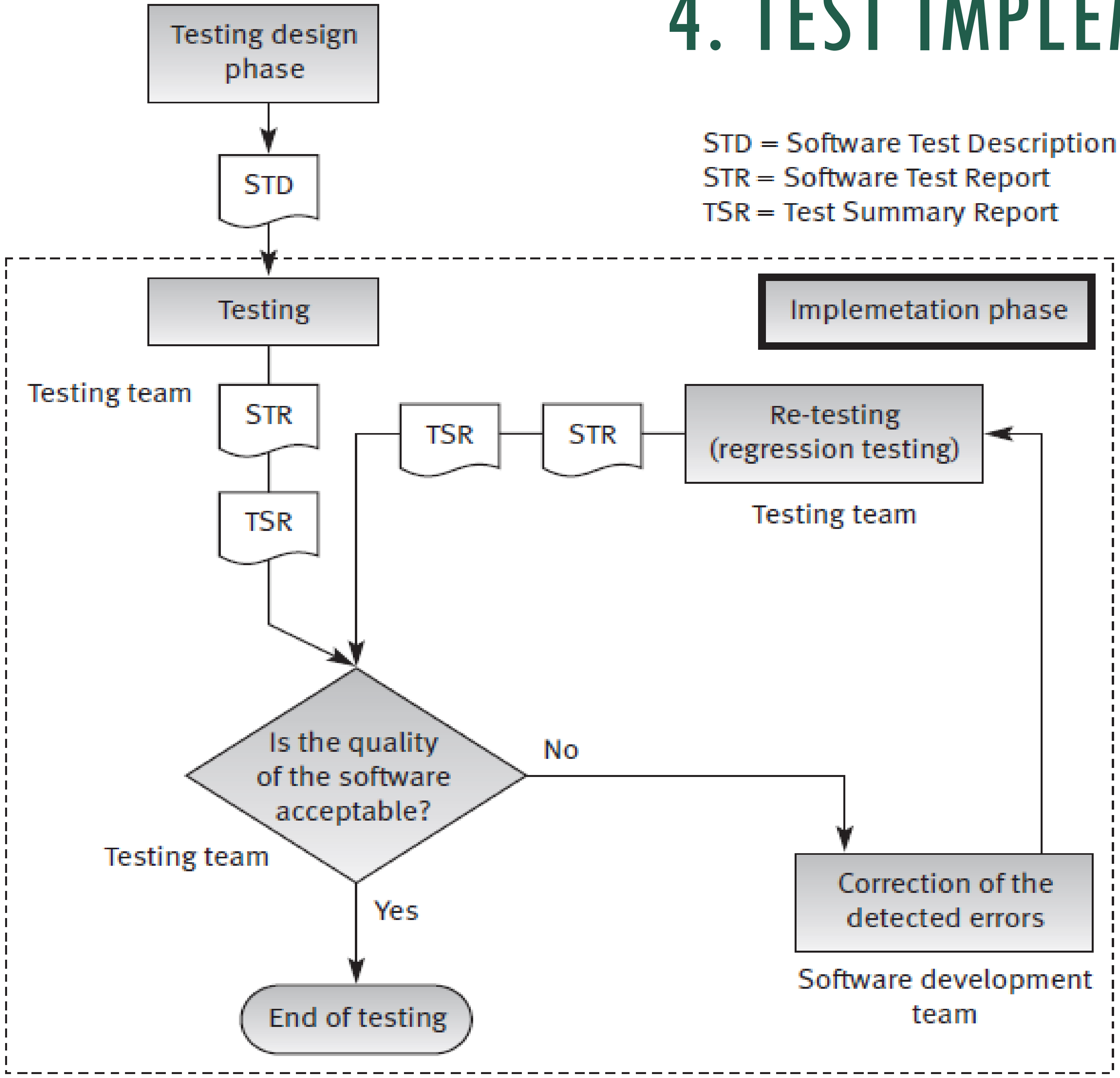

Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

4. TEST IMPLEMENTATION

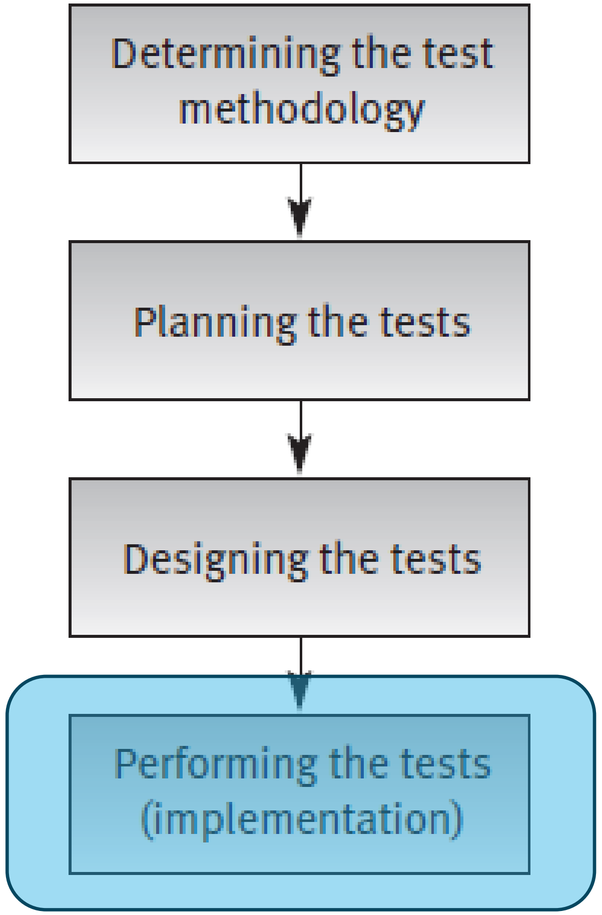
- Commonly, the testing implementation phase activities consist of a series of tests, corrections of detected errors and re-tests (regression tests).
- Testing is culminated when the re-test results satisfy the developers.
- The tests are carried out by running the test cases according to the test procedures. Documentation of the test procedures and the test case database/ file comprises the "software test description" (STD
- Re-testing (also termed "regression testing") is conducted to verify that the errors detected in the previous test runs have been properly corrected, and that no new errors have entered as a result of faulty corrections. It is


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

4. TEST IMPLEMENTATION CONT.

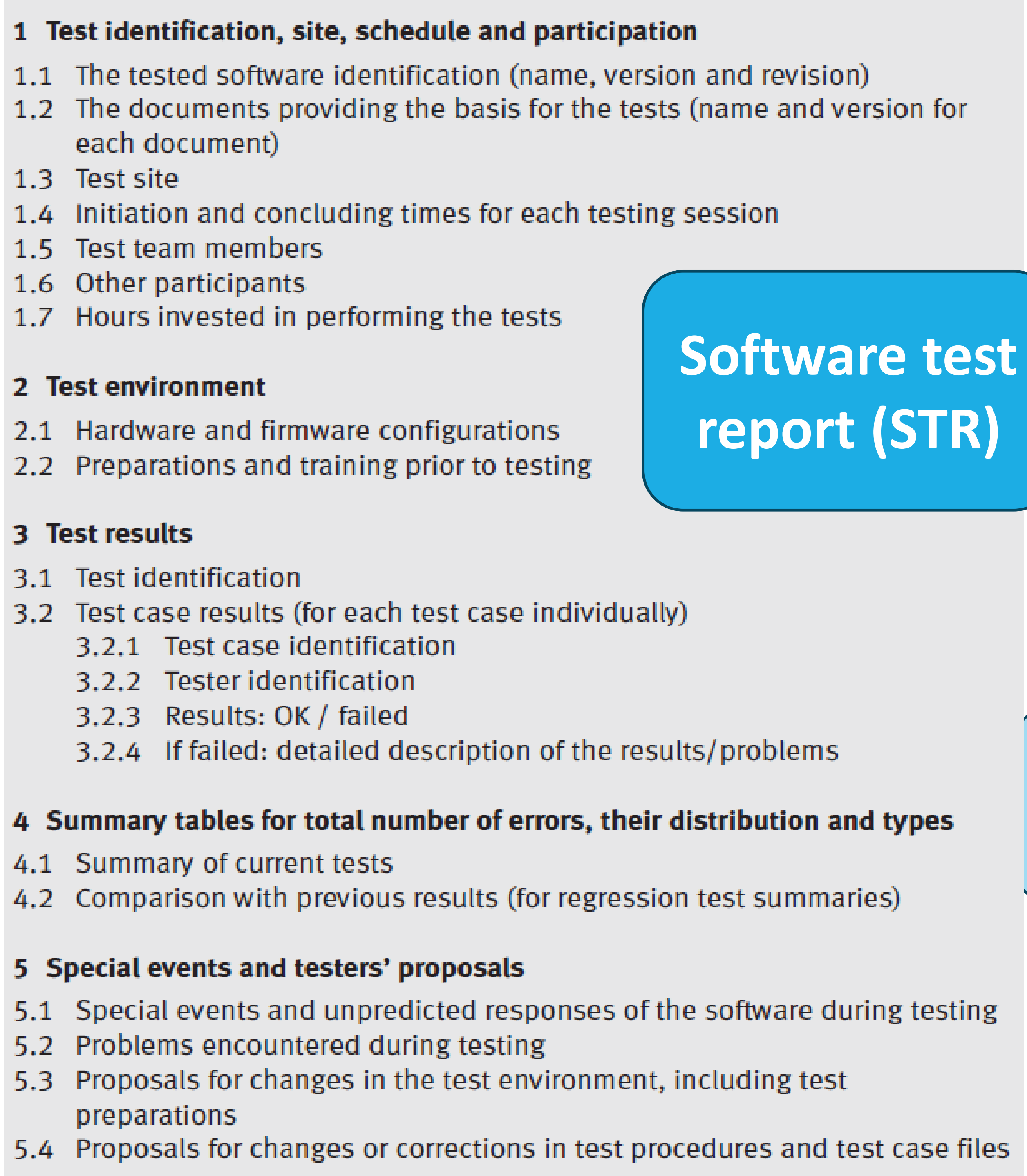


Testing design phase
STD
STD = Software Test Description
STR = Software Test Report
TSR = Test Summary Report
Testing
Implemetation phase
Testing team
STR
TSR
TSR
STR
Re-testing (regression testing)
Testing team
Is the quality of the software acceptable?
No
Testing team
Yes
Correction of the detected errors
Software development team
End of testing



Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

4. TEST IMPLEMENTATION CONT



1 Test identification, site, schedule and participation

1.1 The tested software identification (name, version and revision)
1.2 The documents providing the basis for the tests (name and version for each document)
1.3 Test site
1.4 Initiation and concluding times for each testing session
1.5 Test team members
1.6 Other participants
1.7 Hours invested in performing the tests

2 Test environment

2.1 Hardware and firmware configurations
2.2 Preparations and training prior to testing

3 Test results

3.1 Test identification
3.2 Test case results (for each test case individually)
  3.2.1 Test case identification
  3.2.2 Tester identification
  3.2.3 Results: OK / failed
  3.2.4 If failed: detailed description of the results/problems

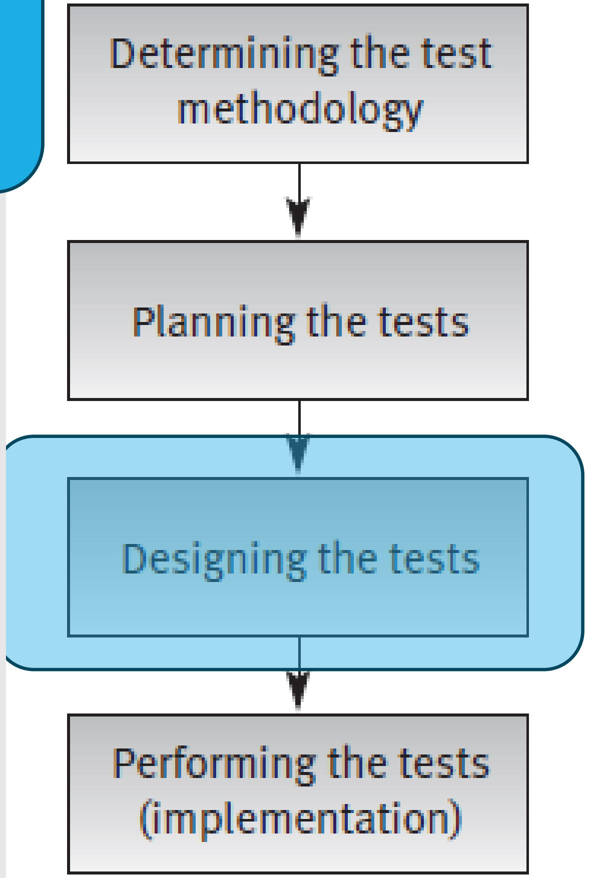
4 Summary tables for total number of errors, their distribution and types

4.1 Summary of current tests
4.2 Comparison with previous results (for regression test summaries)

5 Special events and testers' proposals

5.1 Special events and unpredicted responses of the software during testing
5.2 Problems encountered during testing
5.3 Proposals for changes in the test environment, including test preparations
5.4 Proposals for changes or corrections in test procedures and test case files

Software test report (STR)


Determining the test methodology
Planning the tests
Designing the tests
Performing the tests (implementation)

END