Unit 5

Software Engineering | Software Maintenance

Software Maintenance is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software applications after delivery to correct faults and to improve performance.

Need for Maintenance –
Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.

Challenges in Software Maintenance:

The various challenges in software maintenance are given below:

- The popular age of any software program is taken into consideration up to ten to fifteen years. As software program renovation is open ended and might maintain for decades making it very expensive.
- Older software program's, which had been intended to paintings on sluggish machines with much less reminiscence and garage ability can not maintain themselves tough in opposition to newly coming more advantageous software program on contemporary-day hardware.
- Changes are frequently left undocumented which can also additionally reason greater conflicts in future.
- As era advances, it turns into high priced to preserve vintage software program.
- Often adjustments made can without problems harm the authentic shape of the software program, making it difficult for any next adjustments.

Process of Software Maintenance:

Software Maintenance is an important phase of Software Development Life Cycle (SDLC), and it is implemented in the system through a proper software maintenance process, known as **Software Maintenance Life Cycle (SMLC)**. This life cycle consists of seven different phases, each of which can be used in iterative manner and can be extended so that customized items and processes can be included. These seven phases of Software Maintenance process are:

1. **Identification Phase:**
   In this phase, the requests for modifications in the software are identified and analysed. Each of the requested modification is then assessed to determine and classify the type of maintenance activity it requires. This is either generated by the system itself, via logs or error messages, or by the user.

2. **Analysis Phase:**

The feasibility and scope of each validated modification request are determined and a plan is prepared to incorporate the changes in the software. The input attribute comprises validated modification request, initial estimate of resources, project documentation, and repository information. The cost of modification and maintenance is also estimated.

3. **Design Phase:**
   The new modules that need to be replaced or modified are designed as per the requirements specified in the earlier stages. Test cases are developed for the new design including the safety and security issues. These test cases are created for the **validation and verification** of the system.

4. **Implementation Phase:**
   In the implementation phase, the actual modification in the software code are made, new features that support the specifications of the present software are added, and the modified software is installed. The new modules are coded with the assistance of structured design created in the design phase.

5. **System Testing Phase:**
   Regression testing is performed on the modified system to ensure that no defect, error or bug is left undetected. Furthermore, it validates that no new faults are introduced in the software as a result of maintenance activity. **Integration testing** is also carried out between new modules and the system.

6. **Acceptance Testing Phase:**
   Acceptance testing is performed on the fully integrated system by the user or by the third party specified by the end user. The main objective of this testing is to verify that all the features of the software are according to the requirements stated in the modification request.

7. **Delivery Phase:**
   Once the acceptance testing is successfully accomplished, the modified system is delivered to the users. In addition to this, the user is provided proper consisting of manuals and help files that describe the operation of the software along with its hardware specifications. The final testing of the system is done by the client after the system is delivered.

Categories of Software Maintenance –

1. **Corrective maintenance:**
   Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

2. **Adaptive maintenance:**
   This includes modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

3. **Perfective maintenance:**
   A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

4. **Preventive maintenance:**
   This type of maintenance includes modifications and updations to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

Reverse Engineering –

Reverse Engineering is processes of extracting knowledge or design information from anything man-made and reproducing it based on extracted information. It is also called back Engineering.

Software Reverse Engineering –

Software Reverse Engineering is the process of recovering the design and the requirements specification of a product from an analysis of it's code. Reverse Engineering is becoming important, since several existing software products, lack proper documentation, are highly unstructured, or their structure has degraded through a series of maintenance efforts.

Why Reverse Engineering?

- Providing proper system documentation.
- Recovery of lost information.
- Assisting with maintenance.
- Facility of software reuse.
- Discovering unexpected flaws or faults.

Used of Software Reverse Engineering –

- Software Reverse Engineering is used in software design, reverse engineering enables the developer or programmer to add new features to the existing software with or without knowing the source code.
- Reverse engineering is also useful in software testing, it helps the testers to study the virus and other malware code .

Software Reliability

Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Software Failure Mechanisms

The software failure can be classified as:

Transient failure: These failures only occur with specific inputs.

Permanent failure: This failure appears on all inputs.

Recoverable failure: System can recover without operator help.

Unrecoverable failure: System can recover with operator help only.

Non-corruption failure: Failure does not corrupt system state or data.

Corrupting failure: It damages the system state or data.

Software failures may be due to bugs, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems.

Software Quality Assurance

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.
Generally the quality of the software is verified by the third party organization like international standard organizations.

Elements Of Software Quality Assurance:

1. **Standards:** The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. The job of SQA is to ensure that standards that have been adopted are followed an all work products conform to them.
2. **Reviews and audits:** Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel(people employed in an organization) with the intent of ensuring that quality guidelines are being followed for software engineering work.
3. **Testing:** Software testing is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted for primary goal of software.
4. **Error/defect collection and analysis**: SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.
5. **Change management:** SQA ensures that adequate change management practices have been instituted.
6. **Education:** Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement which is key proponent and sponsor of educational programs.
7. **Security management:** SQA ensures that appropriate process and technology are used to achieve software security.
8. **Safety:** SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
9. **Risk management**: The SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

Software quality assurance focuses on:

- software's portability
- software's usability
- software's reusability
- software's correctness
- software's maintainability
- software's error control

ISO 9000:

It is a set of International Standards on quality management and quality assurance developed to help companies effectively document the quality system elements needed to an efficient quality system.

SEICMM:
SEI (Software Engineering Institute), Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization.
Difference between ISO9000 and SEI-CMM:

ISO 9000	SEICMM
ISO 9000 is a set of international standards on quality management and quality assurance developed to help companies effectively document the quality system elements needed to an efficient quality system.	SEI (Software Engineering Institute), Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization.
Focus is customer supplier relationship, attempting to reduce customer's risk in choosing a supplier.	Focus on the software supplier to improve its interval processes to achieve a higher quality product for the benefit of the customer.
It is created for hard goods manufacturing industries.	It is created for software industry.
ISO9000 is recognized and accepted in most of the countries.	SEICMM is used in USA, less widely elsewhere.
It specifies concepts, principles and safeguards that should be in place.	CMM provides detailed and specific definition of what is required for given levels.
This establishes one acceptance level.	It assesses on 5 levels.
Its certification is valid for three years.	It has no limit on certification.
It focuses on inwardly processes.	It focus outwardly.
It has no level.	It has 5 levels: **(a).** Initial **(b).** Repeatable **(c).** Defined **(d).** Managed **(e).** Optimized
It is basically an audit.	It is basically an appraisal.

ISO 9000	SEICMM
It is open to multi sector.	It is open to IT/ITES.
Follow set of standards to make success repeatable.	It emphasizes a process of continuous improvement.

Software Re-engineering is a process of software development which is done to improve the maintainability of a software system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form.

Computer aided software engineering (CASE) is the implementation of computer facilitated tools and methods in software development. CASE is used to ensure a high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers and others to see the project milestones during development.

CASE Tools:
The essential idea of CASE tools is that in-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes. Throughout the 1990, CASE tool became part of the software lexicon, and big companies like IBM were using these kinds of tools to help create software.
Various tools are incorporated in CASE and are called CASE tools, which are used to support different stages and milestones in a software development life cycle.

Types of CASE Tools:

1. **Diagramming Tools:**
   It helps in diagrammatic and graphical representations of the data and system processes. It represents system elements, control flow and data flow among different software components and system structure in a pictorial form.
   For example, Flow Chart Maker tool for making state-of-the-art flowcharts.

2. **Computer Display and Report Generators:**
   It helps in understanding the data requirements and the relationships involved.

3. **Analysis Tools:**
   It focuses on inconsistent, incorrect specifications involved in the diagram and data flow. It helps in collecting requirements, automatically check for any irregularity, imprecision in the diagrams, data redundancies or erroneous omissions.
   For example,
   - (i) Accept 360, Accompa, CaseComplete for requirement analysis.
   - (ii) Visible Analyst for total analysis.

4. **Central Repository:**
   It provides the single point of storage for data diagrams, reports and documents related to project management.

5. **Documentation Generators:**
   It helps in generating user and technical documentation as per standards. It creates documents for technical users and end users.
   For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

6. **Code Generators:**
   It aids in the auto generation of code, including definitions, with the help of the designs, documents and diagrams.

Advantages of the CASE approach:

- As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.

- The overall quality of the product is improved as an organized approach is undertaken during the process of development.

- Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.

- CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

Disadvantages of the CASE approach:

- **Cost:** Using case tool is a very costly. Mostly firms engaged in software development on a small scale do not invest in CASE tools because they think that the benefit of CASE are justifiable only in the development of large systems.
- **Learning Curve:** In most cases, programmers productivity may fall in the initial phase of implementation , because user need time to learn the technology. Many consultants offer training and on-site services that can be important to accelerate the learning curve and to the development and use of the CASE tools.
- **Tool Mix:** It is important to build an appropriate selection tool mix to urge cost advantage CASE integration and data integration across all platforms is extremely important.