5 TEST MANAGEMENT

Geoff Thompson

INTRODUCTION

This chapter provides a generic overview of how testing is organised and how testing is managed within organisations. A generic view of testing will, inevitably, not match the way testing is organised in specific organisations. The issues addressed are nevertheless important for any organisation and need to be considered by all.

We will start by looking at how testing and risk fit together, as well as providing detailed coverage of test planning and the control of testing, and we will identify how independence assists the test process. One very important area in managing the test process is the understanding of the different roles and tasks associated with the testing role such as the test manager and the tester.

We cannot, in one chapter, provide all the knowledge required to enable the reader to become a practising test manager, but we do aim to provide the background information necessary for a reader to understand the various facets of the test management role.

Learning objectives

The learning objectives for this chapter are listed below. You can confirm that you have achieved these by using the self-assessment questions that follow the ‘Check of understanding’ boxes distributed throughout the text and the example examination questions provided at the end of the chapter. The chapter summary will remind you of the key ideas.

The sections are allocated a K number to represent the level of understanding required for that section; where an individual topic has a lower K number than the section as a whole, this is indicated for that topic; for an explanation of the K numbers, see the **Introduction**.

Test organization

- FL-5.1.1 Explain the benefits and drawbacks of independent testing.

- FL-5.1.2 Identify the tasks of a test manager and tester. (K1)

Test planning and estimation

- FL-5.2.1 Summarize the purpose and content of a test plan. (K2)
- FL-5.2.2 Differentiate between various test strategies. (K2)
- FL-5.2.3 Give examples of potential entry and exit criteria. (K2)
- FL-5.2.4 Apply knowledge of prioritization, and technical and logical dependencies, to schedule test execution for a given set of test cases.
- FL-5.2.5 Identify factors that influence the effort related to testing. (K1)
- FL-5.2.6 Explain the difference between two estimation techniques: the metrics-based technique and the expert-based technique. (K2)

Test monitoring and control

- FL-5.3.1 Recall metrics used for testing. (K1)
- FL-5.3.2 Summarize the purposes, contents, and audiences for test reports.

Configuration management

- FL-5.4.1 Summarize how configuration management supports testing.

Risks and testing

- FL-5.5.1 Define risk level by using likelihood and impact. (K1)
- FL-5.5.2 Distinguish between project and product risks.
- FL-5.5.3 Describe, by using examples, how product risk analysis may influence the thoroughness and scope of testing.

Defect management

- FL-5.6.1 Write a defect report, covering defects found during testing.

Self-assessment questions

The following questions have been designed to enable you to check your current level of understanding for the topics in this chapter. The answers are at the end of the chapter.

Question SA1 (K1)
Which of the following is a valid exit criterion from the test execution phase?

a. All tests have been defined.

b. All defects reported have been corrected.

c. All planned tests have been executed.

d. All testing tasks have been assigned.

Question SA2 (K2)
Which of the following are *most* likely to be used when developing a test strategy or test approach?

i. Failure-based approach.

ii. Test specification approach.

iii. Model-based approach.

iv. Analytical-based approach.

a. iii and iv.

b. i and iv.

c. ii and i.

d. i and iii.

Question SA3 (K1)
What can a risk-based approach to testing provide?

a. The types of test techniques to be employed.

b. The total tests needed to provide 100 per cent coverage.

c. An estimation of the total cost of testing.

d. Only that test execution is effective at reducing risk.

RISK AND TESTING

It is not possible to talk about test management without first looking at risk and how it affects a generic test process as defined in **Chapter 1**. If there were no risk of adverse future events in software or hardware development, then there would be no need for testing. In other words, if risks did not exist then neither would testing.

Risk can be defined as the chance of an event, hazard, threat or situation occurring and its undesirable consequences:

> Risk – a factor that could result in future negative consequences, usually expressed as impact and likelihood.

In a project, a test manager will manage two different types of risk: project and product. In both instances the calculation of the risk will be:

> Level of risk = probability of the risk occurring × impact if it did happen

Project risks

While managing the testing project, a test manager will use project risks to manage the capability to deliver.

Project risks include:

- Project issues:
  - Delays in delivery.
  - Inaccurate estimates.
  - Late changes.
- Organisational issues:
  - Skills and training or staff may be inadequate.
  - Personal issues between staff impacting progress.
  - Users, business staff or subject matter experts may be unavailable when needed.
- Political issues:
  - Testers may not communicate their needs and/or test results adequately.
  - Developers and/or testers may fail to follow up on information found in testing and reviews; for example not following up on process improvements identified.
  - There may be an improper attitude to or understanding of the value of testing.
- Supplier issues:
  - Failure of a third party to deliver on time or at all.
  - Contractual issues, such as meeting acceptance criteria.
- Technical issues:
  - Problems in defining the right requirements.
  - The extent that requirements can be met given existing project constraints.
  - Test environment not ready on time.
  - Late data conversion, migration planning and development, and testing data conversion /migration tools.
  - Weakness in the development process that impacts the quality of the work products.
  - Poor defect management resulting in an increase in technical debt.
  - Low quality of the design, code, configuration data, test data and tests.

For each risk found, a probability (chance of the risk being realised) and impact (what will happen if the risk is realised) should be identified as well as the identification and management of any mitigating actions (actions aimed at reducing the probability of a risk occurring, or reducing the impact of the risk if it did occur).

So, for example, if there was a risk identified that the third-party supplier may be made bankrupt during the development, the test manager would review the supplier's accounts and might decide that the probability of this is medium (1 on a scale of 1 to 5, 1 being a high risk and 5 a low one). The impact on the project if this did happen would be very high (1 using the same scale). The level of risk is therefore $3 \times 1 = 3$. The lower the number, the more the risk. With 3 being in the medium risk area, the test manager would now have to consider what mitigating actions to take to try to stop the risk becoming a reality. This might include not using the third party or ensuring that payment for third-party deliverables is made efficiently.

When analysing, managing and mitigating these risks, the test manager is following well-established project management principles provided within project management methods and approaches. The project risks recognised during test planning should be documented in the test plan (see later in this chapter for details of the test plan); for the ongoing management and control of existing and new project risks, a risk register should be maintained by the test manager.

Product risks

When planning and defining tests, a test manager or tester using a risk-based testing approach will be managing product risks.

Product risks are risks to the quality of the product. In other words, the potential of a defect occurring in the live environment is a product risk. Examples of product risks are:

- Failure-prone software delivered – not able to perform as intended according to the specification and/or the user requirements.
- System architecture may not adequately support non-functional requirement(s) (e.g. security, reliability, usability, performance).
- A particular computation may be performed incorrectly in certain circumstances.
- A loop structure may be coded incorrectly.
- Feedback from users indicates that the product may not meet expectations.
- The potential that a defect in the software/hardware could cause harm to an individual or company.
- Poor data integrity and quality (e.g. data migration issues, data conversion problems, data transport problems, violation of data standards).
- Software that does not perform its intended functions.

Risks are used to decide where to start testing in the Software Development Life Cycle; for example, the risk of poor requirements could be mitigated by the use of formal reviews as soon as the requirements have been documented at the start of a project. Product risks also provide information enabling decisions regarding how much testing should be carried out on specific components or systems; for example, the more risk there is, the more detailed and comprehensive the testing may be. In these ways testing is used to reduce the risk of an adverse effect (defect) occurring or being missed.

Mitigating product risks may also involve non-test activities. For example, in the poor requirements situation, a better and more efficient solution may be simply to replace the analyst who is writing the poor requirements in the first place.

As already stated, a risk-based approach to testing provides proactive opportunities to reduce the levels of product risk starting in the initial stages of a project. It involves the identification of product risks and how they are used to guide the test planning, specification and execution. In a risk-based approach, the risks identified:

- will determine the test techniques to be employed, and/or the extent of testing to be carried out; for example, the Motor Industry Software Reliability Association (MISRA) defines which test techniques should be used for each level of risk: the higher the risk, the higher the coverage required from test techniques;
- will determine the levels and types of testing to be performed, such as security testing or accessibility testing;
- will determine the extent of testing to be carried out; for example what the depth of test coverage should be;
- prioritise testing in an attempt to find the critical defects as early as possible; for example, by identifying the areas most likely to have defects (the most complex) the testing can be focused on these areas;
- will determine any non-test activities that could be employed to reduce risk; for example, to provide training to inexperienced designers.

Risk-based testing draws on the collective knowledge and insights of the project stakeholders, testers, designers, technical architects, business reps and anyone with knowledge of the solution to determine the risks and the levels of testing required to address those risks.

To ensure that the chance of a product failure is minimised, risk management activities provide a disciplined approach:

- To analyse (and re-evaluate on a regular basis) what can go wrong. Reviews of existing product risks and looking for any new product risks should occur periodically throughout the life cycle.

- To determine what risks are important to deal with (probability × impact). As the project progresses, owing to the mitigation activities, risks may reduce in importance, or disappear altogether.
- To implement actions to deal with those risks (mitigating actions).
- To make contingency plans to deal with risks should they become actual events.

Testing supports the identification of new risks by continually reviewing risks of the project deliverables throughout the life cycle; it may also help to determine what risks are important to reduce by setting priorities; it may lower uncertainty about risks by, for example, testing a component and verifying that it does not contain any defects; and lastly by running specific tests it may verify other strategies that deal with risks, such as contingency plans.

Testing is a risk control activity that provides feedback about the residual risk in the product by measuring the effectiveness of critical defect removal and by reviewing the effectiveness of contingency plans.

CHECK OF UNDERSTANDING

1. What are the two types of risks that have to be considered in testing?
2. Compare and contrast these two risk types.
3. How early in the life cycle can risk impact the testing approach?
4. What does MISRA determine when the level of risk is understood?

TEST ORGANISATION

Test organisation and independence

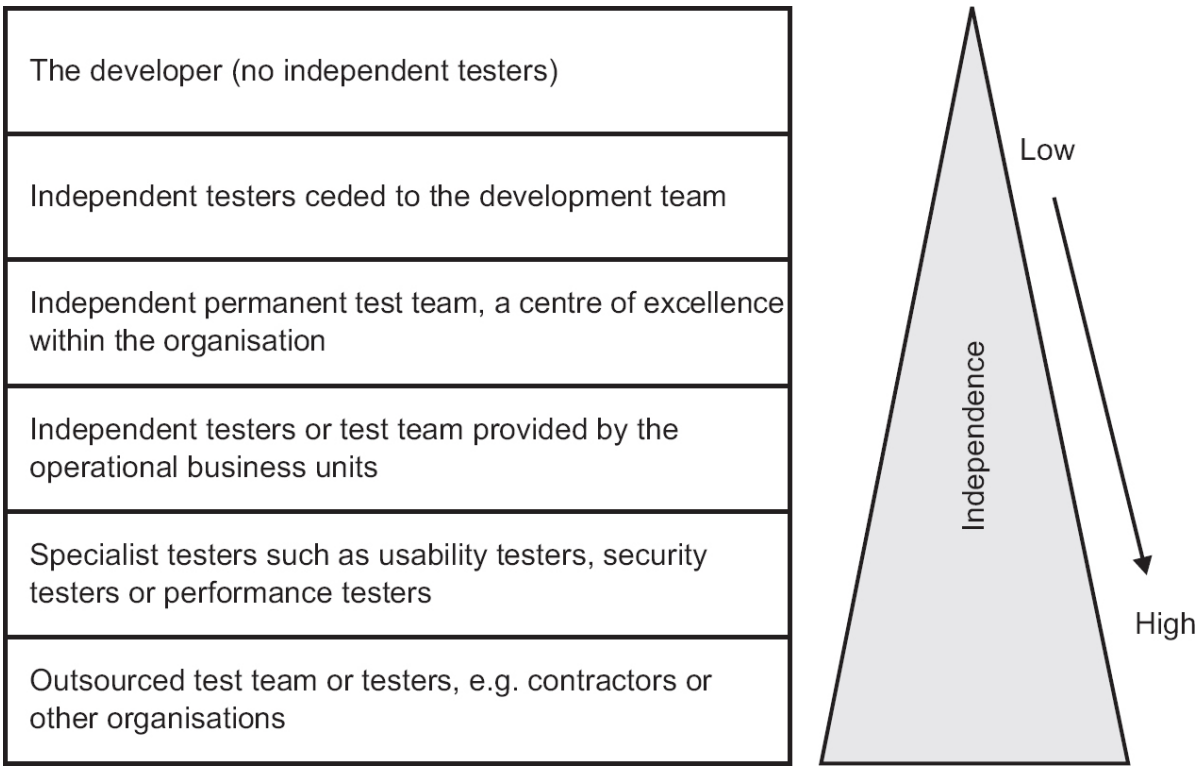
Independent testing is testing carried out by someone other than the creator (developer) of the item being tested. By remaining independent, it is possible to improve the effectiveness of testing if implemented correctly.

As humans we are all capable of making mistakes, from the simplest misspelling or wrong use of syntax to fundamental errors at the core of any documents we write. The problem is that as authors we are less able to see our own errors than someone else, who is less directly associated with the document, would be. This is a problem that is made worse, in the world of software development, by the differing ‘world view’ of testers and developers. A developer, as the creator and owner of documents and code related to development, perceives these deliverables as being correct when they are delivered. The general awareness that we all make mistakes is, at this stage, overridden by the belief that what has been produced is what is required. A tester, by contrast, will take the view that anything delivered for testing is likely to contain errors and will search diligently to identify and locate those errors.

This is where independent testing is important because it is genuinely hard for authors to identify their own errors, but it is easier for others to see them. There are many options for many levels of independence. In general, the more remote a tester is from the production of the item, the greater is the level of independence. Figure 5.1 indicates the most common roles and the levels of independence they bring.

Of course, independence comes at a price. The greater the level of independence, the greater the likelihood of errors in testing arising from unfamiliarity. Levels of independence will also depend on the size of the organisation. In smaller organisations where everybody contributes to every activity, it is harder to differentiate the role of the tester from any other role, and therefore testers may not be very independent at all. The key in these circumstances is for the testers to have independence of mind, not necessarily to be in an independent (separate) team. In organisations where there are clearly defined roles, it is a lot easier for a tester to remain independent.

Figure 5.1 Levels of independent testing



It is also possible to mix and match the levels of independence; for example, a test team made up of permanent resources, business unit resources and contractors. For large, complex or safety-critical projects, it is usually best to have multiple levels of testing, with some or all of the levels done by independent testers.

The Agile approach to development challenges the traditional approach to independence. In this approach everybody takes on multiple roles, so maintaining total independence is not always possible. A tester in this situation has to be able to switch to an independent view, at the relevant points in the project. Testers achieve this independence of view by not assuming anything and by

not starting to own the software in the way that a developer might; for example, taking the view that the way the software works is the way it was developed to work.

Independence in the implementation of testing has some key benefits and drawbacks, as in Table 5.1.

Table 5.1 Features of independent testing

Benefits	Drawbacks
• Independent testers are likely to recognise different kinds of failures compared to developers, because of the different backgrounds, technical perspectives and biases. • The tester can see what has been built rather than what the developer thought had been built. • The tester verifies, challenges or disproves assumptions made by stakeholders during specification and implementation of the system.	• Isolation from the development team leading to a lack of collaboration, delays in providing feedback to the development team, or an adversarial relationship between test and development. • Developers lose a sense of responsibility for quality because it may be assumed that they need not worry about errors as the independent test team will find them. • Independent testers can be seen as a bottleneck or to blame for delays in releases. • Independent testers may lack some of the important information (e.g. about the test object), as they have no connection to the stakeholders or developers.

CHECK OF UNDERSTANDING

1. Why is independent testing more effective at finding errors than simply allowing the developer and author to test their own product?
2. Name two benefits of independence.
3. Which organisation provides the lowest level of independence and which provides the highest?

Tasks of a test manager and tester

Test tasks are traditionally carried out by people who make testing a career; however, test tasks may also be carried out by non-testers such as a project manager, quality manager, developer, business and domain expert, infrastructure personnel or IT operations. The availability of resources usually determines the resource types that are deployed on each project; for example, if

there are no career testers available an organisation may identify non-testing IT or business resources to carry out the role of tester for a specific project or time period.

The syllabus defines two testing roles: the test manager and the tester. Other roles may exist in your organisation, but they are not covered here.

The testing roles can be undertaken by anyone with the required skills or anyone who is given the right training. For example, the role of a test manager could be undertaken by a project manager. The decision as to who does what will depend on how a project or organisation is structured, as well as the size and number of resources working on a given project.

It is important to understand here the difference between a testing role and a testing job. A role is an activity, or a series of activities, given to a person to fulfil; for example, the role of test manager. A person may therefore have more than one role at any moment depending on their experience and the level of workload on a project. A job is effectively what an individual is employed to do, so one or many roles could make up a job. For example, a test manager could also be a tester.

The tasks undertaken by a test manager align very closely with those undertaken by a project manager and align closely with standard approaches to project management. In this context a test manager is anyone who leads a team of testers (be that one or many testers). Test managers are also known as test programme managers, test team leaders and test coordinators.

Typical test manager tasks may include:

- Coordinating or developing the test policy and test strategy for the organisation.
- Planning the test activities by considering the context and understanding the test objectives and risks. This may include selecting test approaches, estimating test time, effort and cost, acquiring resources, defining test levels and test cycles, and planning defect management.
- Writing and updating test plan(s).
- Coordinating the test plan(s) with project managers, product owners and others.
- Sharing test perspectives with other project activities, such as the code integration planning.
- Initiating the analysis, design, implementation and execution of tests, monitoring test progress and results, and checking the status of execution criteria (or definition of ‘done’).
- Preparing and delivering test progress reports and test summary reports based on the information gathered.
- Adapting planning based on test results and progress; for example if more defects than planned are found, this will impact the time taken to complete testing and so action will need to be taken to realign the plan.
- Supporting the setting up of the defect management system and adequate configuration management of testware.

- Introducing suitable metrics for measuring test progress and evaluating the quality of the testing and the product.
- Supporting the selection and implementation of tools to support the test process, including budget, and the allocation of time for the effort required to build and support tools.
- Deciding about the implementation of test environment(s).
- Promoting and advocating the tester, the test team, and the test profession within the organisation.
- Developing the skills and careers of testers through training plans, performance evaluations, coaching and so on.

These tasks are not, however, all of the tasks that could be carried out by test managers, just the most common ones. In fact, other resources could take on one or more of these tasks as required, or they may be delegated to other resources by the test manager. In Agile development, some of the above tasks will be handled by the Agile team, especially with reporting. The key is to ensure that everyone is aware of who is doing what tasks, that they are completed on time and within budget, and that they are tracked through to completion.

The other role covered by the syllabus is that of the tester, also known as test analyst or test executor.

The tasks typically undertaken by a tester may include:

- Reviewing and contributing to test plans.
- Analysing, reviewing and assessing user requirements, user stories and acceptance criteria, specifications and models for testability.
- Creating test specifications from the test basis; for example test conditions, and the traceability between test cases, test conditions and the test basis.
- Setting up the test environment (often coordinating with system administration and network management). In some organisations the setting up and management of the test environment could be centrally controlled; in this situation a tester would directly liaise with the environment management to ensure that the test environment is delivered on time and to specification.
- Designing and implementing test cases and test procedures.
- Preparing and acquiring/copying/creating test data.
- Executing tests on all test levels, logging the tests, evaluating the results and documenting the deviations from expected results as defects.
- Using test administration, or management and test monitoring tools as required.
- Automating tests (may be supported by a developer or a test automation expert).
- Evaluating non-functional characteristics such as performance efficiency, reliability and usability.

- Reviewing tests developed by other testers.

As mentioned earlier, the thing to remember when looking at roles and tasks within a test project is that one person may have more than one role and carry out some or all of the tasks applicable to the role. This is different to having a 'job': a 'job' may contain many roles and tasks.

CHECK OF UNDERSTANDING

1. What other names are given to the test manager role?
2. Detail five possible tasks of a test manager.
3. Detail five possible tasks of a tester.
4. Describe the differences between a test manager role and a test manager task.

TEST STRATEGY AND TEST APPROACHES

The test strategy will define how testing will be implemented either in a project or company-wide. It can be:

- developed early in the life cycle, which is known as preventative – in this approach the test design process is initiated as early as possible in the life cycle to stop defects being built into the final solution;
- left until just before the start of test execution, which is known as reactive – this is where testing is the last development stage and is not started until after design and coding have been completed (sometimes it is identified as the waterfall approach, i.e. all development stages are sequential, the next not starting until the previous one has nearly finished).

There are many strategies that can be employed, and they may include:

- Analytical strategies rely on the analysis of some factor such as risk-based testing, where testing is directed to areas of greatest risk (see earlier in this chapter for an overview of risk-based testing).
- Model-based strategies base tests on a model such as statistical information about failure rates (such as reliability growth models) or usage models (such as operational profiles).
- Methodical strategies rely on the systematic use of some predefined tests or test conditions such as failure based (including error guessing and fault attacks), checklist based and quality-characteristic based.
- Process-compliant (or standard-compliant) strategies adhere to the processes developed for use with standards (see ISO/IEC/IEEE 29119-2 or MISRA) and various Agile or traditional waterfall approaches.

- Reactive (or dynamic and heuristic) strategies, such as exploratory testing where testing is more reactive to events than pre-planned, and where execution and evaluation are concurrent tasks.
- Directed (or consultative) strategies, such as those where test coverage is driven primarily by the advice and guidance of technology and/or business domain experts outside or within the test team.
- Regression-averse strategies are designed with the desire to avoid regression of existing capabilities such as those that include reuse of existing test material, extensive automation of functional regression tests and standard test suites.

Different strategies may be combined if required. The decision as to how and why they will be combined will depend on the circumstances prevalent in a project at the time. For example, an organisation may as a standard use an Agile method, but in a particular situation the structure of the test effort could use a risk-based approach to ensure that the testing is correctly focused.

A test strategy will contain generalised descriptions of the test processes to be used; the test approach provides tailoring of the strategy for a particular project or projects. A test approach includes all of the decisions made on how testing should be implemented, based on the (test) project goals and objectives, as well as the risk assessment. It forms the starting point for test planning, selecting the test design techniques and test types to be employed. It should also define the software under test and test entry and exit criteria, often called the definition of ‘done’ in Agile projects.

The selected approach will depend on the context within which the test team is working, and may consider risks, hazards and safety, available resources and skills, the technology, the nature of the system (e.g. custom built versus COTS), test objectives and regulations.

CHECK OF UNDERSTANDING

1. Name and explain five approaches to the development of the test approach or test strategy.
2. Name one of the standards referred to that dictate the test approach.
3. Can discretion be used when defining a test approach and, if so, what can influence the decision as to which way to approach testing?

TEST PLANNING AND ESTIMATION

Test planning

Test planning is the most important activity undertaken by a test manager in any test project. It ensures that there is initially a list of tasks and milestones in a baseline plan to track progress

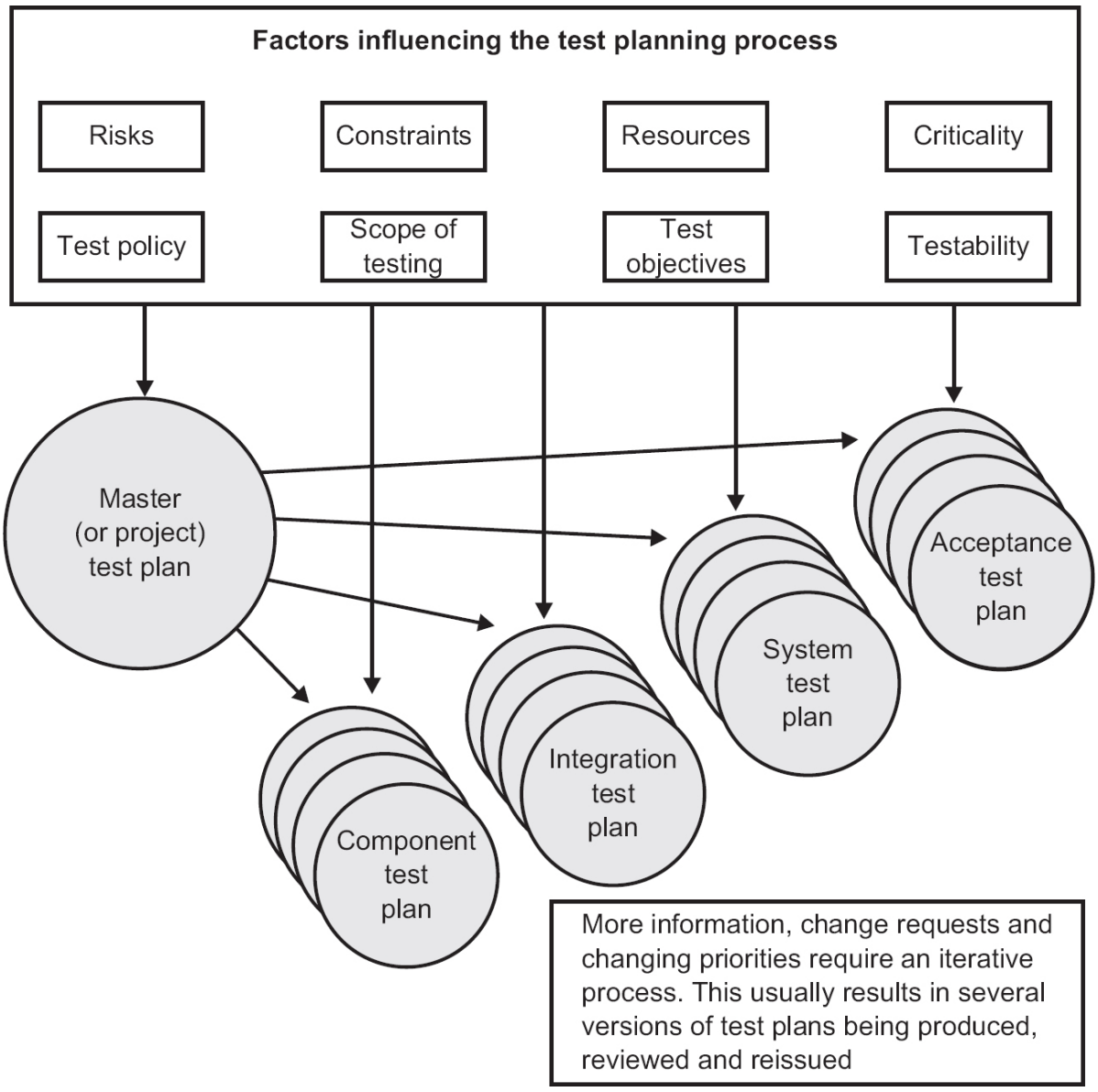
against, as well as defining the shape and size of the test effort. Test planning is used in development and implementation projects (sometimes called ‘greenfield’) as well as maintenance (change and fix) activities.

The main document produced in test planning is often called a master test plan or a project test plan. This document defines at a high level the test activities being planned. It is normally produced during the early phases of the project (e.g. initiation) and updated as required via change control as the project develops. It will provide sufficient information to enable a test project to be established (bearing in mind that at this point in a project little more than requirements may be available from which to plan).

The details of the test-level activities are documented within test-level plans, for example the system test plan. These documents will contain the detailed activities and estimates for the relevant test level.

Figure 5.2 shows where test-level test plans fit into the V model. It shows how a test plan exists for each test level and that they will usually refer to the master test plan.

Figure 5.2 Test plans in the V model



The contents sections of a test plan for either the master test plan or test-level plans are normally identical or very similar. ISO/IEC/IEEE 29119-3, the Software Testing test document standard, contains details of what the content of the plans should be.

The ISO 29119-3 Software Testing Standard identifies that there should be a minimum of 15 sections present in a test plan, as in Table 5.2.

Test planning is a continual activity that spans the life of the test project; it takes place in all life cycle stages. As risks and changes occur, the plan and planning should be amended to recognise these and reflect the current position. The plans will have been baselined (locked down) after initial sign-off, so these changes would normally be managed by the project change process. Baselining a document effectively secures it from further change unless authorised via a change control process.

Table 5.2 Test plan sections

Section no.	Heading	Details
1	Overview	Identifies the document and describes the origins and history
2	Unique identification of the document	The specific unique identifier allocated to this document, e.g. TPR 00001
3	Issuing organisation	Specifies who is responsible for completion and distribution of the document
4	Approval authority	Identifies who is responsible for reviewing and signing off the document before it is issued
5	Change history	A record of each version of the document and any changes that were included for each version
6	Introduction	Explanatory information about the content and structure of the document
7	Scope	Defines the areas of coverage included within the document, test activities, etc.
8	References	Lists referenced documents and identifies repositories for system, software and test information. The references may be separated into 'external' references that are imposed from outside the organisation and 'internal' references that are imposed from within the organisation
9	Glossary	A glossary that defines the terms, abbreviations and acronyms, if any, used in the document

10	Context of testing	Includes: • Details of the project or sub-processes covered by the plan • Test items – items to be tested • Test scope – what is and isn't included within the scope of testing • Assumptions and constraints • Who the stakeholders are • Testing communication lines inside and outside the test team
11	Risk register	A list of the project and product risks
12	Test strategy	Describes the test approach in the following subsections: • Test sub-process – what test sub-process will be conducted • Test deliverables – documents that will be delivered by the test project • Test design techniques to be used and when • Test completion criteria (exit and entry criteria) • Metrics to be collected • Test data requirements • Test environment requirements • Retesting and regression testing approach • Suspension criteria • Deviations from the organisational test strategy
13	Testing activities and estimates	Documents the activities to be undertaken during testing and the estimate of time required to complete those activities
14	Staffing	Details of the staffing requirements to complete the test plan; this will include roles and responsibilities, hiring needs and any training requirements
15	Schedule	Testing milestones

Test-planning activities

During test planning various activities for an entire system or a part of a system have to be undertaken by those working on the plan. They include:

- Working with the project manager and subject matter experts to determine the scope and the risks that need to be tested. Also identifying and agreeing the objectives of the testing, be

they time, quality or cost focused, or a mixture of all three. The objectives will enable the test project to know when it has finished – has time or money run out, or has the right level of quality been met?

- Understanding what delivery model is to be used (waterfall, iterative, Agile, etc.) and defining the overall approach of testing (sometimes called the test strategy) based on this, ensuring that the test levels and entry and exit criteria are defined.
- Liaising with the project manager and making sure that the testing activities have been included within the software life cycle activities such as:
  - design – the development of the software design;
  - development – the building of the code;
  - implementation – the activities surrounding implementation into a live environment.
- Working with the project to decide what needs to be tested, what roles are involved and who will perform the test activities, planning when and how the test activities should be done, deciding how the test results will be evaluated, and defining when to stop testing (exit criteria).
- Building a plan that identifies when and who will undertake the test analysis and design activities. In addition to the analysis and design activities test planning should also document the schedule for test implementation, execution and evaluation. The plan can either be sequential; for example particular dates are defined, or iterative, where the context of each iteration will need to be considered.
- Deciding what the documentation for the test project will be; for example, which plans, how the test cases will be documented and so on.
- Defining the management information, including the metrics required, and putting in place the processes to monitor and control test preparation and execution, defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets; for example, test cases.

ENTRY CRITERIA AND EXIT CRITERIA (DEFINITION OF ‘READY’ OR DEFINITION OF ‘DONE’)

Entry criteria are used to determine when a given test activity can start. This could include the planning, when test design and/or when test execution for each level of testing is ready to start. Entry criteria (also known as definition of ‘ready’ in Agile projects) define the preconditions for undertaking a test activity.

Examples of some typical entry criteria to test execution (for example) may include:

- Availability of testable requirements, user stories or models.
- Test environment available and ready for use (it functions).
- Test tools installed in the environment are ready for use.

- Testable code is available.
- All test data is available and correct.
- All previous test activity has completed and met its exit criteria.

Exit criteria are used to determine when a given test activity has been completed or when it should stop, typically called the definition of ‘done’ in an Agile project. Exit criteria can be defined for all of the test activities, such as planning, specification and execution as a whole, or to a specific test level for test specification as well as execution.

Exit criteria should be included in the relevant test plans.

Some typical exit criteria might be:

- All tests planned have been executed.
- A certain level of coverage has been achieved.
- The number of unresolved defects is within an agreed limit.
- All high-risk areas have been fully tested, with only minor residual risks left outstanding.
- Cost – when the budget has been spent.
- The number of estimated remaining defects is sufficiently low.
- The evaluated level of quality criteria, such as reliability and performance, is sufficient.
- The schedule has been achieved; for example, the release date has been reached and the product has to go live. This was the case with the millennium testing (it had to be completed before midnight on 31 December 1999), and is often the case with government legislation.

Exit criteria should have been agreed as early as possible in the life cycle; however, they can be, and often are, subject to controlled change as the detail of the project becomes better understood and therefore the ability to meet the criteria is better understood by those responsible for delivery.

CHECK OF UNDERSTANDING

1. What is the international standard for testing called?
2. Identify the 15 sections of the test plan.
3. What activities are contained within test planning?
4. Detail four typical exit criteria.

TEST EXECUTION SCHEDULE

Having developed various test cases and test procedures (including any automated test procedures), which have been assembled into test suites, the test suites can be arranged into a test

execution schedule. A test execution schedule documents what test suite will be run in what order and on what day. The order of the execution of test suites will be determined by many things such as prioritisation, processing dependencies; for example suite 1 has to run before suite 7 can be run, whether there are confirmation and regression tests, and finally in the most efficient sequence possible.

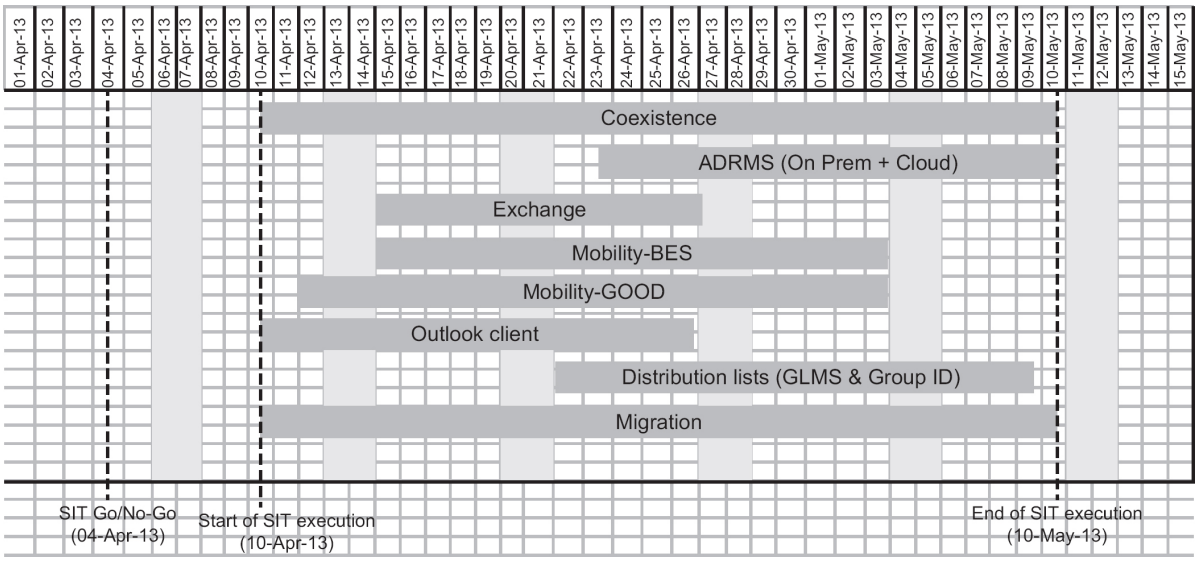
Figure 5.3 (page 175) reflects a high-level test execution schedule for the various system components of a Microsoft Outlook migration.

FACTORS INFLUENCING THE TEST EFFORT

Many things affect the level of effort required to fulfil the test-related aspects of a project to ensure that the objectives of the project, release or iteration are met. These can be split into four main categories, as shown below.

1. Product characteristics:
    - The risks associated with the product (defined during risk-based testing).
    - The quality of the test basis; for example requirements, user stories and so on.
    - The complexity of the product domain.
    - The number of quality characteristics such as reliability.
    - The number of non-functional requirements.
    - The security requirements (perhaps meeting ISO 27001, the security standard).
    - How much documentation is required (e.g. some legislation-driven changes demand a certain level of documentation that may be more than an organisation would normally produce).
    - Requirements for legal or regulatory compliance.
2. Development process characteristics:
    - The stability and maturity of the organisation; for example a very process mature organisation will take a lot less time to achieve what an immature (seat of their pants) organisation would take, as they are likely to make less mistakes.

Figure 5.3 A high-level test execution schedule



- The development model in use, such as Agile or sequential.
- The agreed test approach.
- The tools in use, automation, test management and so on.
- The test process defined in the test strategy and approach.
- Timescales.

3. People characteristics
   - The skills of those involved in the testing and development activity (the lower the skill level in development, the more defects could be introduced, and the lower the skill level in testing, the more detailed the test documentation needs to be).
   - Team cohesion and leadership.
4. Test results:
   - The number and severity of defects expected to be found.
   - The amount of rework needed.

Test estimation

There are many approaches to test estimation: two of the most used are metrics-based and expert-based. The two approaches are quite different, the former being based on data while the latter is a somewhat subjective approach.

The metrics-based approach

This approach relies on data collected from previous or similar projects. This kind of data might include:

- the number of test conditions;
- the number of test cases written;
- the number of test cases executed;

- the time taken to develop test cases;
- the time taken to run test cases;
- the number of defects found;
- the number of environment outages and how long on average each one lasted.

With this approach and the right data, it is possible to estimate quite accurately what the cost and time required for a similar project would be.

It is important that the actual costs and time for testing are accurately recorded. These can then be used to revalidate and possibly update the metrics for use on the next similar project.

The expert-based approach

This alternative approach to metrics is to use the experience of owners of the relevant tasks or experts to derive an estimate (this is also known as the Wide Band Delphi approach). In this context, 'experts' could be:

- business experts;
- test process consultants;
- developers;
- technical architects;
- analysts and designers;
- anyone with knowledge of the application to be tested or the tasks involved in the process.

There are many ways that this approach could be used. Here are two examples:

- Distribute a requirement specification to the task owners and get them to estimate their task in isolation. Amalgamate the individual estimates when received and build in any required contingency, to arrive at the estimate.
- Distribute a requirement specification to known experts who develop their individual view of the overall estimate and then meet together to agree on and/or debate the estimate that will go forward.

Expert estimating can use either of the above approaches individually or mix and match them as required.

Taking all of this into account, once the estimate is developed and agreed, the test manager can set about identifying the required resources and building the detailed plan.

CHECK OF UNDERSTANDING

1. Compare and contrast the two approaches to developing estimates.
2. Provide three examples of what a metrics approach to estimates would use as a base.

3. Name three areas that affect the level of effort to complete the test activity.

TEST MONITORING AND CONTROL

Having developed the test plan, the activities and timescales determined within the test execution schedule need to be constantly reviewed against what is actually happening. This is test monitoring. The purpose of test monitoring is to provide feedback and visibility of the progress of test activities.

The data required to monitor progress can be collected manually; for example, counting test cases developed at the end of each day, or, with the advent of sophisticated test management tools, it is also possible to collect the data as an automatic output from a tool either already formatted into a report, or as a data file that can be manipulated to present a picture of progress.

The progress data is also used to measure exit criteria such as test coverage; for example, 50 per cent requirements coverage achieved.

Having implemented test monitoring to understand progress through the test plan, test control is the corrective action undertaken for issues identified through test monitoring. Slippage of test activity dates or delays in delivery of external components are two potential issue areas. Test-control actions could include:

- reprioritising tests if, for example, software is delivered late (a potential risk);
- changing the test schedule;
- re-evaluating the entry/exit criteria;
- changing the scope of the test activity.

The following test-control activities are likely to be outside the test manager's responsibility. However, this should not stop the test manager making a recommendation to the project manager:

- descoping of functionality; that is, removing some less important planned deliverables from the initial delivered solution to reduce the time and effort required to achieve that solution;
- delaying release into the production environment until exit criteria have been met;
- continuing testing after delivery into the production environment so that defects are found before they occur in production.

Metrics used in testing

In any project, metrics can be collected at any time – either during or at the end of the project, in order to assess:

- progress against the plan, both in terms of activities and budget;

- current quality of the item under test (test object);
- adequacy of the test approach (will it enable all testing to be completed?);
- effectiveness of the test activities with respect to the test objectives.

Common test metrics in use include:

- percentage of planned work done in test case preparation (or percentage of planned test cases prepared);
- percentage of planned work done in test environment preparation;
- test case execution (e.g. number of test cases run/not run, and test cases passed/failed);
- defect information (e.g. defect density, defects found and fixed, failure rate and retest results);
- test coverage of requirements, risks or code;
- subjective confidence of testers in the product;
- task completion, resource allocation and usage, and effort;
- dates of test milestones;
- testing costs, including the cost compared with the benefit of finding the next defect or running the next test.

Ultimately, test metrics are used to track progress towards the completion of testing, which is determined by the exit criteria. So, test metrics should relate directly to the exit criteria.

There is a trend towards ‘dashboards’, which reflect all of the relevant metrics on a single screen or page, ensuring maximum impact. For a dashboard, and generally when delivering metrics, it is best to use a relatively small but impact-worthy subset of the various metric options available. This is because the readers do not want to wade through lots of data for the key item of information they are after, which invariably is ‘Are we on target to complete on time?’

These metrics are often displayed in graphical form, examples of which are shown in Figure 5.4. This reflects progress on the running of test cases and reports on defects found. There is also a box at the top left for some commentary on progress to be documented (this could simply be the issues and/or successes of the previous reporting period).

The graph in Figure 5.5 is the one shown at the bottom left of the dashboard in Figure 5.4. It reports the number of defects raised, and also shows the planned and actual numbers of defects.

Test reporting

Test reporting is the process whereby test metrics are reported in a summarised format both during and at the end of a test activity, to update the reader regarding the testing tasks undertaken. Test reports produced during the test activity are referred to as test progress reports, whereas a test report produced after a test activity has completed may be referred to as a test summary report.

The test manager regularly issues a test progress report during test monitoring and control for the project stakeholders such as the sponsor, project and programme managers and any product owners. When exit criteria have been met and a test activity completes, the test manager issues a test summary report. This report provides an overview of the test activity undertaken, using data derived from the test progress reports.

Figure 5.4 iTesting executive dashboard

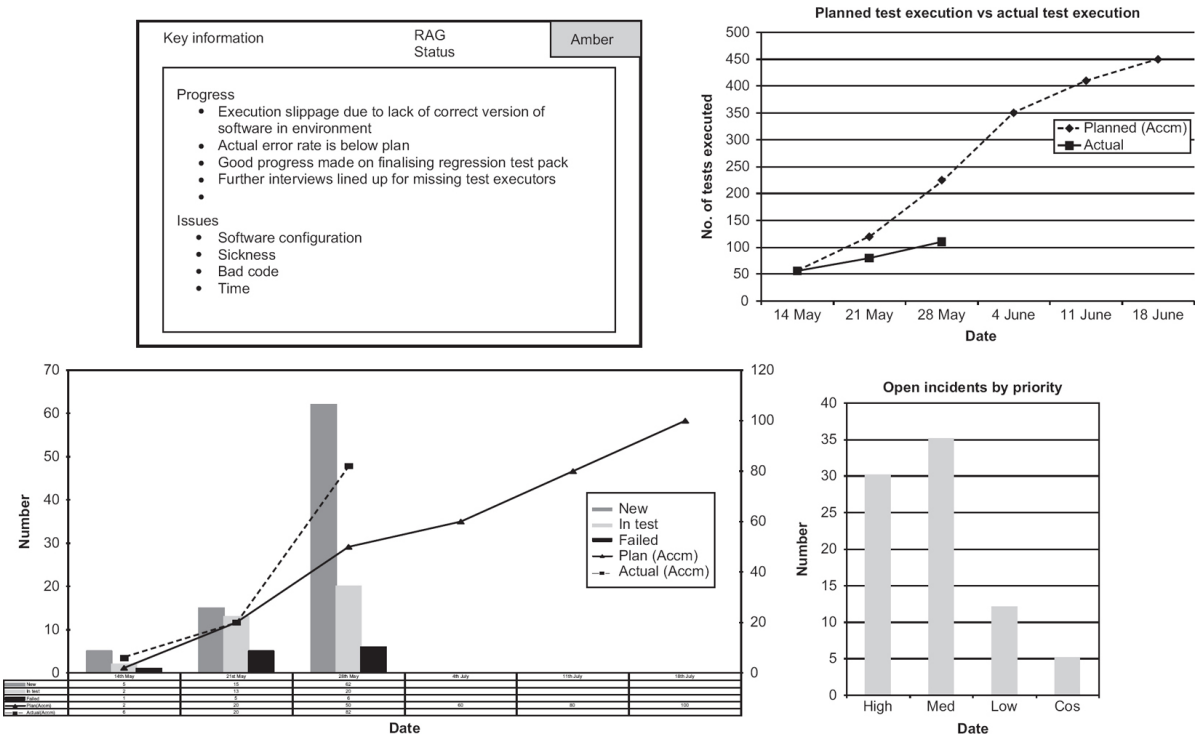
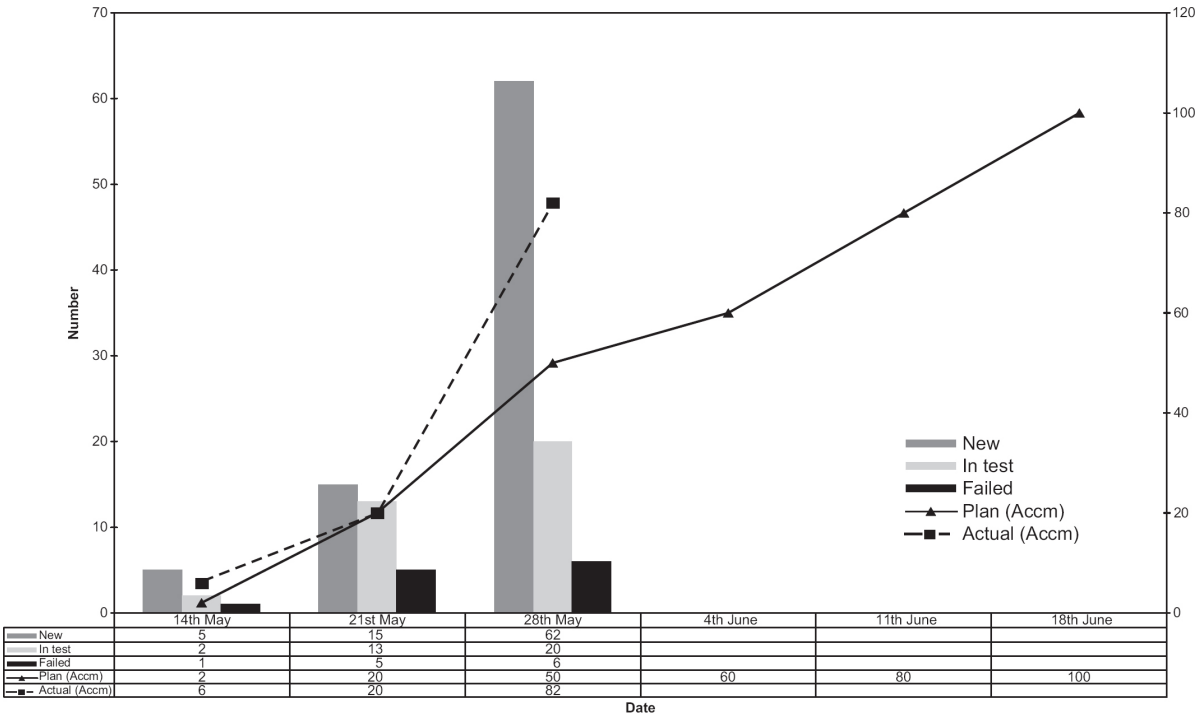


Figure 5.5 Incidents planned/raised



	14th May	21st May	28th May	4th June	11th June	18th June
New	5	15	62			
In test	2	13	20			
Failed	1	5	6			
Plan (Accm)	2	20	50	60	80	100
Actual (Accm)	6	20	82			

The following information may be included in both a test progress report and a test summary report:

- summary of testing performed;
- information on what occurred during a test period;
- any deviations from the plan, such as schedule changes;
- the status of testing and product quality, relating to either the exit criteria or the definition of ‘done’;
- blocking factors that have impacted the test schedule;
- metrics reflecting defects, test cases, test coverage, activity progress and resource consumption;
- residual risk;
- reusable test work products produced.

In addition to the above, test progress reports may also include:

- the status of test activities and progress against the test plan;
- factors impacting progress;
- testing planned for the next reporting period;
- the quality of the test object.

Key for any report is that it is focused on the information required based upon the report’s audience; some recipients may require graphs, while others wish to see the detailed data. It is the responsibility of the test manager to ensure that the recipient requirements for reports are understood before any test activity starts.

In an Agile project, the test progress reporting may be included in task boards, effect summaries and burn-down charts, which may also be discussed in the daily stand-up meeting, where the project team review progress during the previous period (often a day) and what is planned for the next period.

ISO 29119-3 documents required contents for both test progress reports (called test status report in the standard) and a test summary report (called a test completion report in the standard).

Tables 5.3a and 5.3b detail the two separate standard contents.

The information gathered can also be used to help with any process improvement opportunities. This information can be used to assess whether:

- the goals for testing were correctly set (where they achievable; if not why not?);
- the test approach or strategy was adequate (e.g. did it ensure there was enough coverage?);
- the testing was effective in ensuring that the objectives of testing were met.

Table 5.3a Test progress report outline

Section no.	Heading	Details
1	Overview	Identifies the document and describes the origins and history
2	Unique identification of the document	The specific unique identifier allocated to this document, e.g. TP 00001
3	Issuing organisation	Specifies who is responsible for completion and distribution of the document
4	Approval authority	Identifies who is responsible for reviewing and signing off the document before it is issued
5	Change history	A record of each version of the document and any changes that were included for each version
6	Introduction	Explanatory information about the content and structure of the document
7	Scope	Defines the areas of coverage included within the document, test activities etc.
8	References	Lists referenced documents and identifies repositories for system, software and test information. The references may be separated into 'external' references that are imposed from outside the organisation and 'internal' references that are imposed from within the organisation
9	Glossary	A glossary that defines the terms, abbreviations and acronyms, if any, used in the document
10	Test status	Includes: • Reporting period • Progress against the test plan • Factors blocking progress • Test measures • New and changed test risks • Testing planned in the next period

Table 5.3b Test summary report outline

Section no.	Heading	Details
1	Overview	Identifies the document and describes the origins and history
2	Unique identification of the document	The specific unique identifier allocated to this document, e.g. TSR 00001
3	Issuing organisation	Specifies who is responsible for completion and distribution of the document
4	Approval authority	Identifies who is responsible for reviewing and signing off the document before it is issued
5	Change history	A record of each version of the document and any changes that were included for each version
6	Introduction	Explanatory information about the content and structure of the document
7	Scope	Defines the areas of coverage included within the document, test activities, etc.
8	References	Lists referenced documents and identifies repositories for system, software and test information. The references may be separated into 'external' references that are imposed from outside the organisation and 'internal' references that are imposed from within the organisation
9	Glossary	A glossary that defines the terms, abbreviations and acronyms, if any, used in the document
10	Testing performed	Includes: • A summary of testing performed • Deviations from planned testing • Test completion evaluation, e.g. have exit criteria all been met, if not why not • Factors that blocked progress • The collated test measures • Residual risks • Test deliverables • Reusable test assets • Lessons learnt

CHECK OF UNDERSTANDING

1. Name four common test metrics.
2. Name the 10 headings in the ISO 29119-3 test summary report.
3. Identify three ways a test manager can control testing if there are more tests than there is time to complete them.

DEFECT MANAGEMENT

A defect is any unplanned event occurring that requires further investigation. In testing, this translates into anything where the actual result is different from the expected result. A defect when investigated may be a defect; however, it may also be a change to a specification or an issue with the test being run. It is important that a process exists to track all defects through to closure. This process has to be agreed by all parties involved and can be quite informal.

Defects can be raised at any time throughout the Software Development Life Cycle, from reviews of the test basis (requirements, specifications etc.), coding, static analysis, test specification and dynamic testing.

Typical defect reports have the following objectives:

- To provide developers and other parties with feedback on the problem to enable identification, isolation and correction as necessary. It must be remembered that most developers and other parties who will correct the defect or clear up any confusion will not be present at the point of identification, so without full and concise information they will be unable to understand the problem, and possibly therefore be unable to understand how to go about fixing it. The more information provided, the better.
- To provide test managers with a means of tracking the quality of the system under test and the progress of the testing. Key metrics used to measure progress is a view of how many defects are raised, their priority and finally that they have been corrected and signed off.
- To provide ideas for test process improvement. For each defect the point of injection should be documented, for example a defect in requirements or code, and subsequent process improvement can focus on that particular area to stop the same defect occurring again.

A defect report filed during dynamic testing typically includes:

- an identifier;
- a title or a short summary of the defect being raised;
- date of the defect report, issuing organisation and author;
- identification of the test item and environment being used;
- the development life cycle phase it was identified in;
- a description of the defect to enable reproduction and resolution;
- expected and actual results;
- scope or degree of impact (severity) of the defect on the stakeholders;
- urgency (priority) to fix;
- state of the defect report; for example is it open, deferred, closed and so on?

- conclusion, recommendations and approvals;
- change history (updates reflecting the sequence of action taken to resolve the defect);
- any references.

Defect management is the process of recognising, investigating, taking action and disposing of defects. It involves recording defects, classifying them and identifying the impact. The process of defect management ensures that defects are tracked from recognition to correction, and finally through retest and closure. It is important that organisations document their defect management process and ensure that they have appointed someone (often called a defect manager/coordinator) to manage/police the process.

Defects are raised on defect reports, either electronically via a defect management system (from Microsoft Excel to sophisticated defect management tools) or on paper.

The syllabus also recognises that ISO 29119-3 defines a test defect report (called a test defect report) which has sections aligned with those documented above.

CHECK OF UNDERSTANDING

1. Identify three details that are usually included in a defect report.
2. What is the name of the standard that includes an outline of a test defect report?
3. What is a test defect?

CONFIGURATION MANAGEMENT

The purpose of configuration management is to establish and maintain the integrity of the component or system, the testware and their relationships to one another throughout the project and product life cycle. It involves managing products, facilities and processes by managing the information about them, including changes, and ensuring that they are what they are supposed to be in every case.

For testing, configuration management will involve controlling both the versions of code to be tested and the documents used during the development process; for example, requirements, design and plans.

In both instances, configuration management should ensure that each test item is uniquely identified and provide full traceability throughout the test process; for example, a requirement should be traceable through to the test cases that are run to test its levels of quality and vice versa.

Effective configuration management is important for the test process as the contents of each release of software into a test environment must be understood and at the correct version,

otherwise testers could end up wasting time because either they are testing an invalid release of the software or the release does not integrate successfully, leading to the failure of many tests.

In most instances the project will have already established configuration management processes that will define the documents and code to be held under configuration management. If this is not the case, then during test planning the process and tools required to establish the right configuration management processes will need to be selected/implemented by the test manager.

The same principle applies to testware. Each item of testware (such as a test procedure) should have its own version number and be linked to the version of the software it was used to test. For example, test procedure TP123a might be used for software Release A and TP123b might be used for software Release B – even though both have the same purpose and even expected results. However, another test procedure, TP201, may be applicable to all releases.

A good configuration management system will ensure that the testers can identify exactly what code they are testing as well as have control over the test documentation such as test plans, test specification, defect logs and so on.

CHECK OF UNDERSTANDING

1. Define configuration management.
2. What can be stored under configuration management?
3. Why is it important to have effective configuration management?

SUMMARY

In this chapter we have looked at the component parts of test management. We initially explored risk and testing. When developing the test plan, the test manager and tester will look at the product risks (risks that relate directly to the failure of the product in the live environment) to decide what is important to test, as well as ensuring that any project risks (risks relating to the delivery of the project) are mitigated.

The importance of independence in the test organisation and how independence helps to ensure that the right focus is given to the test activity was reviewed. Independence is gained by separating the creative development activity from the test activity and we looked at the different levels of independence that are achievable:

- the developers – low independence;
- independent testers ceded to the development team;
- independent permanent test team, a centre of excellence within the organisation;
- independent testers or test team provided by the operational business unit;

- outsourced test team or the use of independent contractors – high independence.

We have looked at the test strategy and approach and how they shape the test activity based on many influences, including risks and the objectives of the testing.

We have reviewed two roles that exist within a test project: test manager and tester. Both roles are important to the delivery of testing, but could be vested in one or many people; for example, one person could have the role of test manager and tester. A test manager has responsibility for all of the planning activity, while the tester has responsibility for activities that surround the preparation of test cases.

ISO 29119-3 provides outlines of four test-planning documents:

- the test plan;
- the test progress report;
- the test summary report;
- the test defect report.

Test management depends not only on the preparation of the required documents but also on the development of the right entry and exit criteria and estimates, the monitoring of progress through the plan and the control activities implemented to ensure the plan is achieved.

Test estimating can be achieved in one of two ways: metrics or an expert-based approach.

After a plan of activity has been developed and time begins to pass, the test manager needs to monitor the progress of the activities. If any activity is delayed or there has been a change of any kind in the project itself, the test manager may need to revise the plan or take other actions to ensure that the project is delivered on time.

We explored how the defects found during testing are recorded, and we reviewed the level of detail that needs to be recorded to ensure that any defect is fully understood and that any fix then made is the right one.

Finally, we looked at configuration management. When running test cases against the code, it is important that the tester is aware of the version of code being tested and the version of the test being run. Controlling the versioning of the software and test assets is called configuration management. Lack of configuration management may lead to issues like loss of already-delivered functionality, reappearance of previously corrected errors and no understanding of which version of the test was run against which version of the code.

Example examination questions with answers

E1. K1 question

When assembling a test team to work on an enhancement to an existing system, which of the following has the highest level of test independence?

a. A business analyst who wrote the original requirements for the system.
b. A permanent programmer who reviewed some of the new code but who has not written any of it.
c. A permanent tester who found the most defects in the original system.
d. A contract tester who has never worked for the organisation before.

E2. K2 question
Which of the following correctly identify a metrics-based approach to estimation?

a. Groups of experts provide estimates based on their experience.
b. Volumes of defects identified at a given stage in a project.
c. Records of defects found in a similar stage in another project and the time taken to remove them.
d. Comparison of the estimates given by testers on the project and independent experts.

E3. K2 question
Which of the following is appropriate content for a test summary report?

i. The status of testing and progress against the test plan.
ii. Information about what occurred during a test period.
iii. A review of test activity progress and resource consumption for the system testing phase.
iv. An assessment of the quality of the test object at the present stage of testing.

a. i and ii.
b. ii and iii.
c. iii and iv.
d. i and iv.

E4. K1 question
Which of the following terms is used to describe the management of software components comprising an integrated system?

a. Configuration management.
b. Defect management.
c. Test monitoring.
d. Risk management.

E5. K1 question

A new system is about to be developed. Which of the following functions has the *highest* level of risk?

a. Likelihood of failure = 20%; impact value = £100,000.
b. Likelihood of failure = 10%; impact value = £150,000.
c. Likelihood of failure = 1%; impact value = £500,000.
d. Likelihood of failure = 2%; impact value = £200,000.

E6. K2 question
Which of the following statements about risks is most accurate?

a. Project risks rarely affect product risk.
b. Product risks rarely affect project risk.
c. A risk-based approach is more likely to be used to mitigate product rather than project risks.
d. A risk-based approach is more likely to be used to mitigate project rather than product risks.

Answers to questions in the chapter

SA1. The correct answer is c.
SA2. The correct answer is a.
SA3. The correct answer is a.

Answers to example examination questions

E1. The correct answer is d.
In this scenario, the contract tester who has never worked for the organisation before has the highest level of test independence. The three others are less independent because they are likely to make assumptions based on their previous knowledge of the requirements, code and general functionality of the original system.

Note that independence does not necessarily equate to most useful. In practice, most test or project managers would recruit a permanent tester who has worked on the original system in preference to a contract tester with no knowledge of the system. However, when assembling a team, it is useful to have staff with varying levels of test independence and system knowledge.

E2. The correct answer is c.

a. This approach is known as the Wide Band Delphi estimation technique and is based on multiple, well-informed estimates but not on data, as required by a metrics-based approach.

b. Volumes of defects identified is a valid metric and could be valuable in estimation, but without data about how much effort was required to remove them we would not be able to use the data about volumes in a metrics-based approach.

c. This is a valid metrics-based approach because data about the volume of defects and the time taken to remove them is provided, albeit on a different project.

d. This is potentially an effective way to improve the ability of testers to estimate accurately, but it is expert-based rather than metrics-based because there is no actual data to provide measures of achievement.

E3. The correct answer is b.
While all of the options describe content that summarises activity, option i is focused on identifying specific progress information, that is, whether or not progress is consistent with the plan. Item iv is also specific to progress towards an objective (the quality of the test object). Items ii and ii, in contrast, provide a broader view of what happened and what has been done, that is, they summarise rather than report specific progress. For these reasons, items ii and iii are more appropriate to a test summary report, while items i and iv are more appropriate to a test progress report.

E4. The correct answer is a.
Defect management is the collection and processing of defects raised when errors and defects are discovered. Test monitoring identifies the status of the testing activity on a continual basis. Risk management identifies, analyses and mitigates risks to the project and the product. Configuration management is concerned with the management of changes to software components and their associated documentation and testware.

E5. The correct answer is a.
In b, the product of probability × impact has the value £15,000; in c, the value is £5,000 and in d, it is £4,000. The value of £20,000 in a is therefore the highest.

E6. The correct answer is c.
In general, project risk and product risk can be hard to differentiate. Anything that impacts on the quality of the delivered system is likely to lead to delays or increased costs as the problem is tackled. Anything causing delays to the project is likely to threaten the delivered system's quality. The risk-based approach is an approach to managing product risk through testing, so it impacts most directly on product risk.