MCA PART II

Paper-XI

Topic: SOFTWARE CHANGE MANAGEMENT

Prepared by: Dr. Kiran Pandey

(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

Introduction

Changes are inevitable when software is built. A primary goal of software engineering is to improve the ease with which changes can be made to software. 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.

Configuration management is all about change control. Every software engineer has to be concerned with how changes made to work products are tracked and propagated throughout a project. To ensure that quality is maintained the change process must be audited. A Software Configuration Management (SCM) Plan defines the strategy to be used for change management.

Generally, once the SRS is finalized there is less chance of requirement of changes from user. If they occur, the changes are addressed only with prior approval of higher management, as there is a possibility of cost and time overrun.

Software Configuration Items (SCI)

- Computer programs (both source and executable)
- Documentation (both technical and user)
- Data (contained within the program or external to it)

Fundamental Sources of Change

- New business or market conditions dictate changes to product requirements or business rules
- New stakeholder needs demand modification of data, functionality, or services delivered by a computer-based system
- Business reorganization causes changes in project priorities or software engineering team structure
- Budgetary or scheduling constraints cause system to be redefined

Configuration Management System Elements

- *Component elements* – set of tools coupled within a file management system to enable access to and management of each SCI
- *Process elements* – collection of procedures and tasks that define and effective approach to change management for all stakeholders
- *Construction elements* – set of tolls that automate the construction of software by ensure a set of validated components is assembled
- *Human elements* – team uses a set of tools and process features encompassing other CM elements

Baseline

In configuration management, a "baseline" is an agreed description of the attributes of a product, at a point in time, which serves as a basis for defining change. A "change" is a movement from this baseline state to a next state. The identification of significant changes from the baseline state is the central purpose of **baseline** identification.

Typically, significant states are those that receive a formal approval status, either explicitly or implicitly. An approval status may be marked individually, when a prior definition for that status has been established by project leaders, or signified by association to a position above or below the established baseline. Nevertheless, this approval status is usually recognized publicly. Thus, a baseline may also mark an approved configuration item, e.g. a project plan that has been signed off for execution. In a similar manner, associating multiple configuration items with such a baseline indicates those items as being approved.

Generally, a baseline may be a single work product, or set of work products that can be used as a logical basis for comparison. A baseline may also be established as the basis for subsequent select activities when the work products meet certain criteria. Such activities may be attributed with formal approval.

Conversely, the configuration of a project often includes one or more baselines, the status of the configuration, and any metrics collected. The current configuration refers to the current status, current audit and/or current metrics. Similarly, but less frequently, a baseline may refer to all items associated with a specific project. This may include all

revisions of all items, or only the latest revision of all items in the project, depending upon context.

A baseline may be a specific type of baseline, such as the body of items at a particular certifying review. Some examples include:

- Functional Baseline: initial specifications established; contract, etc.
- Allocated Baseline: state of work products after requirements are approved
- Developmental Baseline: state of work products amid development
- Product Baseline: contains the releasable contents of the project
- others, based upon proprietary business practices

Baseline can be summarized as follows:

- A work product becomes a baseline only after it is reviewed and approved.
- A baseline is a milestone in software development that is marked by the delivery of one or more configuration items.
- Once a baseline is established each change request must be evaluated and verified by a formal procedure before it is processed.
- Baseline work products are place in a project database or repository

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 bottom line 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**: Controlling 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.

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.**

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.

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.

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

- Source code control
- Revision control
- Concurrent version control

Change Control

- Change control is manual step in software lifecycle. It combines human procedures and automated tools.
    - Change control process is illustrated in following figure 2.
    - Change request submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration object and system function, and project cost of change.
- The result of the evaluation are presented as a change report, which is used by the change control authority(CCA) – A person or group who make final decision on the status and priority of the change.
- An engineering change order (ECO) is generated for each approved change. The ECO describes the change order to be made, the constraints that must be respected, and the criteria for view and audit.
- The object to be changed can be placed in a directory that is controlled by software engineer making the change. As an alternative, the object to be changed can be

"checked out" of the project database, change is made, and appropriate SQA activities are applied.

- The object are then "checked in" to the database and appropriate version control mechanism are used to create the next version of the software.
- Checked in and Checked out mechanism require two important elements
- Access Control
- Synchronization Control
    - The Access control mechanism gives the authority to the software engineer to access and modify the specific configuration object.
    - The Synchronization control mechanism allows to make parallel changes or the change made by two different people without overwriting each other's work.

Change control is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations.

A change in the configuration of product goes through following steps -

- **Identification** - A change request arrives from either internal or external source. When change request is identified formally, it is properly documented.
- **Validation** - Validity of the change request is checked and its handling procedure is confirmed.
- **Analysis** - The impact of change request is analyzed in terms of schedule, cost and required efforts. Overall impact of the prospective change on system is analyzed.
- **Control** - If the prospective change either impacts too many entities in the system or it is unavoidable, it is mandatory to take approval of high authorities before change is incorporated into the system. It is decided if the change is worth incorporation or not. If it is not, change request is refused formally.
- **Execution** - If the previous phase determines to execute the change request, this phase take appropriate actions to execute the change, does a thorough revision if necessary.

- **Close request** - The change is verified for correct implementation and merging with the rest of the system. This newly incorporated change in the software is documented properly and the request is formally is closed.

Benefits of change control

- 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.

Auditing and Reporting

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.

Software Re-engineering

When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.

Legacy software cannot keep tuning with the latest technology available in the market. As the hardware become obsolete, updating of software becomes a headache. Even if software grows old with time, its functionality does not.

For example, initially Unix was developed in assembly language. When language C came into existence, Unix was re-engineered in C, because working in assembly language was difficult.

Other than this, sometimes programmers notice that few parts of software need more maintenance than others and they also need re-engineering.

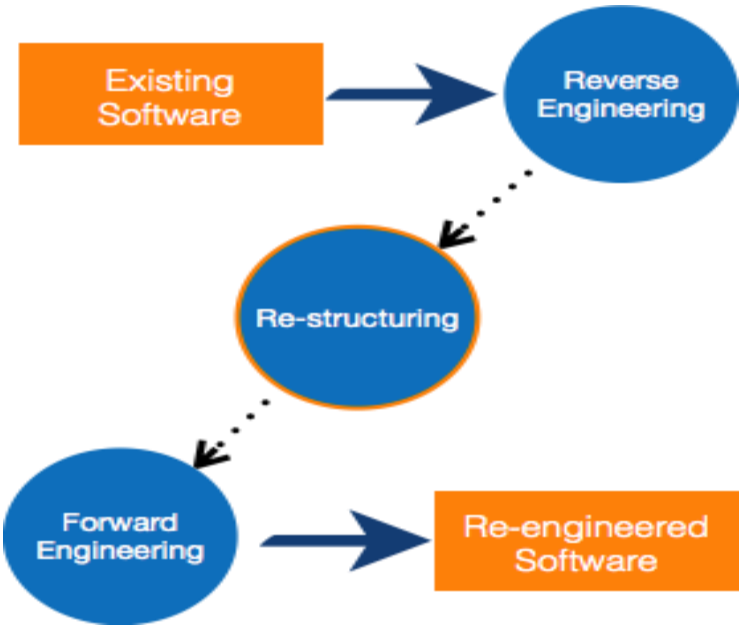

Figure 1: Software Re-engineering process

Re-Engineering Process

- **Decide** what to re-engineer. Is it whole software or a part of it?
- **Perform** Reverse Engineering, in order to obtain specifications of existing software.
- **Restructure Program** if required. For example, changing function-oriented programs into object-oriented programs.
- **Re-structure data** as required.
- **Apply Forward engineering** concepts in order to get re-engineered software.

There are few important terms used in Software re-engineering

Reverse Engineering

It is a process to achieve system specification by thoroughly analyzing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels.

An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.

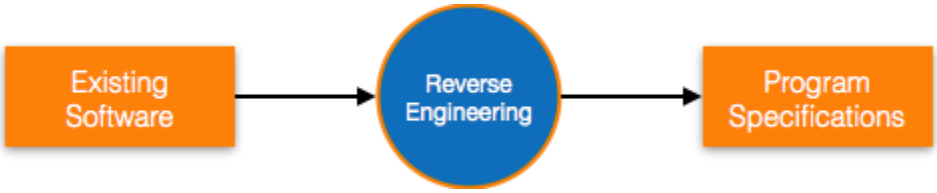

Figure 2: Reverse engineering

Program Restructuring

It is a process to re-structure and re-construct the existing software. It is all about re-arranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code-restructuring and data-restructuring or both.

Re-structuring does not impact the functionality of the software but enhance reliability and maintainability. Program components, which cause errors very frequently can be changed, or updated with re-structuring.

The dependability of software on obsolete hardware platform can be removed via re-structuring.

Forward Engineering

Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.

Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering.

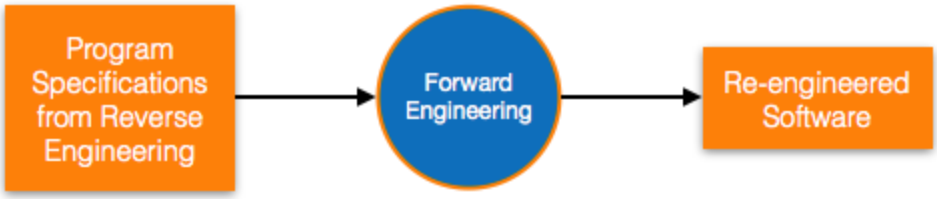

Figure 3: Forward engineering