Software Maintenance

BASICS OF SOFTWARE MAINTENANCE

IEEE defines maintenance as :

'A process of modifying a software system or component after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment'.

- In software engineering a software needs to be 'serviced' so that it is able to meet the changing environment (such as business and user needs) where it functions.

IMPORTANCE OF SOFTWARE MAINTENANCE

- **Providing Continuity of Service**
  Fixing errors, recovering from failures, such as hardware failures or incompatibility of hardware with software, and accommodating changes in the operating system and the hardware.

- **Supporting Mandatory Upgrades**
  Due to changes in government regulations or students to maintain competition with other software that exist in the same category.

- **Improving the Software to Support User Requirements**
  To enhance functionality in a software, to improve performance .

- **Facilitating Future Maintenance Work**
  Include restructuring of the software code and database used in the software.

Changing a Software System

- Software maintenance and evolution of system was first introduced by Lehman and Belady in 1974.

- One of the key observations of the studies was that large system are never complete and continue to evolve and as these system evolve, they become more complex unless some actions are taken to reduce the complexity.

- Lehman stated eight laws for software maintenance and evolution of large systems.

Lehman Laws for software maintenance and evolution of large systems

Law	Description
Law of continuing change	A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment.
Law of increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Law of conservation of familiarity	For E-Systems to efficiently continue to evolve a deep understanding of how the system functions, and why it has been developed to function in that manner, must be preserved at all costs. The incremental change in each release over the life time of the system is approximately constant.
Law of conservation of organizational stability	Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.

Lehman Laws for software maintenance and evolution of large systems

Law	Description
Law of self regulation	Global E-type system evolution processes are self-regulating, and distribution of product and process measures close to normal.
Law of continuing growth	E-type system's functional capability must be continually enhanced to maintain user satisfaction over system lifetime, BUT expansion of the system size can have negative affects to the ability to be comprehended along with its ability to evolve.
Law of declining quality	Poorly modified systems lead to introduction of defects; & The quality of E-type systems will appear to be declining as newer products emerge.
Law of feedback system	To sustain continuous change or evolution, & to minimize threats of software decay & loss of familiarity, feedback to monitor the performance is must. Feedback helps to collect metrics on the systems and maintenance efforts performance.

Components of Software Maintenance Framework

Components	Features
User requirements	•Request for additional functionality, error correction, capability and improvement in maintainability. • Request for non-programming related support.
Organizational Environment	• Change in business policies. • Competition in market.
Operational Environment	• Hardware platform. • Software specifications.

Components of Software Maintenance Framework

Components	Features
Maintenance Process	• Capturing requirements • Variation in programming and working practices. • Paradigm shift. • Error detection and correction.
Software Product	• Quality of documentation. • Complexity of programs. • Program Structure.
Software Maintenance team	• Staff turnover. • Domain expertise.

Factors Affecting Software Maintenance

- Relationship of Software product and Environment
- Relationship of Software product and User
- Relationship of Software product and Software Maintenance team

- **Software Maintenance Team:**

  Various functions performed by the Software Maintenance team are:

  - Locating information in system documentation
  - Keeping system documentation up-to-date
  - Extending existing functions to accommodate new or changing requirements
  - Adding new functions to the system
  - Finding the source of system failures or problems
  - Managing change to the system as they are made.

- **The aspects of a maintenance team that lead to high maintenance costs are:**

  - Staff turnover
  - Domain expertise

TYPES OF SOFTWARE MAINTENANCE

- Corrective maintenance is concerned with fixing reported errors in the software.
- Adapting maintenance means changing the software to some new environment, such as adapting a new version of an operating system.
- Perfective maintenance involves implementing new functional or non-functional requirements.
- Preventive maintenance involves implementing, changes to prevent occurrence of errors.

Challenges in Software Maintenance:

- Maintaining software is though considered essential these days, it is not a simple procedure and entails extreme efforts. The process requires knowledgeable experts who are well versed in latest software engineering trends and can perform suitable programming and testing. Furthermore, the programmers can face several challenges while executing software maintenance which can make the process time consuming and costly. Some of the challenges encountered while performing software maintenance are:
  - Finding the person or developer who constructed the program can be difficult and time consuming.
  - Changes are made by an individual who is unable to understand the program clearly.
  - The systems are not maintained by the original authors, which can result in confusion and misinterpretation of changes executed in the program.
  - Information gap between user and the developer can also become a huge challenge in software maintenance.
  - The biggest challenge in software maintenance is when systems are not designed for changes.

Software Maintenance Life Cycle

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:
  - **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.
  - **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.

Process of Software Maintenance

- **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.
  - **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.
  - **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
    - 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

- **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.

SMLC

Phases	Input	Process	Control	Output
Problem identification phases	• Modification request	•Assign change number •Classify modification request • Accept or reject change • Prioritize	• Uniquely identified modification request • Enter modification request in repository	• Validated modification request • Validated Process determinations
Analysis phases	• Project document • Repository information • Validated modification request	• Feasibility analysis • Detailed analysis	•Conduct technical review • Verify test strategy • Verify whether the documentation is updated or not • Identify security issues	• Feasibility report • Detailed analysis report • Updated requirements • Preliminary modification list • Test strategy
Design phases	• Project document • Source code • Databases • Analysis phases output	• Software test cases • Revise requirements • Revise implementation plan	• Software inspections/ reviews • Verify design	• Revised modification list • Revised detailed analysis • Updated test plan

Phases	Input	Process	Control	Output
Implementa-tion phases	• Source code • System documentation • Results of design phases	• Software code • Unit test • Test preparation review	• Software inspections/ review	• Updated Software • Updated design documents • Updated test documents • Updated user documents • Test preparation review report
System test phase	• Updated software documentation • Test preparation review report • Updated System	• Functional test • Interface testing • Test preparation review	• Software code listing • Modification request • Test documentation	• Test system • Test reports
Acceptance test phase	• Test preparation review report • Full integrated system • Acceptance test plans • Acceptance test cases • Acceptance test procedures	• Acceptance test • Inter-operability test	• Acceptance test	• Acceptance test report
Delivery phase	• Tested/ accepted system	• Installation • Training	• Version description document	• Version description document

SOFTWARE MAINTENANCE MODELS

- Quick-fix model.
- Boehm's model
- Osborne's model
- Iterative enhancement model.
- Reuse-oriented model.

Assignment Question 1

TECHNIQUES FOR MAINTENANCE

Configuration Management

- In organizations Configuration Control Board (CCB) is constituted to oversee the entire change process.
- Request for change is received on a formal change request form.
- The change control form should include information about how the system works, nature of the problem, and how the new (expected) system should work.
- The request for change is reported to the Configuration Control Board.
- The representative of CCB meets the user to discuss the problem.

Configuration Management – Cont….

- When the user requests for a reported failure, the CCB discusses the source of the problem.
- If the requested change is an enhancement, the CCB discusses the parts or the components that will be affected by the change.
- In both the cases, developers describe the scope of change and the expected time to implement them.
- Finally, all the relevant documentation is updated according to the requested change.
- The developers then record all the changes made to the operational system in a change report to keep track of the next release or version of the software system.

Software Versions

- Version control implies the process by which the contents of the software, hardware, or documentation are revised.
- Note that the software configuration management manages how the versions differ, who made the changes, and why they were made.
- The records, such as name of the component, date and time, version status, and account of all changes are managed.
- This helps the software configuration management to identify the current version and the revised number of the operational system.

Impact Analysis

Impact analysis is used to evaluate risks associate with change. Includes estimating effect on effort schedule, and resources. Impact analysis is also used to determine the consequences of making modifications in the software system and to analyze the cost and benefits of the modifications.

Advantages of Impact Analysis are:

- To understand situations when the modifications required in the software system affect large segments of software code or several components of the software.
- To understand relationship of the components that are to be modified along with the structure of the software.
- To record the history of modification, which helps in maintaining quality in the software system.

Software Rejuvenation

- May include enhancing or completely replacing a software system. To preserve or increase the software quality, and its maintainability while keeping the costs low.

- Four types of software rejuvenation exist, namely:
  - redocumentation,
  - restructuring,
  - reverse engineering, and
  - re-engineering.

Assignment Question 2

Software Maintenance Tools

Name	Description
File comparators	Compares two files or systems and maintains the record for the differences in files. In addition, it determines whether the two files or the systems are identical or not.
Compilers and linkers	Compilers are used to check syntax errors and (in some cases) locate the type of errors. When the code is compiled , a linker is used to link code with other components, which are required for executing the program. Linkers sometimes are used to track the version numbers of the components so that appropriate versions are linked together.
Debugging	Allows to trace the logic of the program and examines the contents of the registers and memory areas.
Cross-reference generators	Assures that the changes in code are in compliance with the existing code. When a change to a requirement is requested, this tool enables one to know which other requirements, design, and code components will be affected.
Static code analyzers	Measures information about the code attributes, such as number of lines of codes, number of spanning paths and so on. This can be calculated when the new versions of system are developed.