CONFIGURATION MANAGEMENT

Introduction

Changes –those due to the evolution of work products and those due to requirements changes—take place continuously in a software project.

Scenarios.

- Two different changes requests came from customer. The project manager assigned one request to Rao for implementation, and the other to Meera. Both had to modify code for Module X. When Meera finished her modification, she saved the file for X, inadvertently overwriting the changes Rao made a day early.
- Friday was the deadline for release of a module for which three members were developing the code. Integration of unit-tested code was planned for the final two days. On Tuesday night, Subbu, the developer of several key functions, left town to attend an emergency. The next day the module leader and the team members spent many hours looking through Subbu's files. They managed to identify some files containing the various functions Subbu was developing, but they found umpteen versions of these files. One of the team members had to work on the problem over weekend. Starting with some version of Subbu's programs, he developed and tested the unit, redoing the work that Subbu had almost finished before leaving. The module was finally delivered three days late.
- Srinath's team was in good spirits. They had finished the development on time, and the final testing had shown no bugs. The software was released to the customer for implementation. The next day Srinath received angry e-mails from the users and the customer, reporting problems in the software. After frantic effort by the team, the cause was found. The release version of the software contained an older version of a key component.

Concepts in Configuration Management

Definition. Configuration Management (CM) is the aspect of project management that focuses exclusively on systematically controlling the changes so that such problems do not occur.

A primary objective of CM is to manage the evolving configuration of the software system. In a project, a program's evolution takes it through many states.

1. At the beginning, when a programmer develops it, the program is under development ("or private")
2. Once the programmer is satisfied with the program, it moves into the "ready for unit testing" state. Only when the program reaches this state can it be unit tested.
3. After it has been unit tested, the programmer must fix any defects found. If the unit testing succeeds, however, then the program's state changes to "ready for system testing".
4. Only when all programs reach this state can system testing commence. Again, if defects are found during system testing, the state of a program reverts to "private"; otherwise, it moves to "ready for acceptance testing".
5. If the acceptance testing succeeds, the state of all programs changes to "ready for release", implying that they can now be released for "production use".
6. Once a program is released and is in production use, all programs (and associated documentation) move to the "baselined" state, which represents the state of the production system.

Kinds of CM functionality projects normally require:

- **Give the states of the programs** to decide when to start testing or when to release the software.
- **Give the latest version of a program** to make modifications in the latest version; otherwise, early changes may be lost.
- **Handle concurrent update requests** to be able to change the same program concurrently. Avoiding such a situation, requires access control so that only one person can make changes to a program at a time. If multiple parallel changes are allowed, reconciliation procedures should be implemented to ensure that all changes are reflected in the final version.
- **Undo a program change.** A change is made to a program (to implement a change request), but later a need arises to reverse this change request.
- **Prevent unauthorized changes or deletions**. Access control mechanisms are needed to disallow unapproved changes.
- **Provide traceability between requirement change requests and program changes.** Suppose a requirement change request dictates that three programs be modified, and these modifications have been assigned to three team members. How does a project manager ensure that this change requests has been properly implemented –that is, that all programs have been changed and that the changed programs have gone through their life

cycle and are in a "ready for release" state? Answering this question requires a mechanism to track change requests than can specify all programs to be changed as well as the state of each program.

- **Undo a requirement change.** A requirement change request that was implemented (by changing many programs) may later need to be undone (perhaps because the users do not like the new features).
- **Show associated changes.** Suppose a bug is found in a program, and it is suspected that this bug came from the implementation of a change request. It is desirable to review all changes made as a result of that change result.
- **Gather all sources, documents, and other information for the current system.** As a result of file corruption or a system crash, it might be necessary to recover all files.

The main purpose of CM is to provide mechanisms that handle the type of scenarios in the preceding list. These mechanisms include the following:

- **Conventions for naming and organization of files.** To help in finding a desired file quickly.
- **Version Control.** In order to preserve older versions of the programs whenever they are changed.
- **Change request traceability.** In order to provide mapping from a requirement change request to subsequent changes in the programs, and that helps in managing requirement changes.
- **Access control.** To ensure that only authorized people can modify certain files and that only one person can modify a file at any given time.
- **Reconciliation procedures.** To specify how two changes made independently to a program can be merged to create a new version that reflects both.
- **Modification login programs**. To trace change back to the change request.

If these mechanisms are provided, the scenarios given earlier can be handled satisfactorily. Some of these scenarios necessitate the use of more than one mechanism. For example, undoing a requirement change involves a mechanism to show the traceability of a requirement change to subsequent changes in programs, as well as a version control mechanism to actually undo the changes.

The discussion so far has focused on programs. The documents that are produced in a project (such as requirements documents, design documents, and plans) also need configuration management. During the normal course of a

project a document passes through three states: "under development", "under review", and "baselined".

The Configuration Management Process

As with most activities in project management the first stage in the CM process is **planning**: identifying those items that need to be under CM (know as *configuration items*), locations to store them, procedures for change control, and so on. The project manager or the configuration controller (CC) of the project prepares this plan. Then the process must be **executed**, perhaps by deploying a tool (the use of which also must be planned). Finally, because any CM plan requires discipline on the part of the project personnel in terms of maintaining versions, storing items in proper locations, and making changes properly, **monitoring** the status of the configuration items and **performing CM audits** are therefore other activities in the CM process.

1. Planning and Setting Up Configuration Management

Planning for configuration management involves identifying the configuration items and specifying the procedures to be used for controlling and implementing changes to them. Typical examples of configuration items include requirements specifications, design documents, source code, test plans, test scripts, test procedures, test data, standards used in the project (such as coding standards and design standards), the acceptance plan, documents such as the CM plan and the project plan, user documentation such as the user manual, and documents such as the training material, contract documents, quality records (review records, test records), and CM records (release records, status tracking records). Any customer-supplied products or purchased items that will be part of the delivery (called "included software product") are also configuration items.

During planning, the project manager must decide how to maintain the state of a program. One way is collecting items in different states is to create separate directories for them. All items in a certain state reside in the directory for that state. When the state of a program changes, that program is moved from the directory for the old state to the directory for the new state. This approach is a general one and does not require the use of any tool to maintain the state information.

The activities in this state include the following:

- Identify configuration items, including customer-supplied and purchased items
- Define a naming and numbering scheme for the configuration items
- Define the directory structure needed for CM
- Define access restrictions
- Define change control procedures
- Identify and define the responsibility and authority of the CC or Configuration Control Board (CCB)
- Define a method for tracking the status of configuration items
- Define a backup procedure
- Define a reconciliation procedure, if needed
- Define a release procedure
- Define an archival procedure
- Identify points at which the configuration items will be moved to the baseline

The output of this phase is the CM plan. The CC is responsible for the implementation of the CM plan. Depending on the size of the system under development, CC's role may be a part-time or a full-time job. The CC can also be responsible for managing the release, archiving the release, retrieving and releasing appropriate versions when required, and more.

2. Perform Configuration Control

Two main configuration control activities are performed: one that deals with managing the state transitions of programs (and documents), and one that deals with managing the change requests that must be implemented.

State transition management involves moving the items from one directory to another when the state changes and then creating versions when changes are made. Frequently, tools are used to manage the states and versions of items and access to them.

To implement requirement changes, which in turn trigger changes to configuration items, a change request is first analyzed by performing an impact analysis. This analysis determines the programs and documents that need to be changed and the cost and schedule implications of making the change. Once the change is approved by the project leader and the CC, all programs and documents identified in the impact analysis must be changed appropriately. A

spreadsheet-based mechanism is frequently used for tracking the status of change requests. For each change request, a spreadsheet is created that lists all programs being changed and their status.

3. Status Monitoring and Audits

If projects used a mechanism based on a directory structure to represent the state of a program, mistakes are possible. This type of mechanism requires that the programs be moved properly from one directory to another when their state changes and that the change of state be reflected in the master table that maintains the states of the various items. To minimize mistakes and catch any errors early, projects must perform regular status checking of the configuration items. A report can be produced about discrepancies, and all such discrepancies must be resolved.

In addition to checking the status of the items, projects must check the status of change requests. To accomplish this goal, change requests that have been received since last CM status monitoring operation are examined. For each change request, the state of the item as mentioned in the change request records is compared to the actual state. Checks can also be done to ensure that all modified items go through their full life cycle (that is, the state diagram) before they are incorporated in the baseline.