UNIT 8 SOFTWARE CHANGE MANAGEMENT

Structure **Page Nos.**

8.0 INTRODUCTION

Software change management is an umbrella activity that aims at maintaining the integrity of software products and items. Change is a fact of life but uncontrolled change may lead to havoc and may affect the integrity of the base product. Software development has become an increasingly complex and dynamic activity. Software change management is a challenging task faced by modern project managers, especially in a environment where software development is spread across a wide geographic area with a number of software developers in a distributed environment. Enforcement of regulatory requirements and standards demand a robust change management. The aim of change management is to facilitate justifiable changes in the software product.

8.1 OBJECTIVES

After studying this unit, you should be able to:

- define baselines;
- know the concept of version control and change control, and
- audit and report software change management activity.

8.2 BASELINES

Baseline is a term frequently used in the context of software change management. Before we understand the concept of baseline, let us define a term which is called software configuration item. A software configuration item is any part of development and /or deliverable system which may include software, hardware, firmware, drawings, inventories, project plans or documents. These items are independently tested, stored, reviewed and changed. A software configuration item can be one or more of the following:

- System specification
- Source code
- Object code
- Drawing
- Software design

- Design data
- Database schema and file structure
- Test plan and test cases
- Product specific documents
- Project plan
- Standards procedures
- Process description

Definition of Baseline: A baseline is an approved software configuration item that has been reviewed and finalised. An example of a baseline is an approved design document that is consistent with the requirements. The process of review and approval forms part of the formal technical review. The baseline serves as a reference for any change. Once the changes to a reference baseline is reviewed and approved, it acts as a baseline for the next change(s).

A baseline is a set of configuration items (hardware, documents and software components) that have been formally reviewed and agreed upon, thereafter serve as the basis for future development, and that can be changed only through formal change control procedures.

Figure 8.1 depicts a baseline for design specification.

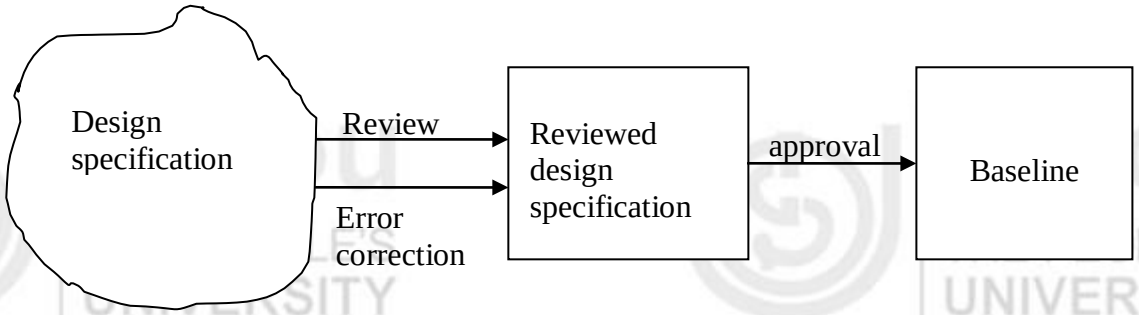

Figure 8.1 : A baseline for design specification

Figure 8.2 depicts the evolution of a baseline.

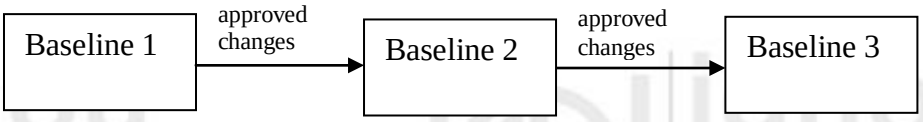

Figure 8.2: Evolution of a baseline

A baseline is functionally complete, i.e., it has a defined functionality. The features of these functionalities are documented for reference for further changes. The baseline has a defined quality which has undergone a formal round of testing and reviews before being termed as a baseline. Any baseline can be recreated at any point of time.

The process of change management

The domain of software change management process defines how to control and manage changes.

A formal process of change management is acutely felt in the current scenario when the software is developed in a very complex distributed environment with many versions of a software existing at the same time, many developers involved in the development process using different technologies. The ultimate bottomline is to maintain the integrity of the software product while incorporating changes.

The following are the objectives of software change management process:

1. **Configuration identification**: The source code, documents, test plans, etc. The process of identification involves identifying each component name, giving them a version name (a unique number for identification) and a configuration identification.
2. **Configuration control**: Cntrolling changes to a product. Controlling release of a product and changes that ensure that the software is consistent on the basis of a baseline product.
3. **Review**: Reviewing the process to ensure consistency among different configuration items.
4. **Status accounting** : Recording and reporting the changes and status of the components.
5. **Auditing and reporting**: Validating the product and maintaining consistency of the product throughout the software life cycle.

Process of changes: As we have discussed, baseline forms the reference for any change. Whenever a change is identified, the baseline which is available in project database is copied by the change agent (the software developer) to his private area. Once the modification is underway the baseline is locked for any further modification which may lead to inconsistency. The records of all changes are tracked and recorded in a status accounting file. After the changes are completed and the changes go through a change control procedure, it becomes a approved item for updating the original baseline in the project database.

Figure 8.3 depicts the process of changes to a baseline.

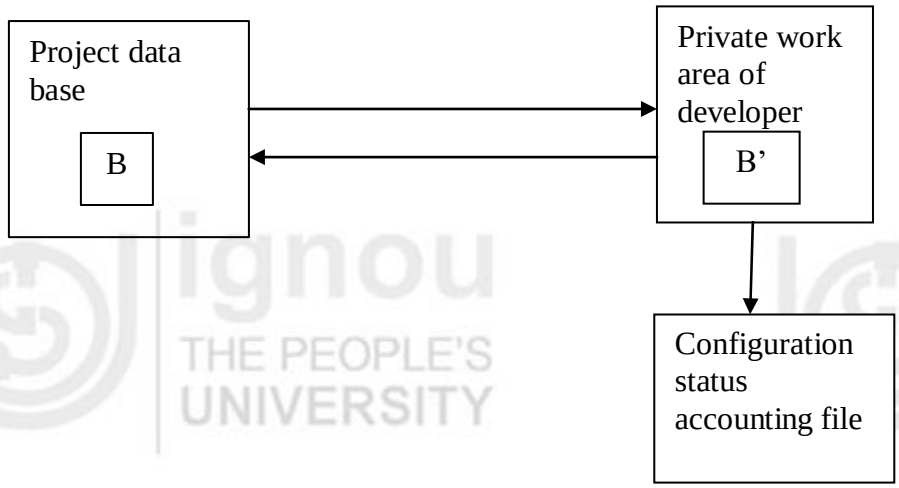

Figure 8.3: Process of changes to baseline

All the changes during the process of modification are recorded in the configuration status accounting file. It records all changes made to the previous baseline B to reach the new baseline B'. The status accounting file is used for configuration authentication which assures that the new baseline B' has all the required planned and approved changes incorporated. This is also known as auditing.

Check Your Progress 1

1) _____serves as reference for any change.

2) What is the aim of software change management process?

……...

……...

8.3 VERSION CONTROL

Version control is the management of multiple revisions of the same unit of item during the software development process. For example, a system requirement specification (SRS) is produced after taking into account the user requirements which change with time into account. Once a SRS is finalized, documented and approved, it is given a document number, with a unique identification number. The name of the items may follow a hierarchical pattern which may consist of the following:

- Project identifier
- Configuration item (or simply item, e.g. SRS, program, data model)
- Change number or version number

The identification of the configuration item must be able to provide the relationship between items whenever such relationship exists.

The identification process should be such that it uniquely identifies the configuration item throughout the development life cycle, such that all such changes are traceable to the previous configuration. An evolutionary graph graphically reflects the history of all such changes. The aim of these controls is to facilitate the return to any previous state of configuration item in case of any unresolved issue in the current unapproved version.

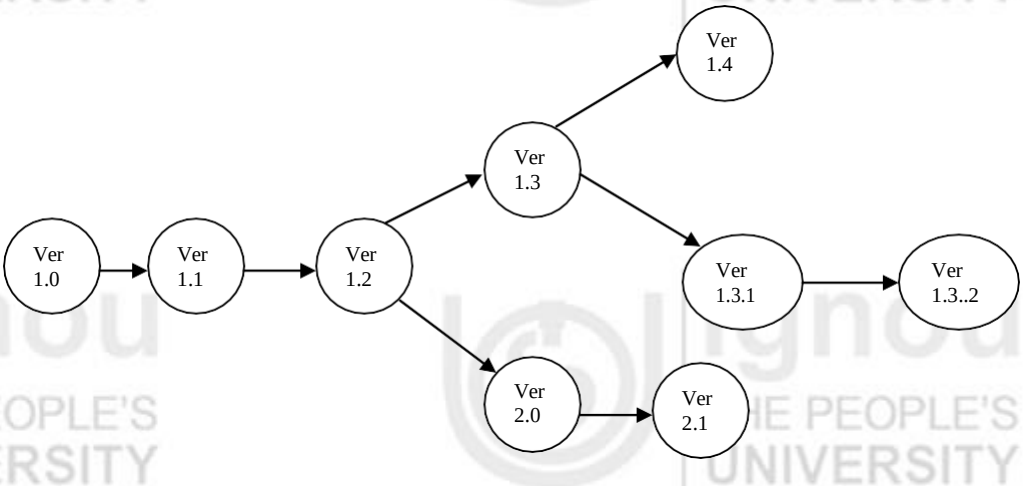

Figure 8.4 : An evolutionary graph for a different version of an item

The above evolutionary graph(*Figure 8.4*) depicts the evolution of a configuration item during the development life cycle. The initial version of the item is given version number Ver 1.0. Subsequent changes to the item which could be mostly fixing bugs or adding minor functionality is given as Ver 1.1 and Ver 1.2. After that, a major modification to Ver 1.2 is given a number Ver 2.0 at the same time, a parallel version of the same item without the major modification is maintained and given a version number 1.3.

Depending on the volume and extent of changes, the version numbers are given by the version control manager to uniquely identify an item through the software development lifecycle. It may be noted that most of the versions of the items are released during the software maintenance phase.

Software engineers use this version control mechanism to track the source code, documentation and other configuration items. In practice, many tools are available to store and number these configuration items automatically. As software is developed and deployed, it is common to expect that multiple versions of the same software are deployed or maintained for various reasons. Many of these versions are used by developers to privately work to update the software.

It is also sometimes desirable to develop two parallel versions of the same product where one version is used to fix a bug in the earlier version and other one is used to develop new functionality and features in the software. Traditionally, software developers maintained multiple versions of the same software and named them uniquely by a number. But, this numbering system has certain disadvantages like it does not give any idea about a nearly identical versions of the same software which may exist.

The project database maintains all copies of the different versions of the software and other items. It is quite possible that without each other's knowledge, two developers may copy the same version of the item to their private area and start working on it. Updating to the central project database after completing changes will lead to overwriting of each other's work. Most version control systems provide a solution to this kind of problem by locking the version for further modification.

Commercial tools are available for version control which performs one or more of following tasks;

- Source code control
- Revision control
- Concurrent version control

There are many commercial tools like Rational ClearCase, Microsoft Visual SourceSafe and a number of other commercial tools to help version control.

Managing change is an important part of computing. The programmer fixes bugs while producing a new version based on the feedback of the user. System administrator manages various changes like porting database, migrating to a new platform and application environment without interrupting the day to day operations. Revisions to documents are carried out while improving application.

An example of revision control

Let us consider the following simple HTML file in a web based application (welcome.htm)

```
<html>
<head>
<Title> A simple HTML Page</title>
</head>
<body>
<h1> Welcome to HTML Concepts</h1>
</body>
</html>
```

Once the code is tested and finalized, the first step is to register the program to he project database. The revision is numbered and this file is marked read-only to prevent any further undesirable changes. This forms the building block of source control. Each time the file is modified, a new version is created and a new revision number is given.

The first version of the file is numbered as version 1.0. Any further modification is possible only in the developer's private area by copying the file from the project

database. The process of copying the configuration object (the baseline version) is called check-out.

Figure 8.5 depicts the changes to a baselined file in project database.

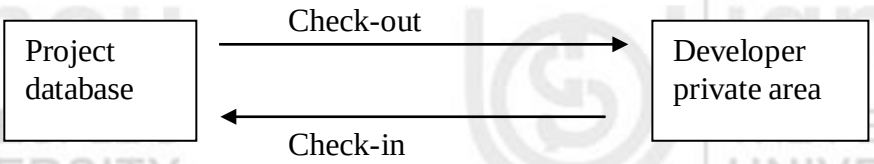

Figure 8.5: Changes to a baselined file in project database

The version (revision) control process starts with registering the initial versions of the file. This essentially enforces a check on the changes which ensure that the file can't be changed unless it is checked-out from the project database.

When a change is required to be made, allowing an email address to be added to the above html file, the developer will extract the latest version of the file welcome.htm from the project database. Once it is checked-out from the project database for modification, the file is locked for further modifications by any other developer. Other developers can check-out the file for read-only purpose.

Now, the following lines are added to the file welcome.htm.

```
<hr>
a href=mailto:webmaster@xyz.com> webmaster</a>
<hr>
```

The revised version of the file welcome.htm become

```
<html>
<head>
<Title> A simple HTML Page</title>
</head>
<body>
<h1> Welcome to HTML Concepts</h1>
<hr>
a href=mailto:webmaster@xyz.com> webmaster</a>
<hr>
</body>
</html>
```

Then the developer check-in's the revised version of the file to the project database with a new version (revision) number version 1.1 i.e. the first revision along with the details of the modification done.

Suppose another modification is done by adding a graphic to the html file welcome.htm. This becomes version 1.2. The version tree after two modifications looks as shown below (*Figure 8.6*).

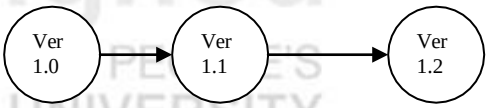

Figure 8.6: Version tree of welcome.htm

Suppose further modification is required for text-based browser as graphic will not be supported by text-based browser. Then the version 1.1 will be selected from the project database. This shows the necessity of storing all versions of the file in the project database.

☞ **Check Your Progress 2**

1) _____is an example of source code control tool.

2) Version control mechanism allows multiple versions of an item to be maintained at the same time. (Yes/No)

3) How do version control systems ensure that two software developers do not attempt the same change at the same time?

……

……...

8.4 CHANGE CONTROL

Change is a fact of life, and the same applies to software development. Although, all changes requested by the user are not justified changes, but most of them are. The real challenge of change manager and project leader is to accept and accommodate all justifiable changes without affecting the integrity of product or without any side effect. The central to change management process is change control which deals with the formal process of change control.

The adoption and evolution of changes are carried out in a disciplined manner. In a large software environment where, as changes are done by a number of software developers, uncontrolled and un-coordinated changes may lead to havoc grossly diverting from the basic features and requirements of the system. For this, a formal change control process is developed.

Change control is a management process and is to some extent automated to provide a systematic mechanism for change control. Changes can be initiated by the user or other stake holder during the maintenance phase, although a change request may even come up during the development phase of the software.

A change request starts as a beginning of any change control process. The change request is evaluated for merits and demerits, and the potential side effects are evaluated. The overall impact on the system is assessed by the technical group consisting of the developer and project manager. A change control report is generated by the technical team listing the extent of changes and potential side effects. A designated team called change control authority makes the final decision, based on the change control report, whether to accept or reject the change request.

A change order called engineering change order is generated after the approval of the change request by the change control authority. The engineering change order forms the starting point of effecting a change in the component. If the change requested is

not approved by the change control authority, then the decision is conveyed to the user or the change request generator.

Once, change order is received by the developers, the required configuration items are identified which require changes. The baseline version of configuration items are copied from the project data base as discussed earlier.

The changes are then incorporated in the copied version of the item. The changes are subject to review (called audit) by a designated team before testing and other quality assurance activity is carried out. Once the changes are approved, a new version is generated for distribution.

The change control mechanisms are applied to the items which have become baselines. For other items which are yet to attain the stage of baseline, informal change control may be applied. For non- baseline items, the developer may make required changes as he feels appropriate to satisfy the technical requirement as long as it does not have an impact on the overall system.

The role of the change control authority is vital for any item which has become a baseline item. All changes to the baseline item must follow a formal change control process.

As discussed, change request, change report and engineering change order (change order) are generated as part of the change control activity within the software change management process. These documents are often represented inprinted or electronic forms. The typical content of these documents is given below:

Software Change Request Format

1.0 Change request Identification

1.1 Name, identification and description of software configuration item(s): The name, version numbers of the software configuration is provided. Also, a brief description of the configuration item is provided.

1.2 Requester and contact details: The name of the person requesting the change and contact details

1.3 Date, location, and time when the change is requested

2.0 Description of the change

2.1 Description : This section specifies a detailed description of the change request.

2.1.1 Background Information, Background information of the request.

2.1.2 Examples: Supporting information, examples, error report, and screen shoots

2.1.3 The change : A detailed discussion of the change requested.

2.2 Justification for the change : Detailed justification for the request.

2.3 Priority : The priority of the change depending on critical effect on system functionalities.

Software Change Report Format

1.0 Change report Identification

1.1 Name, identification and description of software configuration item(s): The name, version numbers of the software configuration item and a brief description of it.

1.2 Requester: The name and contact details of the person requesting the change.

1.3 Evaluator : The name of the person or team who evaluated the change request.

1.4 Date and time : When change report was generated.

2.0 Overview of changes required to accommodate request

2.1 Description of software configuration item that will be affected

2.2 Change categorization : Type of change, in a generic sense

2.3 Scope of the change : The evaluator's assessment of the change.

2.3.1 Technical work required including tools required etc. A description of the work required to accomplish the change including required tools or other special resources are specified here

2.3.2 Technical risks : The risks associated with making the change are described.

3.0 Cost Assessment : Cost assessment of the requested change including an estimate of time required.

4.0 Recommendation

4.1 Evaluator’s recommendation : This section presents the evaluator's recommendation regarding the change

4.2 Internal priority: How important is this change in the light of the business operation and priority assigned by the evaluator.

Engineering Change Order Format

1.0 Change order Identification

1.1 Name, identification and description of software configuration item(s) : The name, version numbers including a brief description of software configuration items is provided.

1.2 Name of Requester

1.3 Name of Evaluator

2.0 Description of the change to be made

2.1 Description of software configuration(s) that is affected

2.2 Scope of the change required

The evaluator's assessment of scope of the change in the configuration item(s).

2.2.1 Technical work and tools required : A description of the work and tools required to accomplish the change.

2.3 Technical risks: The risks associated with making the change are described in this section.

3.0 Testing and Validation requirements

A description of the testing and review approach required to ensure that the change has been made without any undesirable side effects.

3.1 Review plan : Description of reviews that will be conducted.

3.2 Test plan

Description of the test plans and new tests that are required.

Benefits of change control management

The existence of a formal process of change management helps the developer to identify the responsibility of code for which a developer is responsible. An idea is achieved about the changes that affect the main product. The existence of such mechanism provides a road map to the development process and encourages the developers to be more involved in their work.

Version control mechanism helps the software tester to track the previous version of the product, thereby giving emphasis on testing of the changes made since the last approved changes. It helps the developer and tester to simultaneously work on multiple versions of the same product and still avoid any conflict and overlapping of activity.

The software change management process is used by the managers to keep a control on the changes to the product thereby tracking and monitoring every change. The existence of a formal process reassures the management. It provides a professional approach to control software changes.

It also provides confidence to the customer regarding the quality of the product.

8.5 AUDITING AND REPORTING

Auditing

Auditing and Reporting helps change management process to ensure whether the changes have been properly implemented or not, whether it has any undesired impact on other components. A formal technical review and software configuration audit helps in ensuring that the changes have been implemented properly during the change process.

A Formal Technical Review generally concentrates on technical correctness of the changes to the configuration item whereas software configuration audit complements it by checking the parameters which are not checked in a Formal Technical Review.

A check list for software configuration audit

- Whether a formal technical review is carried out to check the technical accuracy of the changes made?
- Whether the changes as identified and reported in the change order have been incorporated?
- Have the changes been properly documented in the configuration items?
- Whether standards have been followed.
- Whether the procedure for identifying, recording and reporting changes has been followed.

As it is a formal process, it is desirable to conduct the audit by a separate team other than the team responsible for incorporating the changes.

Reporting: Status reporting is also called status accounting. It records all changes that lead to each new version of the item. Status reporting is the bookkeeping of each release. The process involves tracking the change in each version that leads the latest(new) version.

The report includes the following:

- The changes incorporated
- The person responsible for the change
- The date and time of changes
- The effect of the change
- The reason for such changes (if it is a bug fixing)

Every time a change is incorporated it is assigned a unique number to identify it from the previous version. Status reporting is of vital importance in a scenario where a large number of developers work on the same product at the same time and have little idea about the work of other developers.

For example, in source code, reporting the changes may be as below:

****
****
****

Title : Sub routine Insert to Employee Data

Version : Ver 1.1.3

Purpose : To insert employee data in the master file

Author : John Wright

Date : 23/10/2001

Auditor : J Waltson

Modification History:

12/12/2002 : by D K N
To fix bugs discovered in the first release

4/5/2003 : by S K G
to allow validation in date of birth data

6/6/2004 : by S S P
To add error checking module as requested by the customer

☞ Check Your Progress 3

1) Who decides the acceptance of a change request?.

……

……

2) How auditing is different from a Formal Technical Review (FTR)?

……

……

8.6 SUMMARY

Software change management is an activity that is applied throughout the software development life cycle. The process of change management includes configuration identification, configuration control, status reporting and status auditing. Configuration identification involves identification of all configurations that require changes. Configuration or change control is the process of controlling the release of the product and the changes. Status accounting or reporting is the process of recording the status of each component. Review involves auditing consistency and completeness of the component.

8.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Baseline.
2) The domain of software change management process defines how to control and manage changes. The ultimate aim is to maintain the integrity of the software product while incorporating changes.

Check Your Progress 2

1) Microsoft Visual SourceSafe
2) Yes
3) Version control system locks the configuration item once it is copied from project database for modification by a developer.

Check Your Progress 3

1) Change control authority
2) Formal Technical Review is a formal process to evaluate the technical accuracy of any process or changes. Whereas software change or audit is carried out by a separate team to ensure that proper change management procedure has been followed to incorporate the changes.

8.8 FURTHER READINGS

1) *Software Engineering, Sixth Edition, 2001,* Ian Sommerville; Pearson Education.
2) *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill International Edition.

Reference websites

http://www.rspa.com
http://www.ieee.org