

---

## 3 Management Issues

*While management issues are, by definition, the realm of the test manager, there are still areas in this category that are of concern for the test analyst and the technical test analyst. We need to understand the types of systems we are dealing with and how they might affect our testing approach. We also need to understand the overall process and what our contribution will be at each step. In addition, a good understanding of risk-based testing and risk management in terms of project and product risk is an asset for any test analyst.*

### 3.1 Types of Systems

The types of systems we may need to test are many and varied. Each represents different levels of risk that may lead to particular testing strategies being proposed. In a book on test analysis, a full coverage of specific types of systems and their architectures would be inappropriate. However, certain specific types of systems are described in the following sections because they have significant and direct influence on the software quality characteristics to be addressed in testing strategies. We'll consider the following system types:

- Systems of systems
- Safety-critical systems
- Real-time and embedded systems

*Testing strategies are influenced by the type of system under test*

#### 3.1.1 Systems of Systems

Today, we are frequently involved in testing systems of systems. As you will see from the summary points outlined later, the very nature of such systems represents a particular challenge for all those with testing responsibilities.

The architecture that makes up a system of systems features several individual components that themselves may be considered systems. They cooperate to provide benefit to a particular stakeholder (e.g., business owner). The components of the overall system of systems typically consist of various software applications or services, communications infrastructure, and hardware devices. These may themselves be driven by software applications.

*The Marathon example is a system of systems*

Systems of systems are developed using a “building block” concept. Individual component systems are integrated with each other so that entire systems can be created without having to develop applications from new. A system of systems frequently makes use of reusable software components, third-party applications, commercial off-the-shelf (COTS) software, and distributed business objects.

On the upside, this concept may result in cost reductions for the development organization but there is a downside when we consider the cost of testing, which may increase substantially. Why is this?

- High levels of complexity

Complexity is inherent in systems of systems. This arises from a number of sources, including system architectures employed, the different software lifecycle development models that may be used for individual application development efforts, and complex compatibility issues of both a technical and functional nature (i.e., do the building blocks actually fit together?) Testing professionals know that complexity is a major driver of product risk; where we have high levels of complexity we generally expect there to be more defects in the product, both from a functional (domain) and a non-functional (technical) perspective.

- The time and effort needed to localize defects

Within a system of systems, the localization of defects can be a technical and organizational challenge. It may take a long time and considerable effort to localize defects since the testing organization typically does not have complete access to all system components. As a result, it may simply not be possible to perform detailed analysis or set up monitors where we would like to.

- More integration testing may be required

Whereas the development of an individual system normally calls for an integration testing stage, with systems of systems we have an additional “layer” of integration testing to perform at the intersystem level. This

*System integration tests play a critical role*

testing level, which is often called system integration testing, may require the construction of simulators to compensate for the absence of particular component systems.

■ Higher management overhead

More effort often results from managing the testing among the many organizational entities involved in developing a system of systems. These could include various product suppliers, service providers, and many supplier companies that are perhaps not even involved in the project directly. This may give rise to a lack of a coherent management structure which makes it difficult to establish ownership and responsibilities for testing. Test analysts need to be aware of this when designing particular tests such as “end-to-end” tests of business processes. For example, when a user initiates a transaction, the technical and organizational responsibility for handling that transaction may change several times and may be completed on systems that are totally outside the control of the originating organization.

■ Lack of overall control.

Because we may not always have control over all system components, it is common for software simulations to be constructed for particular component systems so that system integration testing can be performed with some certainty. For the same reasons, the test manager will also need to establish well-defined supporting processes such as release management so that the software can be delivered to the testing team from external sources in a controlled manner. Test analysts will need to work within the framework of these supporting processes so that, for example, tests are developed to defined releases and baselines.

*Who's in charge here?*

Many of the characteristics exhibited by a system of systems are present in the Marathon example:

- Individual components like the customer relations management system can be considered systems in their own right.
- System components used consist of various software applications (e.g., invoice system) and software-driven hardware devices (e.g., run unit).
- Two of the applications (the customer relations system and invoicing system) are commercial off-the-shelf applications that may not have been used together in a system of systems like Marathon before. This highlights the need for system integration testing.

### 3.1.2 Safety-Critical Systems

A safety-critical system is one that may endanger life or lead to other severe losses in the event of failure. Normally the criticality of a project is estimated as part of the project's feasibility study, or as a result of initial risk management activities. The test analyst and technical test analyst must be aware of how the project's criticality has been assessed and, in particular, whether the term safety-critical applies.

*Safety-critical systems require more rigorous testing*

The strategies we apply to testing safety-critical systems are generally comparable to those discussed throughout this book. For safety-critical systems though, it is the higher level of rigor with which we need to perform test tasks and which shape our testing strategies. Some of those tasks and strategies are listed here:

- Performing explicit safety analysis as part of the risk management. Failure Modes and Effects Analysis (FMEA) can support this task (refer to section 3.10 in the Advanced syllabus for more details).
- Performing testing according to a predefined software development lifecycle model, such as the V-model.
- Conducting failover and recovery tests to ensure that software architectures are correctly designed and implemented.
- Performing reliability testing to demonstrate low failure rates and high levels of availability.
- Taking measures to ensure that safety and security requirements are fully implemented.
- Showing that faults are correctly handled.
- Demonstrating that specific levels of test coverage have been achieved.
- Creating full test documentation with complete traceability between requirements and test cases.
- Retaining test data, results, or test environments (possibly for formal auditing).

*Industry standards often apply to safety-critical systems*

Often these issues are covered by standards, that may be specific to particular industries, as in the following examples:

- Space industry:  
The European Cooperation on Space Standardization (ECSS) [URL: ECSS] recommends methods and techniques depending on the criticality of the software.
- Food and drug industry  
The US Food and Drug Administration (FDA) recommends certain

structural and functional test techniques for medical systems subject to Title 21 CFR Part 820.

- Aircraft industry

The international Joint Aviation Authorities (JAA) defined the levels and type of structural coverage to be demonstrated for avionics software, depending on a defined level of software criticality.

The test manager will convey the level of safety criticality of the system and software under test and whether particular standards need to be applied. We must ensure that the tests we design comply to any such standards and that we can support the test manager by demonstrating compliance not only within the testing project but also possibly to external auditors.

### 3.1.3 Real-Time and Embedded Systems

In real-time systems, there are usually particular components present whose execution times are critical to the correct functioning of the system. These may be responsible, for example, for calculating data at high update rates (e.g., 50 times per second), responding to specific events within a minimum time period, or monitoring processes.

Software that needs to function in real-time is often “embedded” within a hardware environment. This is the case with many everyday consumer items such as mobile phones and also in safety-critical systems such as aircraft avionics.

*Embedded systems are all around us*

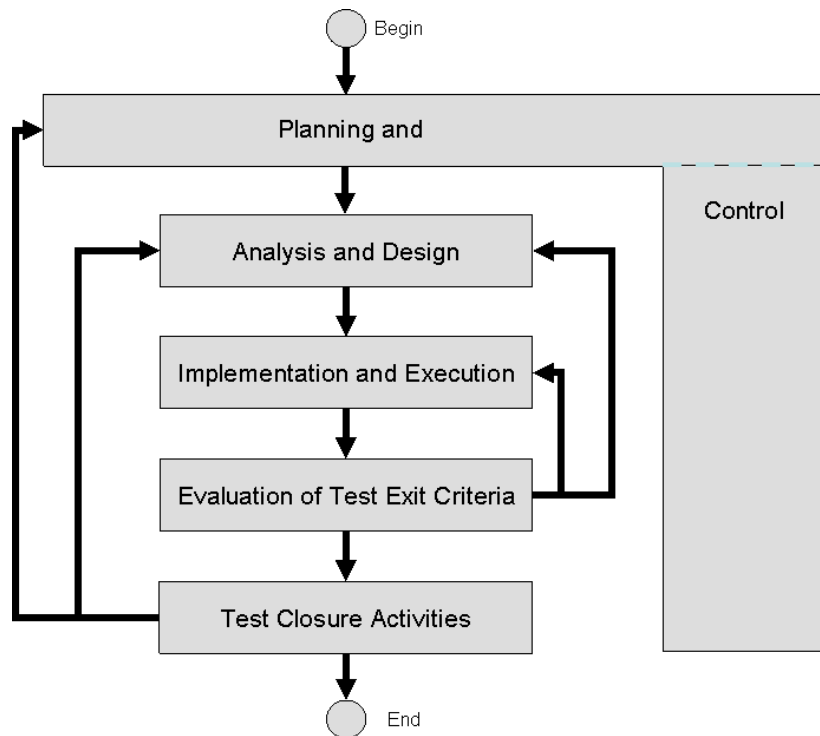
Real-time and embedded systems are particularly challenging for the technical test analyst. For example:

- We may need to apply specific testing techniques to detect, for example, “race” conditions.
- We will need to specify and perform dynamic analysis with tools (see section 8.2, “Dynamic Analysis”).
- A testing infrastructure must be provided that allows embedded software to be executed and results obtained.
- Simulators and emulators may need to be developed and tested to be used during testing (see section 18.3.3 for details).

## 3.2 Test Process

Test processes may consist of many individual activities, but there are five steps into which these activities generally fit. These five steps to the generic test process are shown in the following diagram:

**Figure 3-1**  
The fundamental test process



*Test analysts and technical test analysts are a critical part of a successful test process implementation*

Managing the test process is the job of the test manager. So why include this diagram? Because the test analysts and technical test analysts are critical to implementing the test process and a general understanding of the process is required if we are to do our job the right way at the right time and supply the right documentation. The process will break down if steps are skipped and we will soon find ourselves reacting rather than approaching a project with a planned effort.

Let's look at each of the steps in the test process more closely.

### 3.2.1 Test Planning and Test Control

At the test planning stage, the test manager is determining the testing approach, planning the resources, setting the strategy, creating the test schedule, and determining the metrics that will be needed for adequate control and monitoring of the project. In terms of resources, at this point we are considering equipment, software and people resources that will be needed to accomplish the testing. The test manager is looking at training needs and hiring requirements in order to be sure the staff is in place when the project begins. Most of this information is documented in the test plan document. The IEEE 829 Test Plan Specification provides the following outline for the test plan document:

1. Test plan identifier
2. Test items (including version/revision and documentation such as requirements)Introduction
3. Features to be tested
4. Features not be to tested
5. Approach (activities, techniques, tools)
6. Item pass/fail criteria
7. Suspension criteria and resumption requirements
8. Test deliverables
9. Testing tasks
10. Environmental needs (facilities, hardware, software, network, supplies, level of security, special tools)
11. Responsibilities
12. Staffing and training needs
13. Schedule (test milestones and item transmittal events)
14. Risks and contingencies
15. Approvals

Notice that there is a section to identify risks and mitigation plans for those risks. When we are looking to control a project, we are hoping to control the risks. When a risk is identified, we have to deal with it by creating a mitigation plan, transferring the risk elsewhere, or deciding to ignore it. In the case where we can identify risks at the beginning of a project, we are better able to make plans to deal effectively with those risk items should they occur.

Risk management is usually the responsibility of the test manager. Test analysts and technical test analysts must contribute information for

*If a project isn't planned, it can't be controlled*

*Caution! Everything may be assumed to be in scope unless you state otherwise*

risk identification and possible mitigation plans as part of the planning process. When we are looking at the risk of the project, we need to consider the two risk types: project risk and product risk. Project risks are sometimes called planning risks and are oriented toward anything that could cause the overall project to fail to meet its objectives. Project risks include such things as personnel issues (vacations, training, availability), vendor or third party issues, or delivery schedule issues. Product risks are the risks within the product itself, such as unfound defects. Testing and following good quality practices are ways we mitigate product risk.

Before we go any further into the test process, let's look at figure 3-2 to see how all the IEEE 829 Standard Test Documents fit together.

### 3.2.2 Test Analysis and Test Design

In the test analysis and design step, we are considering the details of the testing project. This is where we are figuring out what to test, how much effort to expend, what types of testing we should do and any tools that will be required for this effort. For example, after reviewing the requirements documents, we may determine that performance testing is warranted. In that case, we need to be defining performance test cases, purchasing performance testing tools and procuring the resources we will need to do this testing.

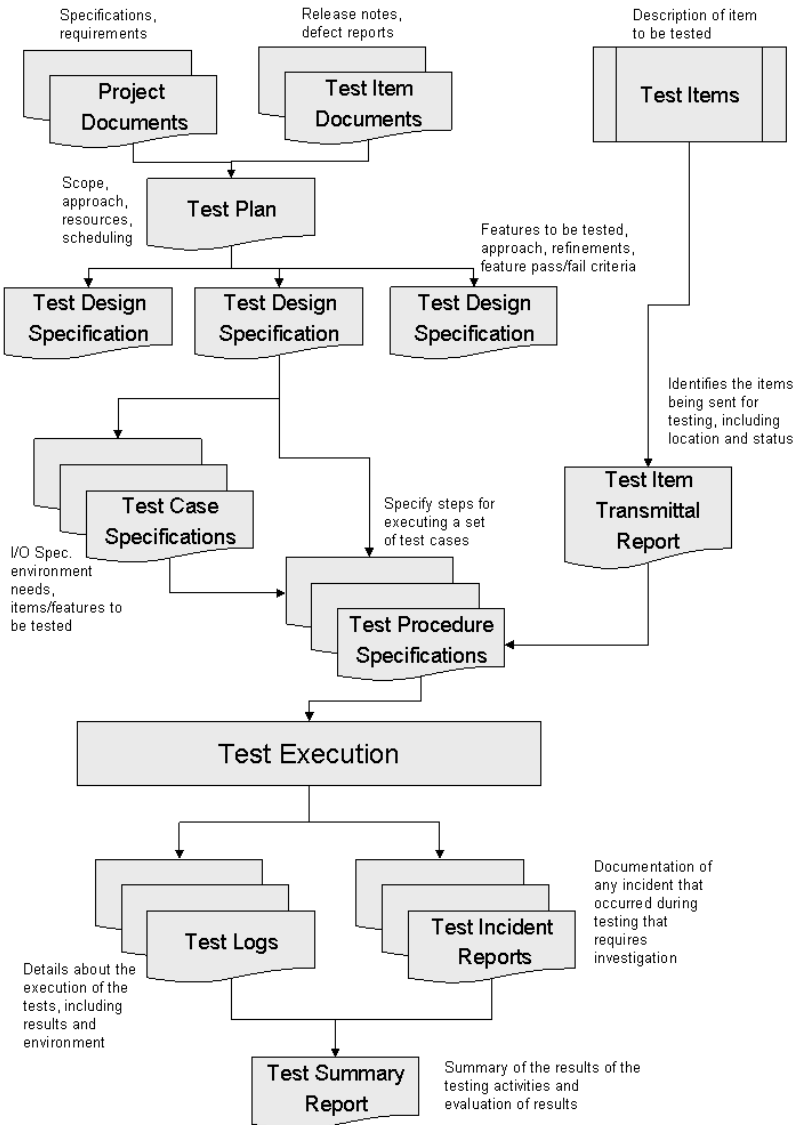
*Start static testing as soon as you have something to read*

As we review the test basis, the documents from which we are determining what to test, we are doing static testing. We are examining each document, and we should be documenting any problems we find both to ensure resolution and to use for future process improvement initiatives (reviews are discussed in depth in chapter 17). During these reviews, we also gather the data needed to write the test specification. This document, cleverly named, specifies what we are going to test. The IEEE 829 test design specification document consists of the following information:

1. Test design specification identifier
2. Features to be tested (test items, features, and combinations of features that will be covered by this spec)
3. Approach refinements (refinements from the test plan)
4. Test identification (list of the test cases associated with this design)
5. Feature pass/fail criteria

*Is it wrong, or just practical, to plan to run out of time?*

As we work on building the test design specification, we should also be building the risk analysis that will be used to guide a risk-based testing



**Figure 3-2**  
IEEE-829 document tree

effort. This risk analysis defines the testable items of the project, sometimes categorized by the ISO 9126 quality characteristics, and is used to record the business risk (how important is the correct operation of this item to the business) and the technical risk (how likely is it to fail) for each item. By recording this information now, in the analysis phase, we can

organize our actual test execution so that the highest risk items are addressed with the appropriate amount of testing. Risk-based testing is used to deal with situations in which there is not enough time to test everything (which, in our experience, is most of the time) by prioritizing our testing and test development. This gives us a strategy that allows us to use the time we have in effective risk mitigation.

A good risk analysis is a result of the contribution of the project stakeholders. The business interests have to be represented by someone who truly understands the business and the customer concerns. The technical risk can be determined only by the people who know what is likely to fail—the developers who know that some parts of the code are excessively complicated, the testers who know that some pieces of functionality will be extremely difficult to test well. It's the test manager's responsibility to coordinate and see to the creation of the quality risk analysis. It's the test analyst's job to ensure that the testing concerns are well represented.

### 3.2.3 Test Implementation and Test Execution

*Our old familiar friend,  
the test case*

Now that we have our test design specification, it's time to create our test cases and test procedures. Logical test cases are the high-level test descriptions that do not define the data that is to be used for the tests. Concrete test cases include the actual data that is to be used during testing. Since many test cases refer to spreadsheets that contain the actual data, they become concrete only when the instructions are actually joined with the data.

The IEEE 829 test case specification is commonly used in industry. That specification includes the following information:

1. Test case specification identifier
2. Test items
3. Input specifications
4. Output specifications
5. Environmental needs (hardware, software, and other environmental needs for this case)
6. Special procedural requirements (special constraints on the test procedures that will apply to this case)
7. Intercase dependencies

Notice that we are specifying the output from our test case. This means we have to know what the outcome should be based on the test condition (the

item or event that we are testing) in the test case. The expected outcome is usually determined by some form of an oracle that tells us what the software should do. This oracle should be defined in the test basis in order for us to define accurate test cases.

The IEEE 829 Test Procedure is the document that explains how to run a particular test case or a set of test cases. This procedure is often, in practice, actually included in the test case itself. Officially though, it should be considered a separate document containing the following information:

1. Test procedure specification identifier
2. Purpose (including list of applicable test cases)
3. Special requirements
4. Procedure steps (log, setup, start, proceed, measure, shutdown, restart, stop, wrap-up, contingencies)

When test cases are being designed, consideration should be given to test execution automation potential. If automation is a viable option, the automation coding should be underway at this stage of the project, creating automation scripts (which are actually self-contained test procedures often called test scripts). If we wait until the end to actually create the automation code, we won't get the benefit of using it for this version of this project. That may be acceptable if the project is to live a long time and repeated regression testing will allow the use of automation. Whether we will develop it now or later, we need to be thinking about automation right now (please refer to chapter 18 for more on automation issues).

So what about test execution? Will those tests run themselves? Probably not (unless you have some really amazing automation code!). This is the time for the manual and automated (if available) execution of the test cases we have so carefully designed. But wait, before we start running the test cases, it would probably be good to actually have something to test. The IEEE 829 Test Item Transmittal Report provides the information we need to locate and install the software we will be testing. It consists of the following information:

1. Transmittal report identifier
2. Transmitted items (including version, revision and person responsible)
3. Location
4. Status
5. Approvals

Now we know what we are testing, we know how to install it, we know what changes/fixes we are receiving, and we are ready to execute our test cases. As we run the test cases, we want to be sure we are recording the execution information in the IEEE 829 Test Log. This document is used to record particular information for each test case execution. This information tracking capability may be built into your test management system, but you will want to track the same information as that specified in the test log.

1. Test log identifier
2. Description (including items being tested and environments used)
3. Activity and event entries (dates and times, execution description, procedure results, environmental information, anomalous events, incident report identifiers)

*IEEE 829 has you covered!*

BS-7925-2 is another source for the type of data that should be logged. But what if we find a defect? IEEE 829 has us covered there too. We can fill out an IEEE 829 Test Incident Report that contains the following information:

1. Test incident report identifier
2. Summary
3. Incident description (inputs, expected results, actual results, anomalies, date and time, procedure step, environment, attempts to repeat, testers, observers)
4. Impact

Of course, your defect tracking system may track different information, or you may choose to follow the IEEE 1044 defect specification (discussed further in chapter 19).

The work at this stage tends to be iterative. As we receive new releases of the software, we will rerun test cases. This is usually the longest step in the generic test process, time-wise, because this is where the actual testing occurs.

### **3.2.4 Evaluation of Exit Criteria and Reporting**

Our last IEEE 829 test standard document appears in this step of the process—the IEEE 829 Test Summary Report. The test summary report can be prepared periodically as a type of test progress report at the conclusion of a level of testing (after integration testing, for example), but it is commonly done at the conclusion of the testing effort. The test summary document includes the following information:

1. Test summary report identifier
2. Summary (evaluation of test items)
3. Variances
4. Comprehensiveness assessment
5. Summary of results
6. Evaluation (per test item)
7. Summary of activities
8. Approvals

How do we know when we are finished testing? We determine test completion based on meeting the exit criteria we defined in the planning phase. This is often considered the Ship/No Ship decision point, where the information found in testing is returned to the project team for the release decision.

*You're not finished until the test closure activities complete*

### 3.2.5 Test Closure Activities

The test closure activities occur after the release has been shipped out. Now is the time to do any wrap-up reporting, document and archive the test environments, archive the test documents and data, and generally clear the decks for the next project. These test closure activities are often under-budgeted and receive inadequate attention because the next project is already waiting. It is important for the test manager to hold firm on these activities and to ensure that they are done correctly. Only in this way will we be able to return quickly to this project for maintenance releases or patches. Inadequate time spent on the test closure activities also reduces the effort we can spend looking for ways to improve our processes.

*If we don't learn from our mistakes, we are very likely to do the same things wrong in the next project.*

## 3.3 Learning Check

The following checklists will help you judge the knowledge you have gained from this chapter.

### Terms Used

BS-7925-2, exit criteria, FMEA, IEEE 829, product risk, project risk, risk analysis, risk-based testing, risk identification, risk management, risk mitigation, risk type, safety- critical system, software lifecycle, system of systems, test case, test closure, test condition, test design, test execution, test implementation, test item transmittal report, test log, test plan, test

procedure, test progress report, test schedule, test script, test specification, test summary report

### **Test Analyst and Technical Test Analyst**

- Recall the essential features of systems of systems
- Explain the factors influencing the testing of systems of systems
- Recall the essential features of safety-critical systems
- Recall the testing tasks that are particularly significant for testing safety-critical systems and give examples of industry-specific standards
- Recall the essential features of real-time and embedded systems
- Explain the criteria that influence the level of test condition development
- Explain and provide examples of test oracles and how they can be used in test documentation
- Describe the conditions that must be in place prior to test execution, including the testware, configuration management system, defect tracking system, and test environment
- Determine if the test completion criteria have been fulfilled based on a set of measures

### **Test Analyst (Specific)**

- Analyze and breakdown a requirement specification into a test specification based on IEEE 829, focusing on functional and domain test cases and test procedures
- Explain the stages in the software lifecycle at which functional testing is appropriate
- Understand how test analysis and design are static testing techniques that can be used to discover defects
- Prioritize test case creation and execution based on risk and be able to document this appropriately in the test documentation
- Outline the activities of risk-based testing for domain testing

### **Technical Test Analyst (Specific)**

- Explain the stages in the software lifecycle where non-functional tests and structure-based tests may be effectively applied
- Outline the activities of risk-based testing for technical testing