Unit 5 – Maintenance

Software Maintenance

Learning Outcomes

- Understand the software as an evolutionary entity and recognize the need for maintenance.
- Classify maintenance and estimate the cost of maintenance.
- Analyze the features of software re-engineering and reverse engineering.
- Understand Software Project Management.
- Explain configuration management activities.
- Identify the primary reasons for using CASE tools.

Need of Software Maintenance

- Software maintenance is necessary for making changes to a software product after its delivery to the client.
- Every software product needs maintenance once it is in use.
- Hardware, the components, and the interface of the software need revision if the platform is changed or new.
- Enhanced operating system requires the lower component of the software to be advanced.

Need of Software Maintenance

- Necessity to fix the errors due to regular or extensive usage by the client.
- The software changes due to business status changes or users' expectations.
- Software evolution refers to the modification of the characteristics of a software product over generations.

Software Evolution

- The reasons for software evolution are changes in the requirements of the client's business.
- Defects in the existing system or changes in the system environment results in software evolution.
- Some parts of the software may create errors such as incorrect operations or computation.
- Some of the non-functional requirements may get neglected in the developed software.

Software Evolution

- Software evolution is a must to respond to the demands of change.
- Software evolution is essential as organizations invest a large amount of money in a business and they would prefer modification of the existing system rather than developing a new system from the scratch.

Software Maintenance

- Software maintenance is a general process of modifying a software product after it is delivered to the client. The modifications or changes may be necessary for the following reasons:
- To correct coding bugs.
- To accommodate major changes in the design.
- To manage the new requirements.
- To migrate a legacy software.

Types of Software Maintenance

Types of Maintenance

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

Corrective Maintenance

- Corrective maintenance is related to correcting the faults and errors of a software product.
- After the system is put in use, the customers give the reports of the bugs if any.
- These errors may be in the requirements, design, or coding of a system.

Adaptive Maintenance

- Adaptive maintenance is used when the software environment changes.
- For example, a customer may use a new operating system, or a new web or mobile app-based platform to run the product.
- Adaptive maintenance is necessary when the new hardware is included in the existing system.

Perfective Maintenance

- Perfective maintenance is required when the requirements and the features of the system are advanced.
- The user may realize some new requirements or features that evolve the developed software.
- Removing unwanted features and enhancing the system with the user's experience are the goals of this type of maintenance.

Preventive Maintenance

- Future problems in the software are prevented using some modifications and updates in preventative software maintenance.
- It helps increase the lifetime of the software.
- The software code is optimized and prepared to handle the changes in the environment.
- It addresses the stability and maintainability features of the software.

Reverse Engineering

Software Maintenance

- The maintenance of the software can be efficiently achieved by a good understanding of a system.
- A well-documented legacy system can be easily maintained.
- A reverse engineering process can be used in such situations.

Legacy System

- A legacy system is outdated computing software and/or hardware that is still in use.
- The system still meets the needs it was originally designed for but doesn't allow for growth.
- Windows 7 officially became a legacy operating system in January 2020 after Microsoft stopped security updates and support for it.
- Common Business-Oriented Language or COBOL is still used 55 years after its development.

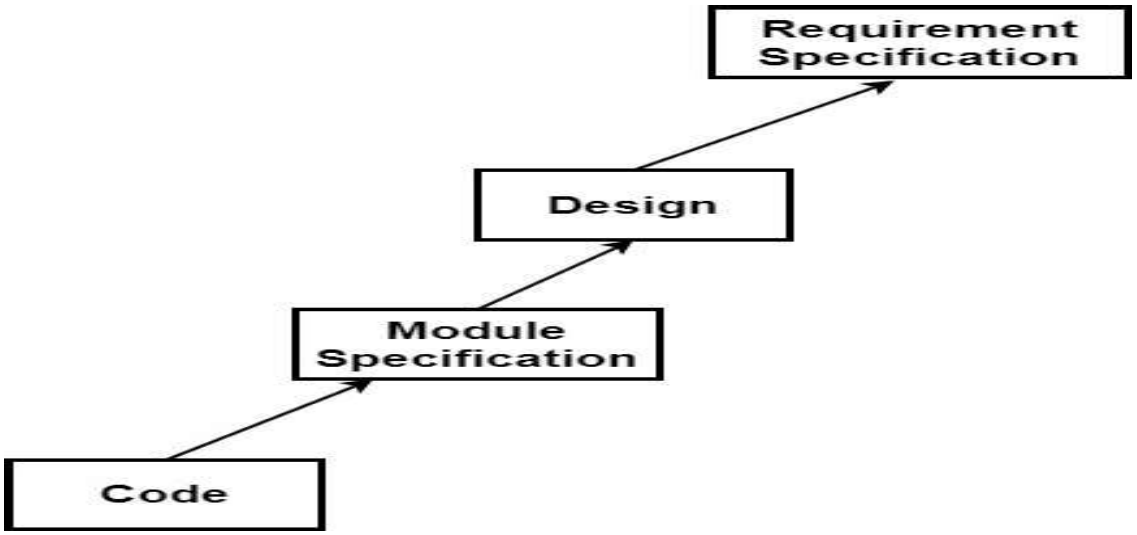
Reverse Engineering

- Several legacy systems are not well structured or there may be wear and tear in the software due to continuous maintenance efforts or lack of documentation.
- Reverse engineering helps in such cases with the maintenance of the software.

Reverse Engineering

- Reverse engineering is the process of recovering the following from the analysis of code.
- Design
- Requirement Specifications
- Functions of a product

Reverse Engineering



Reverse Engineering: Step 1

- The code is improved for better readability, understanding, and structure.
- In many legacy software, the programs are not in proper structure.
- The variable names, and data structures, are given with meaningful names.

Reverse Engineering: Step 1

- The functionality is not changed.
- Complex statements are simplified, and nested loops are replaced with multiple condition statements.
- These changes are known as cosmetic changes.

Reverse Engineering: Steps 2:4

- In the second step, the code is analyzed completely.
- The different modules and their interface is understood.
- In the third step, the design can be extracted using tools such as structure charts, DFDs, and control flow diagrams.
- In the last step, the SRS is written based on the design.

Reverse Engineering Benefits

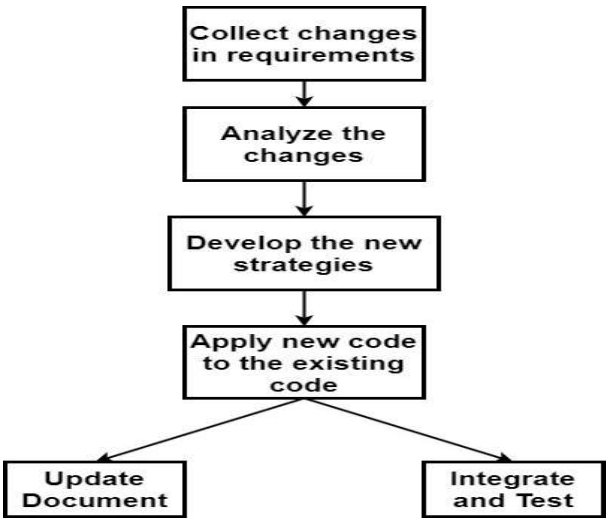
- Understand the complexity.
- Find out the side effects.
- Recover the lost information.
- Create new products.
- Explore reusability.

Software Re-engineering

Software Maintenance Process Model -1

- The software maintenance process models are divided into two categories.
- The first model is used when small changes are required in the code and are documented.
- In this process model, a few team members collect the requirements, analyze and formulate the strategy.
- The existing system is useful for modification and debugging of the errors.

Software Maintenance Process Model -1



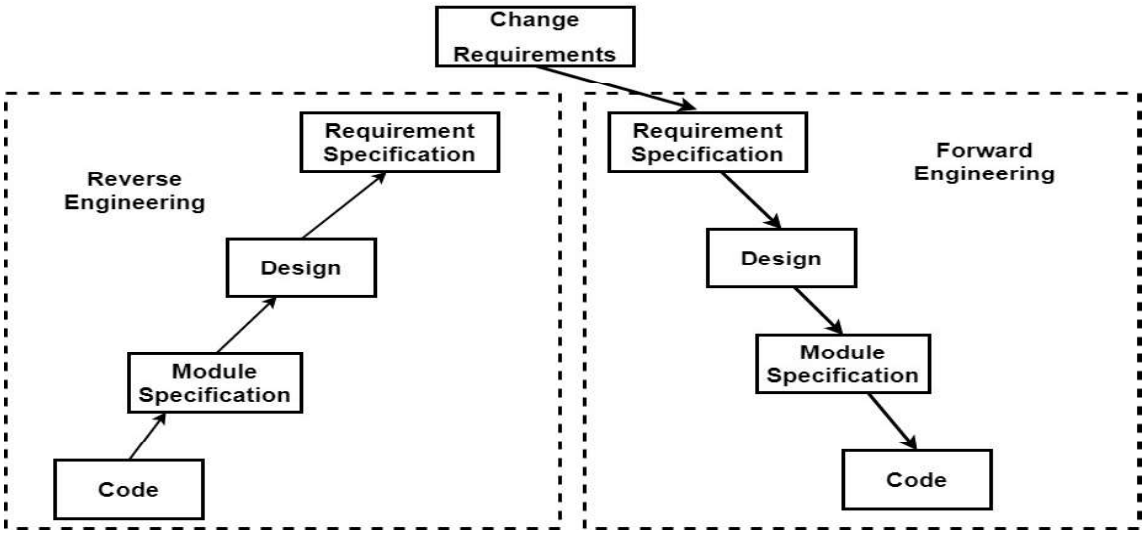
Software Maintenance Process Model 2

- When the requirement changes considerably, the amount of rework is more.
- The modification in the software is done by combining reverse and forward engineering.
- This process is known as software re-engineering.
- The legacy products are reverse engineered to get module specifications.

Software Maintenance Process Model 2

- The specifications are further used to generate the design.
- The design is then explored to produce requirements and new requirements are added.
- Thus, a combination of processes such as software reverse and forward engineering, reconstructing etc is known as software reengineering.

Software Re-engineering



Software Re-engineering

- **Software re-engineering** is a process where the existing software is **reorganized** and **modified** for **efficient maintenance**.
- Helps in **software evolution, adds quality** to product.
- **Identifies** the activities in software maintenance process.
- **Reduces** the **risks,** and **cost** of software development.

Software Re-engineering - Drawbacks

- Software re-engineering keeps challenges in its maintenance.
- Some steps such as data management or modification in software architecture should be done manually.
- Some software products cannot be re-engineered.

Software Maintenance Cost

Software Maintenance Cost

- In 1981, software engineering scientist Boehm proposed a Constructive Cost Model (COCOMO) to determine maintenance costs.
- The model includes annual change traffic (ACT) to determine the quality.
- The ACT is that small subset of software source programs that get changed in a year.

Annual Change Traffic (ACT)

- ACT is determined as follows:
- $ACT = \frac{KLOC_{added} + KLOC_{deleted}}{KLOC_{TOTAL}}$

- $KLOC_{added}$: It is a measure of kilo lines of source code added during maintenance.
- $KLOC_{deleted}$: It is a measure of kilo lines of source code deleted during maintenance.
- $KLOC_{TOTAL}$: It is a measure of total lines of code instructions.
- Code changed = code added + code deleted

Software Maintenance Cost

- The ACT is used to determine the change in the code due to new or modified requirements.
- Maintenance Cost = ACT × Development Cost
- The maintenance cost is always determined approximately as it does not include new hardware and software complexity, experience of the software engineer and the familiarity of the product.

Software Maintenance Cost

- Factors affecting software maintenance cost are:
- Technical: Changes in modules, programming languages and styles, testing, documentation and configuration management.
- Non-Technical: Scope, stable team of people, stability in hardware, external environment etc

Software Configuration Management

Software Configuration

- Software configuration refers to the state of the source code, design, SRS, test reports, user manual, or guide used by software engineers during the SDLC.
- The state of the software product changes during its development and testing.
- The software gets changed due to various reasons.

Reasons for Software Changes

- Release
- Version
- Revision

Software Release

- **Release:** A software is released newly if there is any new bug or new functionality or are small changes in its usage and the technology.
- For example, the addition of new features to correct a particular error of the Windows operating system and a release of the new product.

Software Version

- Version: If there is a major change in the hardware and technology in which the software product is working or the functionality of the product is drastically changed then the software configuration is revised.
- New version of Android updates for supporting new features, Version Windows 95, Windows 2000, Windows XP, Windows 10, etc. are the versions of the Windows Operating system.

Software - Revision

- Revision: If there are minor bugs or errors, then the software is revised in its configuration.
- For example, revision in the typesetting features in the MS-Office
- Thus release, version and revision lead to software configuration management.

Need for Software Configuration Management

Software Configuration

- Software configuration is necessary for software updates and storage.
- When the objects in software are reused by many software engineers, he/she will have their copy.
- If one software makes changes to one object and does not inform others, then there will be inconsistent objects used by many people. This will create bugs in the software.

Software Configuration

- Software bugs due to inconsistent usage can be avoided with proper configuration management in place.
- Also, if the software is modified by several people at the same time, then there will be overwriting of the code by many people disturbing integrity.

Software Configuration

- Consider software with multiple modules managed by many software engineers.
- If one developer is trying to integrate the modules, whereas it has been modified by another engineer.

Software Configuration

- Configuration management provides a freeze option with which the environment for the software development is stable.
- The different variants of the modules are archived in the configuration management and the respective version is given to the developer.

Software Configuration

- Software configuration management maintains the status of the software product and keeps account of the changes in the software.
- The configuration management activities are done by the project manager using automatic tools.
- The configuration activities are performed in the software configuration process (SCP).

Software Configuration Management Process (SCM)

SCM Process

- Configuration identification of the objects
- Controlling versions
- Controlling changes
- Audit configuration
- Generating status report

Configuration Identification

- In this step, the objects in the software are classified into three categories: controlled, pre-controlled and uncontrolled.
- In the controlled category, the objects of the software are already in a configuration.
- Typical controllable objects are requirement specifications, design, source code, test cases, reports, etc.

Configuration Identification

- The pre-controlled objects are those which will be configured in the future.
- Uncontrolled objects are not and will not be subjected to configuration control.

Version Control

- Different tools and methods are used in version control to create different versions of the objects.
- Each object gets a different configuration with different features or attributes.

Change Control

- In the change control step, a request for a variation or modification to the existing object is received.
- How this change creates an impact on the objects and the functions of the system, modification in the cost of the system are evaluated.
- The change control authority (CCA) is a single person or team of people who authorize the request, and they decide whether the change is permitted in the object.

Audit Configuration

- After completing the configuration, the SCM generates audit reports which check the original requirements are fulfilled in the new deliverable product.
- This is done in the configuration audit step.

Generating status report

- The different stakeholders of the project such as the software developer, tester, and customers get a detailed report of the current configuration of the software.
- The frequently asked questions (FAQs), manual, instruction or installation guide, necessary software, and hardware requirements for installation of the software are given in the status reports.

CASE Tools

CASE Tools

- CASE stands for Computer-Aided Software Engineering.
- CASE tools provide automated support in software engineering.
- The different activities such as gathering requirements, preparing software design, developing code, and testing can be performed with the help of CASE tools.

CASE Tools

- The software configuration and project management are also supported by CASE tools.
- The advantages of using CASE tools are they can be used at a low cost.
- There is no development from the scratch for developing quality software with CASE tools.

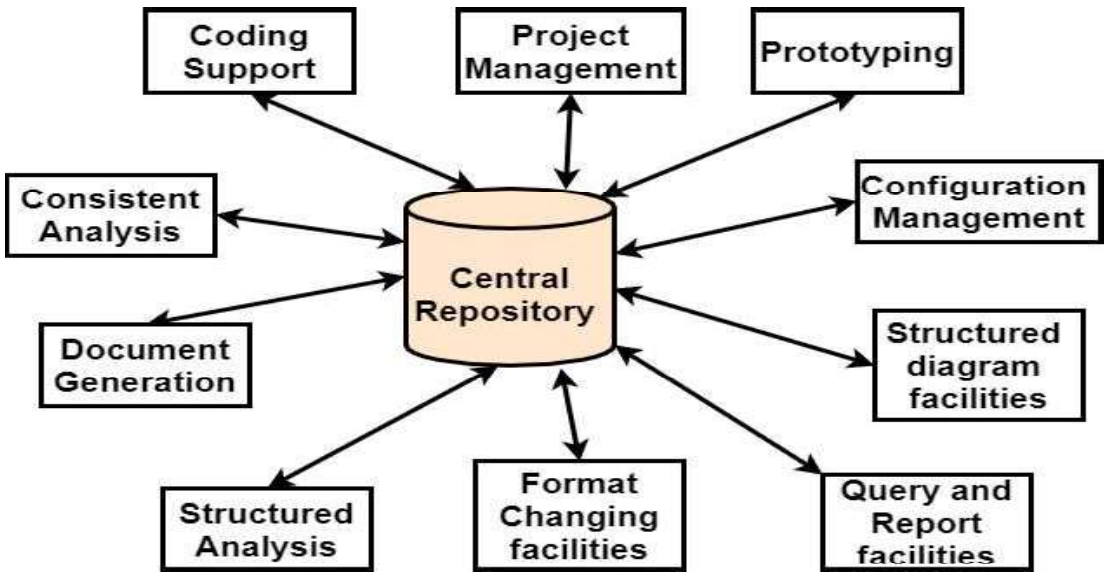
CASE Tools

- The productivity of the software development gets improved and are collectively stored as toolkits in a common environment.
- The tools with common information are integrated into a central repository.
- For example, a data dictionary can be used as a CASE tool for storing elementary and composite data.

CASE Tools

- The CASE environment is different from than programming environment.
- In the programming environment, the tools are available only for coding e.g., Java Virtual Machine supporting Java on different platforms, databases, web, etc.
- The CASE tools environment is for all the phases of SDLC.

CASE Tools



Prototyping CASE Tools

- Prototyping CASE tools are used for requirements.
- Prototyping CASE tools are used to understand user interaction, data storage, and retrieval, the control flow in the program, data processing at different steps, etc. during requirement collection.

CASE Tool for Structured Analysis and Design

- Structured analysis and design CASE tools are used for drawing software design diagrams with fewer efforts.
- Many hierarchical levels can be analyzed using CASE tools.
- The tools must support easy navigation and consistency during the design.

CASE Tool for Structured Analysis and Design

- For e.g., flow chart maker tools, UML diagram tools for use-case diagrams, sequence diagrams, etc. in object-oriented design.

Code generation and CASE tools

- CASE tools cannot be directly used to write code.
- The commonly used programming languages use CASE tools for database tables, interfaces, records, or the outer skeleton of the customized software to be developed.
- Eclipse and Cscope are used in searching code in C, Adobe Edge Inspect tool is used in web page development.

Test case generation using CASE tools

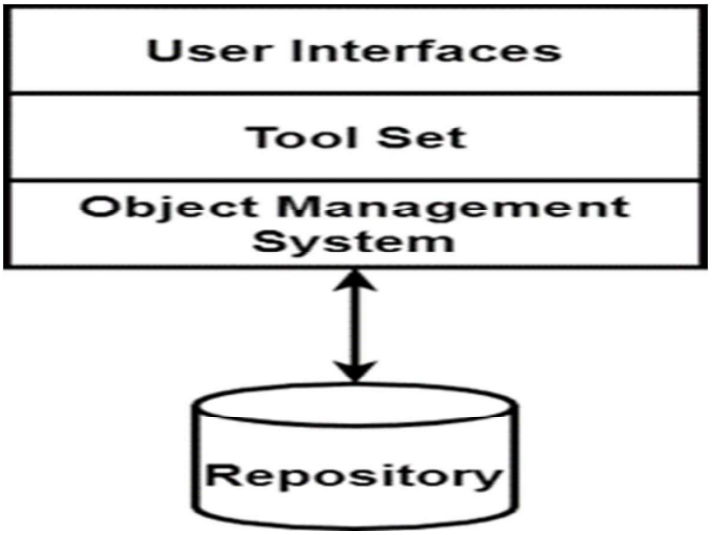
- CASE tools must support testing of requirements and design.
- The test plan generation must be supported by the CASE tools.
- The SoapTest is a CASE tool that allows users to validate the functions, unit testing, and functional testing.
- SoapTest rapidly creates regression tests.

CASE Tools Architecture

CASE Tools Architecture

- The architecture of a typical modern CASE environment consists of three components. These components are connected to the central repository.
- User interface
- Toolset
- Object management system (OMS)

CASE Tools Architecture



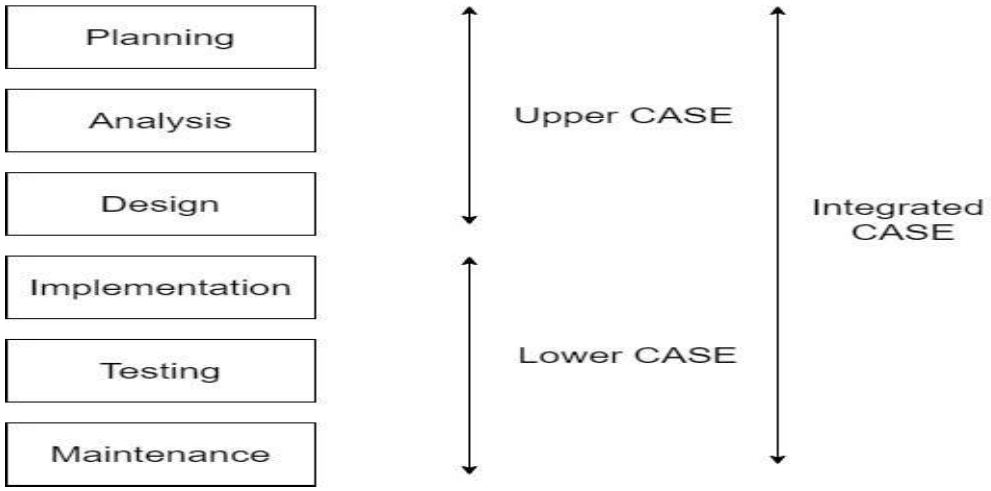
CASE Tools Architecture

- Users can learn how to access the different tools with the help of the interface.
- This helps in reducing the overhead of learning efforts for a CASE tool.
- OMS is used to map the different entities of software such as requirements, design, code, text data, and project plans to the storage management system or repository.

CASE Tools Architecture

- For e.g., in the relational database management system, the relationships between entities, attribute declaration, initialization, types of relationships, etc. can be mapped using the OMS.
- The repository consists of different CASE tools used in the requirements, design, coding, and testing phases of software.

CASE Tools Classification



CASE Tools Classification

- Case tools can be divided based on the components given below.
- Upper Case Tools - Upper CASE tools are used in requirement collection, planning, analysis, and design stages of SDLC.
- Lower Case Tools - Lower CASE tools are used in the coding, testing, and maintenance phases of SDLC.

CASE Tools Classification

- **Integrated Case Tools - Integrated CASE tools are useful in all the stages such as requirement specification, design, coding, testing, and documentation.**

CASE Tools : Advantages and Drawbacks

CASE Tools : Advantages

- CASE tools are beneficial in cost-saving at different phases of SDLC.
- CASE tools improve speed and reduce time for any software development activity.
- Preparing software design manually is time consuming and tedious. CASE tools help in consistent and complete designing of a software.
- The cost of development can be saved up to 30 to 40%.

CASE Tools : Advantages

- The cost of development can be saved up to 30 to 40%.
- As the CASE tools are already undergone testing, the quality of work is better than any newly developed program.
- The CASE tools are useful in generating consistent documentation.
- There are fewer chances of having a human error due to the central repository.

CASE Tools : Advantages

- Preparing software design using CASE tools is easier than the manual or other geometric methods.
- The different phases of SDLC are systematic and consistent due to the usage of the CASE tools.
- This saves a lot of maintenance costs.
- The overall working of the SDLC is structured and systematic with the help of CASE tools.

CASE Tools: Drawbacks

- The staff using the CASE tools must be given training on using the tool for the respective phase of the SDLC.
- In many situations, staff will resist the use of the CASE tool.
- The size of the project may not be suitable for the CASE tool in certain cases.
- The level of quality using CASE tools may not be desirable in certain applications.