engineering task is produced and gets successfully reviewed. Milestones need not be placed for every activity. An approximate rule of thumb is to set a milestone every 10 to 15 days.

3.13 SOFTWARE CONFIGURATION MANAGEMENT

The results (also called as the deliverables) of a large software development effort typically consist of a large number of objects, e.g. source code, design document, SRS document, test document, user's manual, etc. These objects are usually referred to and modified by a number of software developers throughout the life cycle of the software. The state of all these objects at any point of time is called the *configuration* of the software product. The state of each deliverable object changes as development progresses and also as bugs are detected and fixed.

> Software configuration management deals with effectively tracking and controlling the configuration of a software product during its life cycle.

Before we discuss configuration management, we must be clear about the distinction between a version and a revision of a software product. A new version of a software is created when there is significant change in functionality, technology, or the hardware it runs on, etc. On the other hand, a new release is created if there is only a bug fix, minor enhancements to the functionality, usability, etc. Even the initial delivery might consist of several versions and more versions might be added later on. For example, one version of a mathematical computation package might run on Unix-based machines, another on Microsoft Windows, and so on. As a software is released and used by the customer, errors are discovered that need correction. Enhancements to the functionalities of the software may also be needed. A new **release** of software is an improved system intended to replace an old one. Often systems are described as version m, release n; or simple m.n. Formally, a history relation is version of can be defined between objects. This relation can be split into two subrelations is revision of and is variant of. In the following, we first discuss the necessity of configuration management and subsequently we discuss the configuration management activities and tools.

3.13.1 Necessity of Software Configuration Management

There are several reasons for putting an object under configuration management. But, possibly the most important reason for configuration management is to control the access to the different deliverable objects. Unless strict discipline is enforced regarding updation and storage of different objects, several problems can appear. The following are some of the important problems that can appear if configuration management is not used.

Inconsistency problem when the objects are replicated

Consider a scenario where every software developer has a personal copy of an object (e.g. source code). As each developer makes changes to his local copy, he is expected to intimate the changes that he has made to other developer, so that the changes in interfaces are uniformly changed across all modules. However, many times a developer makes changes to the interfaces in his local copies and forgets to intimate other teammates about the changes. This makes the different copies of the object inconsistent. Finally, when the product is in

tegrated, it does not work. Therefore, when several team members work on developing an object, it is necessary for them to work on a single copy of the object, otherwise inconsistency may arise.

Problems associated with concurrent access

Assume that only a single copy of a program module is maintained, and several developers are working on it. Two developers may simultaneously carry out changes to the different portions of the same module, and while saving overwrite each other. Though we explained the problem associated with concurrent access to program code, similar problems can occur for any other deliverable object.

Providing a stable development environment

When a project work is underway, the team members need a stable environment to make progress. Suppose one is trying to integrate module A, with the modules B and C. He cannot make progress if developer of module C keeps changing C; this can be especially frustrating if a change to module C forces recompilation of module A. When an effective configuration management is in place, the manager freezes the objects to form a *base line*. When any one needs to change any of the objects under configuration control, he is provided with a copy of the base line item. The requester makes changes to his private copy. Only after the requester is through with all modifications to his private copy, the configuration is updated and a new base line gets formed instantly. This establishes a baseline for others to use and depend on. Also, baselines may be archived periodically (Archiving means copying to a safe place such as a magnetic tape).

System accounting and maintaining status information

System accounting keeps track of who made a particular change to an object and when the change was made.

Handling variants

Existence of variants of a software product causes some peculiar problems. Suppose you have several variants of the same module, and find a bug in one of them. Then, it has to be fixed in all versions and revisions. To do it efficiently, you should not have to fix it in each and every version and revision of the software separately.