Unit 6.2 software maintenance and configuration management

Prepared by :Dr.Pooja M Bhatt
CE Department
MBIT

Outlines

- Types of Software Maintenance
- Re-Engineering
- Reverse Engineering
- Forward Engineering
- The SCM Process
- Identification of Objects in the Software Configuration
- Version Control and Change Control

Types of Software Maintenance

- In a software lifetime, type of maintenance may vary based on its nature
- It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature
- Following are some types of maintenance based on their characteristics
- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

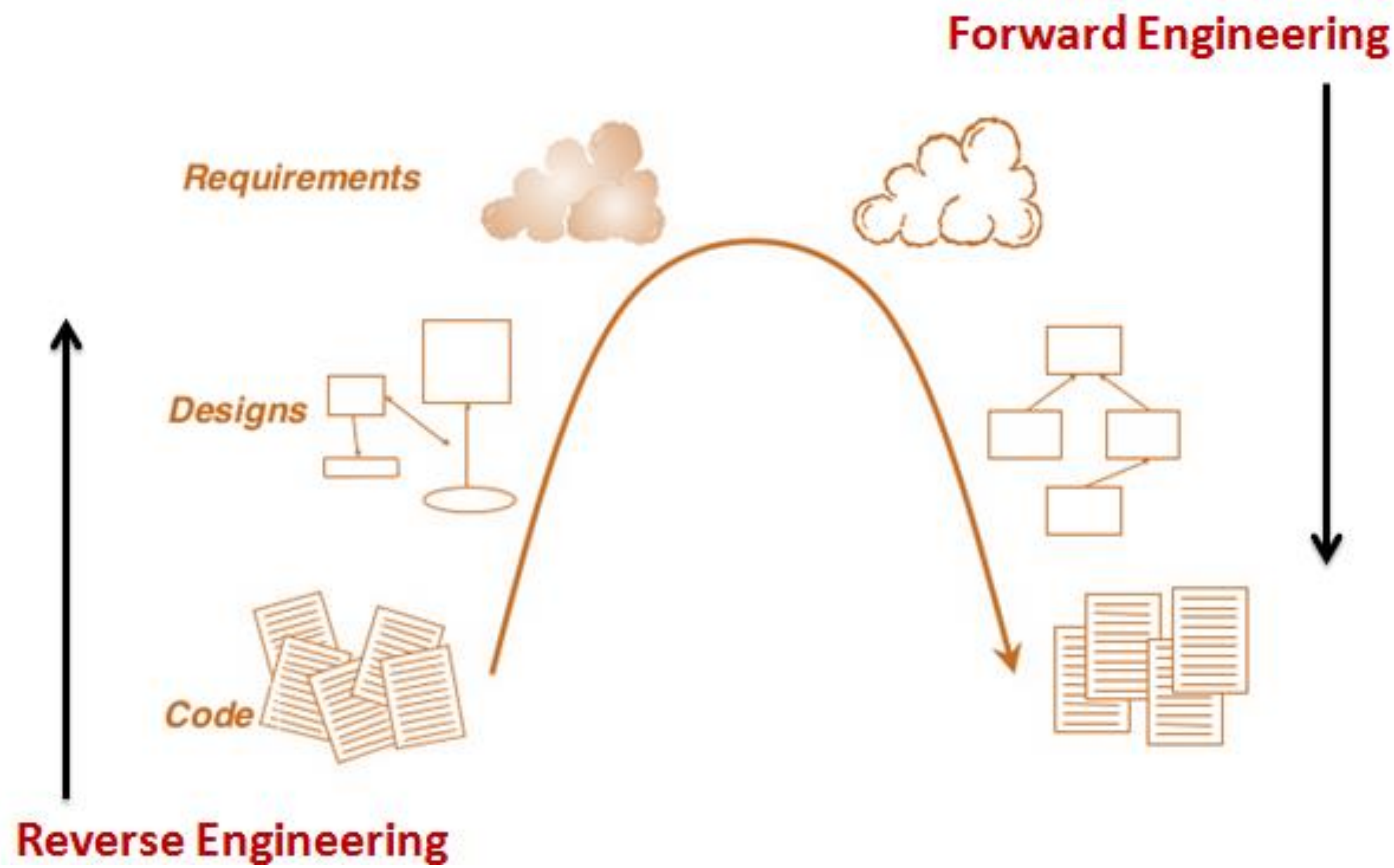
- **Corrective Maintenance**
- This includes **modifications** done in order to **fix problems**
- Corrective maintenance deals with the **repair of defects** found in day-to-day system functions
- **Adaptive Maintenance**
- This includes **modifications** applied to **keep the software product up-to-date**
- Adaptive maintenance is the **implementation of changes** in a part of the system, which has been **affected by a change that occurred in some other part of the system**

- **Perfective Maintenance**
- This includes **modifications** done in order to **keep** the **software usable over long period of time**
- It **includes new features, new user requirements** for refining the software and **improve its reliability** and **performance**.
- This includes **changing** the **functionalities** of as per the **user's changing needs**
- **Preventive Maintenance**
- **Modifications** to **prevent future problems** of software
- It aims to **attend problems**, which are **not significant at this moment** but **may cause serious issues in future**
- It comprises **documentation updating**, **code optimization** and **code restructuring**.

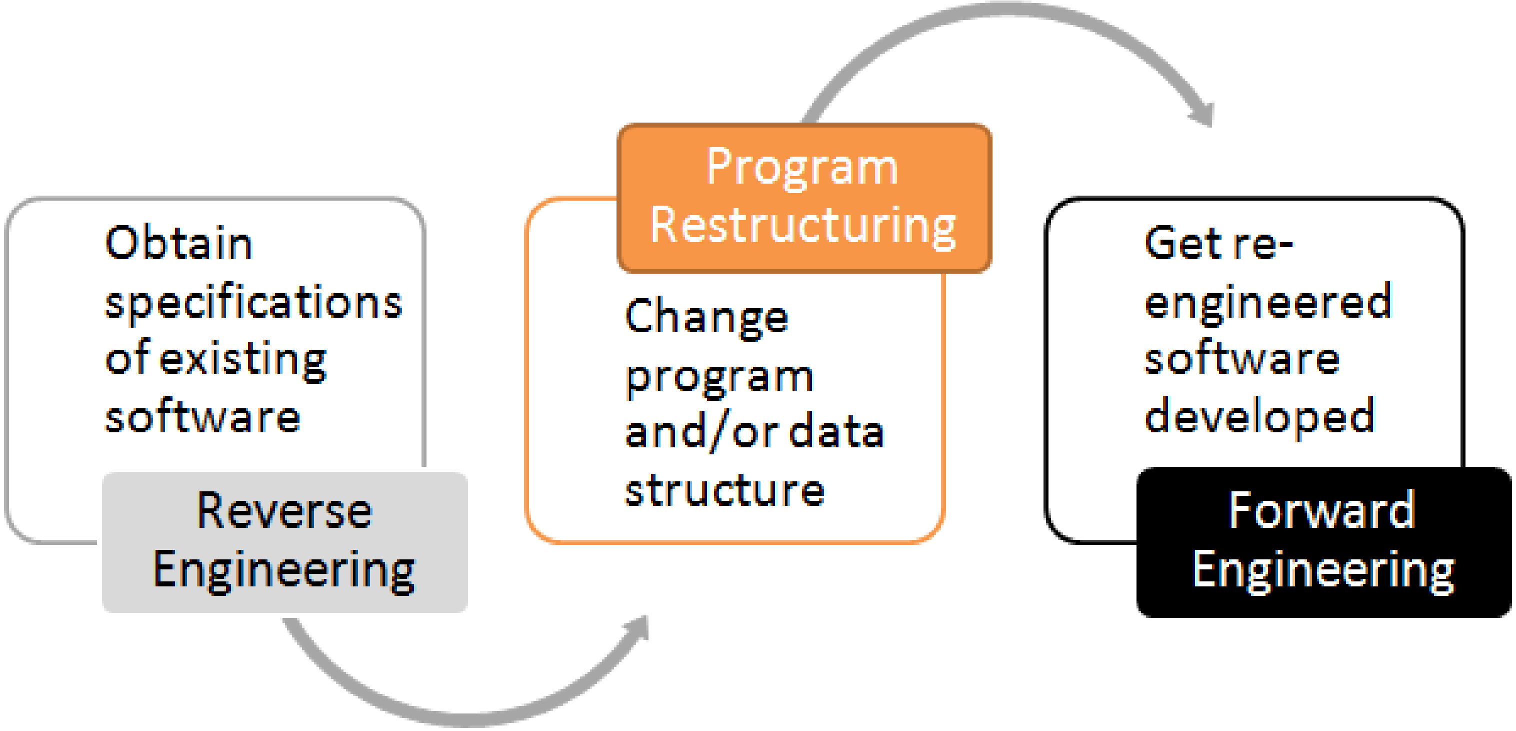
Re-Engineering

- 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 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
- 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

Re-Engineering



Re-Engineering



Obtain specifications of existing software

Reverse Engineering

Program Restructuring

Change program and/or data structure

Get re-engineered software developed

Forward Engineering

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 and re-structure data as required
- Apply Forward engineering concepts in order to get re-engineered software

Reverse Engineering

- **Reverse engineering** can **extract design information** from **source code**
- The **abstraction level** of a **reverse engineering process** refers to the **sophistication of the design information** that can be extracted from source code
- Ideally, the **abstraction level** should be **as high as possible**
- The **reverse engineering** process should be **capable of** Deriving **procedural design representations**(a low-level abstraction)
- **Program** and **data structure information** (a somewhat higher level of abstraction)
- **Object models, data flow models** (a relatively high level of abstraction)
- **Entity relationship models** (a high level of abstraction).

- As the **abstraction level increases**, information will allow **easier understanding of the program**
- **Interactivity refers** to the **degree** to which the **human** is **"integrated"** with **automated tools** to **create an effective reverse engineering** process
- In most cases, as **the abstraction level increases, interactivity must increase**
- The **directionality** of the reverse engineering process is **one-way**, all **information extracted** from the source code is **provided to** the **software engineer**

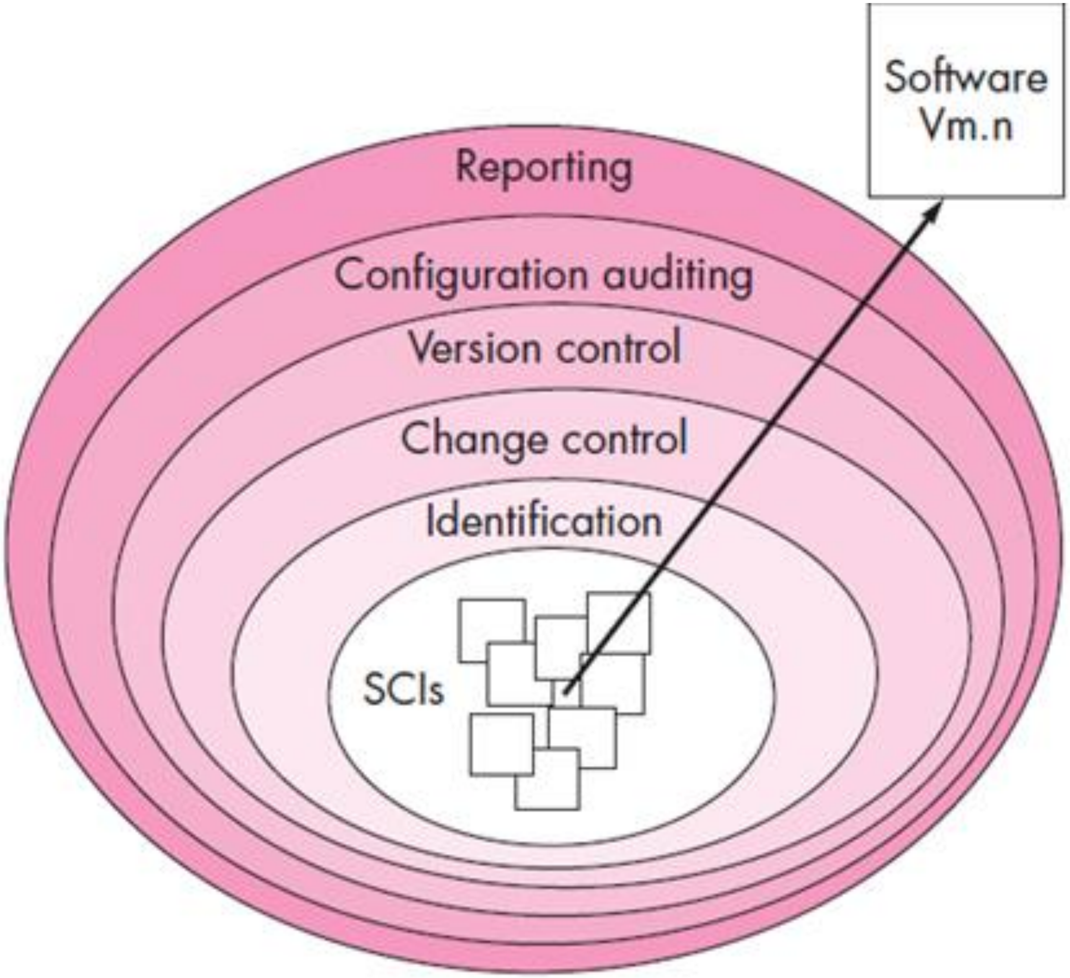
Forward engineering

- **Forward engineering** is a process of **obtaining desired software from** the **specifications**, which were brought by reverse engineering
- Forward engineering is **same** with only one difference **it is engineering as software engineering carried out always after process reverse**
- In most cases, **forward engineering** does **not simply create** a **modern equivalent** of an **older program**
- **Rather**, new **user** and **technology requirements** are **integrated** into the reengineering effort
- The **redeveloped** program **extends** the **capabilities** of the **older application**

Software Configuration Management

- **The SCM (Software Configuration Management)** is a set of **activities** that have been developed **to manage change** throughout the **software life cycle**
- **Purpose:**
- **Systematically control changes** to the configuration and **maintain** the **integrity** and **traceability** of the configuration throughout the system's life cycle
- **Four primary objectives**
- To **identify** all **items** that collectively **define** the **software configuration** • To **manage changes** to one or more of these items
- To facilitate the **construction** of different **versions** of an application
- To **ensure** that software **quality** is maintained as the **configuration evolves over time**

Layers of SCM Process



The SCM Process

- Referring to the figure, SCM tasks can viewed as concentric layers
- **SCIs (Software Configuration Item)** flow outward through these layers throughout their useful life
- As an **SCI moves through a layer**, the **actions** implied by each SCM task **may or may not be applicable**
- For example, when a new SCI is created, it must be identified.
- However**, if no changes are requested** for the SCI, the **change control layer does not apply**
- The **SCI** is **assigned to a specific version** of the software (version control mechanisms come into play)
- A **record** of the **SCI** (its name, creation date, version, etc.) is **maintained** for configuration **auditing purposes**

Identification Task

- **Identification** separately **names** each **SCI** and then **organizes it** in the **SCM repository** using an object-oriented approach
- Objects start out as basic objects and are then grouped into aggregate objects.
- **Each object** has a set of distinct features that identify it.
- A **name** that is **unambiguous** to all **other objects**
- A **description** that contains **the SCI type**, a project **identifier**, and **change** and/or **version** information
- List of **resources needed** by the object
- The **object realization** (i.e., the document, the file, the model, etc.)

Change Control

- **Change control** is a procedural **activity** that **ensures quality** and **consistency** as changes are made to a configuration object
- A **change request** is submitted to a configuration control authority, which is usually a change control board (**CCB**).
- The **request** is **evaluated** for **technical merit**, **potential side effects**, **overall impact** on other configuration objects and system functions, and **projected cost** in terms of **money**, **time** and **resources**
- An **engineering change order** (ECO) is **issued** for each approved change request
- **Describes** the **change** to be made, the **constraints** to follow and the **criteria for review** and **audit**

- The **Base lined SCI** is **obtained** from the SCM **repository**
- **Access control** governs **which** software **engineers** have the **authority** to **access** and **modify** a particular configuration **object**
- **Synchronization control** helps to ensure that **parallel changes** performed by two **different people don't overwrite** one another.

Version Control

- **Version control** is a set of **procedures** and **tools** for **managing** the **creation** and **use of multiple occurrences** of **objects** in the **SCM repository**
- **Version Control Capabilities**
- An SCM **repository** that **stores all relevant configuration** objects
- A **version management capability** that stores **all versions** of a configuration **object**
- A make **facility** that **enables** the software **engineer** to **collect** all **relevant configuration objects** and **construct a specific version** of the **software**
- **Issues or bug tracking** capability that **enables** the **team** to **record** and **track** the **status of all outstanding issues** associated with **each** configuration **object**

- The SCM repository maintains a change set
- Serves as a collection of all changes made to a baseline configuration
- Used to create a specific version of the software
- Captures all changes to all files in the configuration along with the reason for changes and details of who made the changes and when
- **Few version control systems**

git
Microsoft Visual Studio Team Foundation Server
TortoiseSVN
PERFORCE
Version everything.

Configuration Audit

- **Configuration auditing** is an **SQA activity** that **helps** to ensure that **quality** is **maintained** as **changes** are made.
- It complements the **formal technical review** and is **conducted** by the **SQA group**
- **It addresses the following questions**
- Has a **formal technical review** been **conducted** to assess technical **correctness**?
- Has the **software process been followed** and have **software** engineering **standards** been properly **applied**?
- Has the **change** been "**highlighted**" and "**documented**" in the **SCI**? Have the **change data** and **change author** been **specified**? Do the attributes of the configuration object reflect the change?

Configuration Audit Cont.

- Have SCM procedures for noting the change, recording it and reporting it been followed?
- Have all related SCIs been properly updated?
- A configuration audit ensures that:
- The correct SCIs (by version) have been incorporated into a specific build
- That all documentation is up-to-date and consistent with the version that has been built

Status Reporting

- **Configuration status reporting (CSR)** is also called **status accounting**
- Provides **information about** each **change** to those **personnel** in an **organization** with a **need to know**
- **Answers what happened, who did it, when did it** happen and **what else will be affected**?

Sources of **entries** for configuration status reporting

- **Each time a SCI is assigned** new or updated information
- **Each time a configuration audit** is conducted

The configuration status report

- **Placed** in an **on-line database** or on a **website** for s**oftware developers** and **maintainers** to read
- **Given** to **management** and **practitioners** to keep them appraised of important changes to the project SCIs

Assignment

1	**What is role of software maintenance in software product?**
2	Explain Reengineering.
3	Write a short note on: Reverse Engineering.
4	Explain Software Configuration Management (SCM).