Chapter 5: Test Management

"... Project management throughout the development and implementation process was inadequate and at times ambiguous. A major systems integration project such as CAD

Requires full time, professional, experienced project management. This was lacking...”

"... The early decision to achieve full CAD implementation in one phase was misguided. In an implementation as far reaching as CAD it would have been preferable to implement in a step wise approach, proving each phase totally

before moving on to the next...”

Extract from the main conclusions of the official report into the failure of the London Ambulance Service's Computer Systems on October 26th and 27th 1992.

5.1 Overview

This module covers the overall management of the test effort for a particular project and attempts to answer several key questions such as:

How many testers do we need?

How shall the testers be organized?

What's the reporting structure and who is in charge?

How will we estimate the amount of testing effort required for this project?

How do we keep versions of our test material in line with the development deliverables?

How do we ensure the test effort remains on track?

How do we know that we have finished testing?

What is the process for logging and tracking incidents?

5.2 Objectives

After completing this module you will:

Understand how testing might be organized.

Understand the different roles in a test team.

Understand the importance of test estimation, monitoring and control.

Appreciate the need for configuration management of the test assets.

Understand how and why incidents must be logged and tracked.

5.3 Organization

"We trained hard ... but it seemed that every time we were beginning to form up into teams we would be reorganized... I was to learn later in life that we meet any new situation by reorganizing and a wonderful method it can be for creating the illusion of progress while producing confusion, inefficiency, -and demoralization."

A fundamental feature of our lives today is that nothing stays the same. Over time both internal and external pressures on the organizational structures (that we carefully put in place) must change and adapt if our business is to remain competitive. As development and testing organizations grow and evolve, a different structure is required to cope with the changing demands placed upon them. The approach adopted over time may look something like this:

Testing may be each individual developer's responsibility.
Testing is the development team's collective responsibility (either through buddy testing or assigning one person on the team to be the tester).
There is a dedicated independent test team (who do no development).
Internal test consultants 'centers of excellence' provide advice to projects.
A separate company does the testing - this is known as outsourcing.

An excellent description of how the test function can be organized within a company can be found in Ed Kit's book, Software Testing in The Real World [KIT95l

5.5 Configuration management (CM)

We all appreciate the need for testing and assuring quality in our development systems. But how many of us appreciate that Configuration Management is a precursor to these goals?

Configuration Management provides us with the balance to ensure that:

Systems are complete.
Systems are predictable in content.
Testing required is identified.
Change can be ring-fenced as complete.
An audit trail exits.

We've always practiced CM. Such activities an aggregating and releasing software to the production environment may, in the past, have been uncontrolled - but we did it.

Many famous organizations have found the need to develop a standard for CM that they have then since taken into the market place.

Configuration management encompasses much more that simply keeping a version control of your software and test assets, although that is a very good start. Configuration management is crucial to successful testing, especially regression testing because, in order to make repeatable, you must be able to recreate exactly the software and hardware environment that was used in the first instance.

Typical symptoms of poor CM might include:

Unable to match source and object code.
Unable to identify which version of a compiler generated the object code.
Unable to identify the source code changes made in a particular version of the software simultaneous changes are mad e to the same source code by multiple developers (and changes lost).

5.6 Definitions

ISO (International Standards Organization) definition of CM:

Configuration management (CM) provides a method to identify, build, move, control and recover any baseline in any part of the life cycle, and ensures that is secure and free from external corruption.

Configuration identification requires that all configuration items (CI) and their versions in test system are known.

Configuration control is maintenance of CI’s in a library and maintenance of records on how CI’s change over time.

Status accounting is the function of recording and tracking problem reports, change requests, etc.

Configuration auditing is the function to check on the contents of libraries, etc. for standards compliance, for instance.

CM can be very complicated in environments where mixed hardware and software platforms are being used, but sophisticated cross platform CM tools are increasingly available.

5. 7 Simple CM life cycle process

CM contains a large number of components. Each component has its own process and contributes to the overall process.

Let's take a look at a simple process that raises a change, manages it through the life cycle and finally executes an implementation to the production environment. Here we can see how to CM life cycle operates by equating actions with aspects of CM.

As a precursor activity we 'Evaluate Change'. All changes are evaluated before they enter the CM Life Cycle:

1.Raise Change Packet,
Uses Change Management functions to identify and register a change. Uses Change Control functions to determine that action is valid and authorized.

2.Add Configurable Item to Change Packet.

Select and assign configurable items to the Change packet. Execute Impact Analysis to determinate the items that also require some action as a result of the change and the order in which the actions take place.

3.Check-In.
Apply version CI back under CM control.

4.Create Executable.
Build an executable for every contained CI in the order indicated.

5.Sign-Off. .
Uses Change Control to verify that the auctioneer signaling that the auctioneer signaling that testing is complete for the environment in which the change is contained is authorized to do so, and that the action is valid.

6.Check-OK to Propagate.
Uses Change Control, Co Requisite to verify request to move a Change Packet through the life cycle is valid.

a) All precursor tacks completed successfully.
b) Next life cycle environment fit to receive.
c) Subsequent change in current environment has not invalidated Change Packet.

7. Propagation
Affect next environment population by releasing Change Packet and then distributing over a wider-area.

Note: You might notice that it is composed of a number of singular functions and series executed as a cycle. Of particular note is that 'Create Executable' is a singular function. This is because we should only ever build once if at all possible. This, primarily, saves time and computer resources. However, re-building an element in a new environment may negate testing carried out in proceeding one and can lead to a lengthy problem investigation phase.

5.8 What does CM control?

CM should control everything element that is a part of a system application.

Nominally, CM:

1. Configurable Items:
Maintains registration and position of all of our CI's. These may be grouped into logically complete change packets as a part of a development of maintenance exercise.

2. Defines Development Life Cycle:
It is composed of a series of transition points, each having its own Entry/Exit criteria and which maps to a specific test execution stage.

3. Movement:
Controls and Change Packet move and progresses it through the Life Cycle.

4. Environment:
Testing takes place in a physical environment configured specifically for the stage testing that the life cycle transition point and stage reflects.

5.9 How is it controlled?

CM is like every other project in that it requires a plan. It is particularly important that CM has a plan of what it is to provide as it forms a framework for life cycle management in which to work consistently and securely.

The CM plan cover what needs to be done, not by when, and defines three major areas:

The Processes will define or include:
Raising a change
Adding elements
Booking elements in and out Exit/Entry criteria
Life cycle definition
Life cycle progression
Impact analysis,
Ring-fencing change, release aggregation
Change controls.
Naming standards
Genetic processes for build and other activities

The Roles & responsibilities covering who and what can be done:
Configuration Manager & Librarian Project Manager, Operations Personnel Users, Developers and others as necessary

Records, providing necessary audit trail will include:
What is managed, the status of the life cycle position arid change status,
Who did what, where, when and under what authority. Also the success factor for activities.
Only once the CM plan and the processes that support it are defined can we consider automation.

5.10 What does CM look like?

CM has several hubs and functions that will make or break it. Hubs of system are defined as areas where information and source code are stored. Typically major hubs are central inventory and central repository. Surrounding those are four major tool sets that allow us to work on the data:

Version Management
Allows us access to any version or revision of a stored element.

Configuration Control Allows us to group elements into manageable sets.

Change Control & Management
This is global name given to processes that govern change through application development life cycle and stages it passes through from an idea through to implementation. It may include:

Change Control Panel or Board to assess and evaluate change;
Controls: Governing who can do what, when and under what circumstances.
Management: Carrying out an action or movement through the life cycle once
the controls have been satisfied

Build & Release
Control is how our elements are built and manner in which our change is propagated through life cycle.

The view is about as close to genetic global view of CM as you can get. It won't match all tools 100% as it covers all aspects of CM - and very few of the tools (although they might claim to) can do this.

Exercise

Configuration management -1

Make list of items that you think Test Manager should insist placed under configuration management control.

Exercise

Configuration management - 2

There are very many points to consider when implementing CM. We have summarized them into the following three categories:

CM Processes
The framework that specifies how our CM system is to operate and what it is to encompass.

Roles & Responsibilities
Who does what and at what time?

CM Records
The type of records we keep and the manner in which we keep and maintain them.

Quite a short list you might say. Using the information we have learned so far, try and construct a minimal Configuration Management Plan. Do not try and expand the processes required, but give hem suitable titles in an appropriate sequence.
Additionally, for every process you identify, try and match it to one or more segments of the CM Bubble diagram.

5.11 Test estimation, monitoring and control

Test estimation

Effort required to perform activities specified in high-level test plan must be calculated in advance. You must remember to allocate time for designing and writing the test scripts as well as estimating the test execution time. If you are going to use the test automation, there will be a steep learning curve for new people and you must allow for this as well. If your tests are going to run on multiple test environments add in extra time here too. Finally, you will never expect to complete all of the testing in one cycle, as there will be faults to fix and test will have to be re-run. Decide on how many test cycles you will require and try and estimate the amount of rework (fault fixing and re-testing time).

Test monitoring

Many test efforts fail despite wonderful plans. One of the reasons might be that the test team was so engrossed in detailed testing effort (working long hours, finding many faults) that they did not have time to monitor progress. This however is vitally important if the project is to remain on track. (e.g. use a weekly status report).

Exercise

Try and list what you might think are useful measures for tracking test progress.

Test Manager will have specified some exit (or completion) criteria in the master test plan and will use the monitoring mechanism to help judge when the test effort should be concluded. The test manager may have to report on deviations from the project/test plans such as running out of time before completion criteria have been achieved.

Test control - in order to achieve the necessary test completion criteria it may be necessary to re-allocate resources, change the test schedule, increase or reduce test environments, employ more testers, etc.

5.12 Incident Management

An incident is any significant, unplanned event that occurs during testing that requires subsequent investigation and/or correction. Incidents are raised when expected and actual test results differ.

5.12.1 what is an incident?

You may now be thinking that incidents are simply another name for faults but his is not the case. We cannot determine at the time an incident has occurred whether there is really a fault in the software, whether environment was perhaps set up incorrectly or whether in fact test script was incorrect. Therefore we log incident and move on to the next test activity.

5.12.2 Incidents and the test process

An incident occurs whenever an error, query or problem arises during the test process. There must be procedures in place to ensure accurate capture of all incidents. Incident recording begins as soon as testing is introduced into system's development life cycle. First incidents that will be raised therefore are against documentation as project proceeds; incidents will be raised against database designs, and eventually program code of system under test.

5.12.3 Incident logging

Incidents should be logged when someone other than author of product under test performs testing. When describing incident, diplomacy is required to avoid unnecessary conflicts between different teams involved in testing process (e.g. developers and testers). Typically, information logged on an incident will include:

- Name of tester(s), data/time of incident, Software under test ID
- Expected and actual results
- Any error messages
- Test environment
- Summary description
- Detailed description including anything deemed relevant to reproducing/fixing potential fault (and continuing with work)
- Scope
- Test case reference
- Severity (e.g. showstopper, unacceptable, survivable, trivial)
- Priority (e.g. fix immediately, fix by release date, fix in next release)
- Classification. Status (e.g. opened, fixed, inspected, retested, closed)
- Resolution code (what was done to fix fault)

Incidents must be graded to identify severity of incidents and improve quality of reporting information. Many companies use simple approach such as numeric scale of I to 4 or high, medium and low. Beizer has devised a list and weighting for faults as follows:

	Mild	Poor alignment, spelling etc.
	Moderate	Misleading information, redundant information
	Annoying	Bills for 0.00, truncation of name fields etc.
	Disturbing	Legitimate actions refused, sometimes it works, sometimes not
	Serious	Loss of important material, system loses track of data, records etc.
	Very serious	The mis-posting of transactions
	Extreme	Frequent and widespread mis-postings
	Intolerable	Long term errors from which it is difficult or impossible to recover
	Catastrophic	Total system failure or out of control actions
	Infectious	Other systems are being brought down

In practice, in the commercial world at least, this list is over the top and many companies use a simple approach such as numeric scale of 1 to 4 as outlined below:

	Showstopper	Very serious fault and includes GPF, assertion failure or complete system hang
	Unacceptable	Serious fault where software does not meet business requirements and there is no workaround
	Survivable	Fault that has an easy workaround - may involve partial manual operation
	Cosmetic	Covers trivial faults like screen layouts, colors, alignments, etc

Note that incident priority is not the same as severity. Priority relates to how soon the fault will be fixed and is often classified as follows:

1. Fix immediately.
2.Fix before the software is released.
3.Fix in time for the following release.
4.No plan to fix.

It is quite possible to have a severity 1 priority 4 incident and vice versa although the majority of severity 1 and 2 faults are likely to be assigned a priority of 1 or 2 using the above scheme.

5.12.4 Tracking and analysis

Incidents should be tracked from inception through various stages to eventual close-out and resolution. There should be a central repository holding the details of all incidents.

For management information purposes it is important to record the history of each incident. There must be incident history logs raised at each stage whilst the incident is tracked through to resolution for trace ability and audit purposes. This will also allow ht formal documentation of the incidents (and the departments who own them) at a particular point in time.

Typically, entry and exit criteria take the form of the number of incidents outstanding by severity. For this reason it is imperative to have a corporate standard for the severity levels of incidents.

Incidents are often analyzed to monitor test process and to aid in test process improvement. It is often useful to look at sample of incidents and try to determine the root cause.

5.13 Standards for testing

There are now many standards for testing, classified as QA standards, industry-specific standards and testing standards. These are briefly explained in this section. QA standards simple specify that testing should be performed, while industry-specific standards specify what level of testing to perform. Testing standards specify how to perform testing.

Ideally testing standards should be referenced from the other two.

The following table gives some illustrative examples of what we mean:

Type	Standard
QA Standards	ISO 9000
Industry specific standard	Railway signaling standard
Testing Standards	BS 7925-1, BS 7925-2

5.14 Summary.

In module five you have learnt that the Test Manager faces an extremely difficult challenge in managing the test team and estimating and controlling a particular test effort for a project. In particular you can now:

Suggest five different ways in which a test team might be organized.

Describe at least five different roles that a test team might have.

Explain why the number of test cycles and re-work costs are important factors in estimating.

Describe at least three ways that a test effort can be monitored.

List three methods of controlling the test effort to achieve the necessary completion criteria.

Prioritize incidents.

Understand the importance of logging all incidents.

Understand the need for tracking and analysis of incidents.