Software Maintenance

Software maintenance is a part of the Software Development Life Cycle. Its primary goal is to modify and update software application after delivery to correct errors and to improve performance. Software is a model of the real world. When the real world changes, the software require alteration wherever possible.

Software Maintenance is an inclusive activity that includes error corrections, enhancement of capabilities, deletion of obsolete capabilities, and optimization.

Need for Maintenance

- Correct errors
- Change in user requirement with time
- Changing hardware/software requirements
- To improve system efficiency
- To optimize the code to run faster
- To modify the components
- To reduce any unwanted side effects.

Thus the maintenance is required to ensure that the system continues to satisfy user requirements.

Types of Software Maintenance

1. Corrective Maintenance

Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.

2. Adaptive Maintenance

It contains modifying the software to match changes in the ever-changing environment.

3. Preventive Maintenance

It is the process by which we prevent our system from being obsolete. It involves the concept of reengineering & reverse engineering in which an old system with old technology is re-engineered using new technology. This maintenance prevents the system from dying out.

4. Perfective Maintenance

It defines improving processing efficiency or performance or restricting the software to enhance changeability. This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.

Reverse Engineering

Software reverse engineering is the process of analyzing the implementation and working of a software application or system without directly referencing the source code. It involves examining the machine-level code from the application executable files for understanding the software application.

- **Understanding proprietary systems:** Reverse engineering is useful for understanding the functionality of a software application without having direct access to the source code or in the absence of documentation. It helps you gain crucial information related to the software's logic and implementation.

- **Software debugging and maintenance:** Reverse engineering can help you in identifying and fixing software bugs. By analyzing the code (whitebox reverse engineering), you can pinpoint problems and work on their solutions.

- **Security analysis:** Vulnerabilities in software systems can be identified by the process of reverse engineering. Security experts can analyze the software and recommend changes based on their analysis to protect it against potential threats.

- **Competitor analysis:** Reverse engineering is also used for analyzing and understanding the techniques and algorithms used for developing software systems by competitors.

This provides insights into their software design and optimizations techniques.

- **Legal and Regulatory Compliance:** This involves ensuring that a business or organization adheres to the laws and regulations relevant to its industry or geographic location. Compliance efforts are aimed at avoiding legal issues, fines, and other penalties. Reverse engineering can be used to ensure products or systems comply with specific industry standards or legal requirements.

- **Hardware Analysis:** Hardware analysis involves a detailed examination of the physical components and structures of computer hardware, devices, or equipment. Reverse engineering can be used to understand the inner workings of hardware, diagnose issues, and improve or customize hardware systems.

- **Customization and Modification:** Customization and modification entail making changes to a product or system to better align it with specific requirements, preferences, or use cases. Reverse engineering can help in adapting existing products or systems to meet unique needs, enhancing their functionality or appearance.

- **Legacy System Migration:** Legacy system migration involves transitioning from older, often outdated, systems to more modern and efficient solutions. Reverse engineering may be used to analyze and understand legacy systems to facilitate a smooth migration process to newer technology.

- **Recovery of Lost Data:** Data recovery is the process of retrieving lost, deleted, or inaccessible data from storage media or systems. Reverse engineering can be used to reverse data corruption or loss, making it possible to recover critical information.

- **Intellectual Property Protection:** Intellectual property protection involves safeguarding creative works and innovations, such as patents, trademarks, and copyrights, from unauthorized use or duplication. Reverse engineering is used to identify potential infringements on intellectual property rights and take appropriate legal actions to protect these assets.

- **Product Improvement:** Product improvement refers to enhancing the quality, functionality, or performance of existing products. Reverse engineering can be applied to analyze a product's design and features, identify areas for enhancement, and create improved versions.

Software Quality Assurance

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.
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.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating software.
7. Improves the quality of the software.
8. It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs

Software Reliability

Software Reliability means **Operational reliability**. It is described as the ability of a system or component to perform its required functions under static conditions for a specific period.

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 Reliability is an essential connect of software quality, composed with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software Reliability is hard to achieve because the complexity of software turn to be high. While any system with a high degree of complexity, containing software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the speedy growth of system size and ease of doing so by upgrading the software.

What is Software Quality?

Software Quality shows how good and reliable a product is. To convey an associate degree example, think about functionally correct software. It performs all functions as laid out in the SRS document. But, it has an associate degree virtually unusable program. even though it should be functionally correct, we tend not to think about it to be a high-quality product. Another example is also that of a product that will have everything that the users need but has an associate degree virtually incomprehensible and not maintainable code. Therefore, the normal construct of quality as "fitness of purpose" for code merchandise isn't satisfactory.

Factors of Software Quality

1. **Portability:** A software is claimed to be transportable, if it may be simply created to figure in several package environments, in several machines, with alternative code merchandise, etc.
2. **Usability:** A software has smart usability if completely different classes of users (i.e. knowledgeable and novice users) will simply invoke the functions of the merchandise.
3. **Reusability:** A software has smart reusability if completely different modules of the merchandise will simply be reused to develop new merchandise.
4. **Correctness:** Software is correct if completely different needs as laid out in the SRS document are properly enforced.
5. **Maintainability:** A software is reparable, if errors may be simply corrected as and once they show up, new functions may be simply added to the merchandise, and therefore the functionalities of the merchandise may be simply changed, etc

6. **Reliability:** Software is more reliable if it has fewer failures. Since software engineers do not deliberately plan for their software to fail, reliability depends on the number and type of mistakes they make. Designers can improve reliability by ensuring the software is easy to implement and change, by testing it thoroughly, and also by ensuring that if failures occur, the system can handle them or can recover easily.
7. **Efficiency.** The more efficient software is, the less it uses of CPU-time, memory, disk space, network bandwidth, and other resources. This is important to customers in order to reduce their costs of running the software, although with today's powerful computers, CPU time, memory and disk usage are less of a concern than in years gone by.

ISO 9000 Certification

ISO (International Standards Organization) is a group or consortium of 63 countries established to plan and fosters standardization. ISO declared its 9000 series of standards in 1987. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the guidelines for maintaining a quality system. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

Types of ISO 9000 Quality Standards

The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.

1. **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
2. **ISO 9002:** This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from

external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.

3. **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

ISO9000

ISO9000 is an international standard of quality management and quality assurance. It certifies the companies that they are documenting the quality system elements which are needed to run a efficient and quality system.

SEI-CMM

SEI (Software Engineering Institute) - Capability Maturity Model (CMM) is specifically for software organizations to certify them at which level, they are following and maintaining the quality standards.

Following are the important differences between ISO9000 and SEI-CMM---

Key	ISO9000	SEI-CMM.
Definition	ISO9000 is an international standard of quality management and quality assurance. It certifies the companies that they are documenting the quality system elements which are needed to run a efficient and quality system.	SEI-CMM is specifically for software organizations to certify them at which level, they are following and maintaining the quality standards.
Focus	Focus of ISO9000 is on customer supplier relationship, and to reduce the customer's risk.	Focus of SEI-CMM is to improve the processes to deliver a quality software product to the customer.
Target Industry	ISO9000 is used by manufacturing industries.	SEI-CMM is used by software industry.
Recognition	ISO9000 is univesally accepted accross lots of countries.	SEI-CMM is mostly used in USA.
Guidelines	ISO9000 guides about concepts, priciples and safeguards to be in place in a workplace.	SEI-CMM specifies what is to be followed at what level of maturity.
Levels	ISO9000 has one acceptance level.	SEI-CMM has five acceptance levels.
Validity	ISO9000 certificate is valid for three years.	SEI-CMM certificate is valid for three years as well.
Level	ISO9000 has no levels.	SEI-CMM has five levels, Initial, Repeatable, Defined, Managed and Optimized.
Focus	ISO9000 focuses on following a set of standards so that firms delivery are successful everytime.	SEI-CMM focuses on improving the processes.

Computer Aided Software Engineering (CASE)

Computer-aided software engineering (CASE) is the implementation of computer-facilitated tools and methods in software development. CASE is used to ensure 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 can also help as a warehouse for documents related to projects, like business plans, requirements, and design specifications. One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process.

Types of CASE Tools:

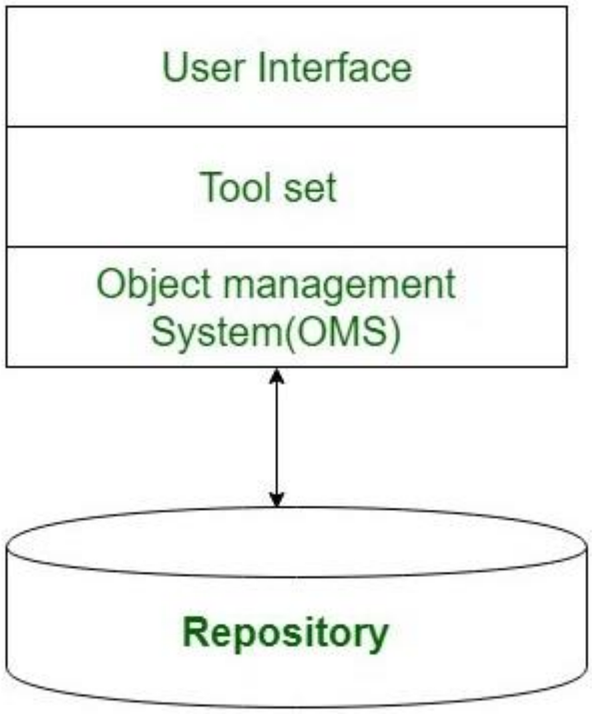
1. **Diagramming Tools:** It helps in diagrammatic and graphical representations of the data and system processes. It represents system elements, control flow and data flow among different software components and system structures in a pictorial form. For example, Flow Chart Maker tool for making state-of-the-art flowcharts.
2. **Computer Display and Report Generators:** These help 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 a 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 designs, documents, and diagrams.

7. **Tools for Requirement Management:** It makes gathering, evaluating, and managing software needs easier.
8. **Tools for Analysis and Design**: It offers instruments for modelling system architecture and behaviour, which helps throughout the analysis and design stages of software development.
9. **Tools for Database Management:** It facilitates database construction, design, and administration.
10. **Tools for Documentation:** It makes the process of creating, organizing, and maintaining project documentation easier.

Advantages of the CASE approach:

- **Improved Documentation:** Comprehensive documentation creation and maintenance is made easier by CASE tools. Since automatically generated documentation is usually more accurate and up to date, there are fewer opportunities for errors and misunderstandings brought on by out-of-current material.
- **Reusing Components:** Reusable component creation and maintenance are frequently facilitated by CASE tools. This encourages a development approach that is modular and component-based, enabling teams to shorten development times and reuse tested solutions.
- **Quicker Cycles of Development:** Development cycles take less time when certain jobs, such testing and code generation, are automated. This may result in software solutions being delivered more quickly, meeting deadlines and keeping up with changing business requirements.
- **Improved Results**: Code generation, documentation, and testing are just a few of the time-consuming, repetitive operations that CASE tools perform. Due to this automation, engineers are able to concentrate on more intricate and imaginative facets of software development, which boosts output.
- **Achieving uniformity and standardization:** Coding conventions, documentation formats and design patterns are just a few of the areas of software development where CASE tools enforce uniformity and standards. This guarantees consistent and maintainable software development.

Architecture of a CASE Environment



1. User Interface

The user interface provides a regular framework for accessing the various tools so creating it easier for the users to act with the different tools and reducing the overhead of learning however the different tools are used.

2. Object Management System (OMS) and Repository

Different case tools represent the product as a group of entities like specification, design, text data, project arrange, etc. the thing management system maps these logical entities such into the underlying storage management system (repository). The industrial on-line database management systems are meshed towards supporting giant volumes of data structured as straightforward comparatively short records. There are some forms of entities however sizable amount of instances. In contrast, CASE tools produce an oversized range of entity and relation varieties with maybe some instances of every. So the thing management system takes care of befittingly mapping into the underlying storage management system.