Unit – V

Software Maintenance

Software maintenance is one type of process, which helps to upgrade, modify, and update software to stay up with client needs. Software maintenance is performed for a variety of purposes, including improving the software overall, addressing faults or bugs, increasing performance, and more, once the software is released or launched.

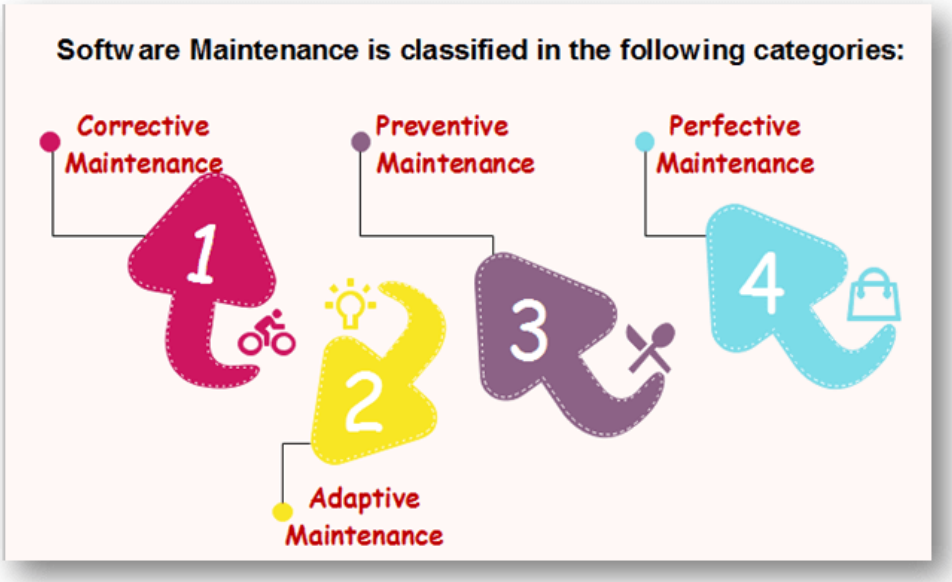
Software maintenance is known as the modification of a software product after it has been delivered to rectify flaws, improve performance, or other features, in terms of software engineering.

If we talk about computer software maintenance, it's a vast management process and part of the SDLC – **Software Development Life Cycle**. The primary goal of software maintenance in software engineering is to modify and update software applications after deployments to fix bugs and improve system performance. The software maintenance processes take place once the software is developed or deployed. Hence, it enhances the software performance by eliminating errors, removing unusable development, and implementing advanced development strategies.

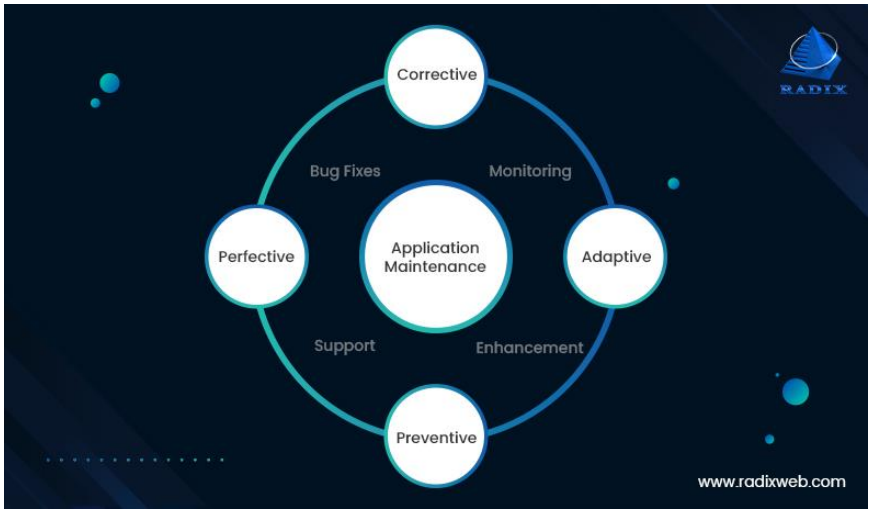
However, maintaining software is not only the scenario of the post-development process. Apart from making your software error-free, you have to make sure that the software is secure and scalable. Your software may become out dated if you don't constantly upgrade it with new features and resolve bugs. You may feel that the software is running smoothly; thus, it doesn't require any maintenance. Therefore, you need to analyze your software on a regular basis and update it constantly.

If a tech lead asks to address the bottleneck (testing) of the software, it becomes your responsibility to offer the software, which satisfies the customers without facing any errors or bugs. In fact, it's the key to survival in this competitive world.

Types of Software Maintenance Services



There are four different types of software maintenance, which are defined for various reasons and purposes. A software product may have to undergo one or more types of maintenance throughout the software maintenance life cycle.



1. Adaptive Maintenance

Adaptive software maintenance is the process of conversion in the system to keep the software compatible with changing business needs and technical evolution. This type of software maintenance primarily focuses on software frameworks. It is made in response to new operating systems, platforms, and hardware to retain continuity with the software.

Adaptive software maintenance is about changing software in response to changes in its environment.

The primary goal of adaptive software maintenance is to update and modify the software when:

- The operating system on which your software executes is evolving (due to technology, laws, policies, rules, operating system, etc.)
- End-users require the product to work with new hardware or software.
- You've foreseen software defects that will harm your customers in the future.

2. Perfective Maintenance

Perfective Maintenance is a process of modifying all elements, functionalities, and abilities to enhance system operations and performance. The software's receptiveness and usability are solved by perfective software maintenance. It includes altering current software functionality by improving, removing, or inserting new features or functions.

Perfective software maintenance focuses on functional enhancements to improve the user experience.

If you want to update the software system to improve its value as per the user requirements, you can execute the perfective software maintenance. This includes:

- Performance enhancement
- Enhanced user interfaces and software usability
- Better software functionality and performance

3. Corrective Maintenance

Identifying errors in the existing solution and correcting them to make it works more accurately. This software maintenance activities aim to eliminate and fix bugs or issues in the software. Corrective software maintenance is usually done in the form of small updates frequently.

In a nutshell, corrective software maintenance occurs when there are errors and faults in logic, code, and design.

Corrective software maintenance is about corrective software bugs, errors, and defects.

You can implement corrective software maintenance when:

- Software doesn't function properly due to some faulty logic flow, wrong implementation, invalid or incomplete tests, etc.
- Users face issues with the software once it is published.

4. Preventive Maintenance

Preventive software maintenance service helps in preventing the system from any forthcoming vulnerabilities. Preventive maintenance defines improvements of the software, which is done to safeguard the software for the future. It is carried out to prevent the product from any potential software alteration. Preventive maintenance also makes it easier to scale or maintain your code and handle your legacy system.

Preventive Software Maintenance defines the adaptions and modifications of the software that mitigate the deterioration risk.

Preventive maintenance offers:

- **Document updation** as per the existing state of the system
- **Code optimization** for better software execution
- **Reconstructing or reducing the code** of the software to make it understandable

Reasons for Software Maintenance

The long lifespan of software depends on its ability to be upgraded to run smoothly on the system. Therefore, here are some reasons you need to maintain software.

Change in user requirement with time

To improve system efficiency

Bug Fixing

In maintenance management, bug fixing comes at a priority to run the software seamlessly. This process contains searching out for errors in code and correcting them. The issues can occur in hardware, operating systems, or any part of the software. This must be done without hurting the rest of the functionalities of existing software.

Capability Enhancement

This comprises an improvement in features and functions to make solutions compatible with the varying market environment. It enhances software platforms, work patterns, hardware upgrades, compilers, and other aspects that affect system workflow. Boost your business using a technically updated solution applying software maintenance services regularly.

Removal of Outdated Functions

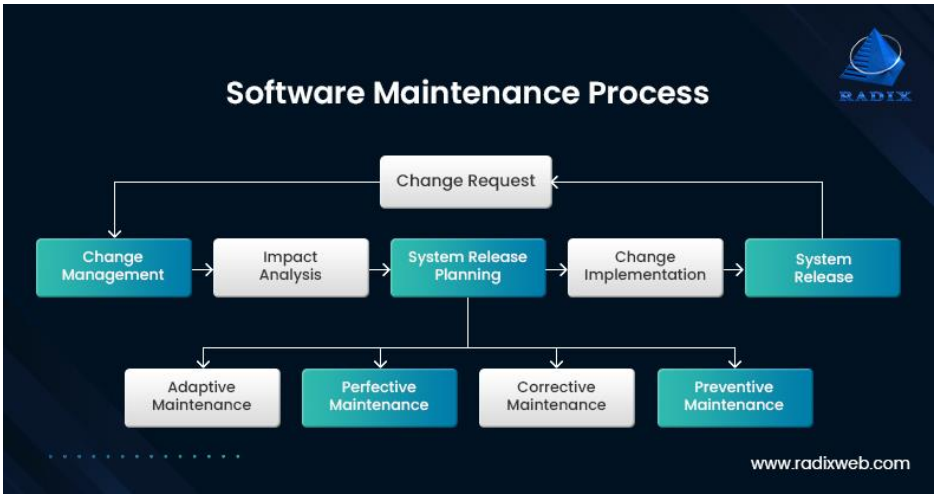
The unwanted functionalities are useless. Moreover, by occupying space in the solution, they hurt the efficiency of the solution. Using a software maintenance guide, such UI and coding elements are removed and replaced with new development using the latest tools and technologies. This elimination makes the system adaptive to cope with changing circumstances.

Performance Improvement

To improve system performance, developers detect issues through testing and resolve them. Data and coding restricting as well as reengineering are part of software maintenance. It prevents the solution from vulnerabilities. This is not any functionality that performs in operations, but it develops to stop harmful activities like hacking.

With the reasons listed above, it becomes necessary to learn the different software maintenance processes or phases and **choose a reliable software development partner**. These go a long way in making the software on the whole robust and free from bugs of any kind.

Software Maintenance Processes



In the life cycle of software development, a software maintenance plan is a very crucial phase. Hence, it is executed in the system through a well-planned software

maintenance process which is known as Software Maintenance Life Cycle (SMLC). SMLC is implemented in seven different phases. Those are:

Phase 1 – Identification

As the name goes, in this phase of the software maintenance life cycle, the modifications are ‘identified’. Before implementing the changes for the requests raised, the modifications are first analyzed and classified according to the attention or maintenance it requires. This phase can be automated or manually done by a user.

Phase 2 – Analysis

The practicality and feasibility of each verified modification request are planned to incorporate changes in the software. The analysis includes validated changes or input where the cost of modification is also estimated.

Phase 3 – Design

The new framework of the software is determined according to the result of the analysis. Survey or test software is also developed for the purpose of safety and security.

Phase 4 – Implementation

This is where the main or new **software framework** is implemented; as in, the codes are crafted, and in the new support system, specifications are added.

Phase 5 – System Testing

In this testing, the implementation of codes and specifications are tested. This stage determines if any further changes or additions are required in the new model of software.

Phase 6 – Acceptance Testing

This stage is performed by third-party end-users. They run a dummy software test, also known as a dry run test, to check if the implemented specifications are working properly, which was mentioned in the modification request.

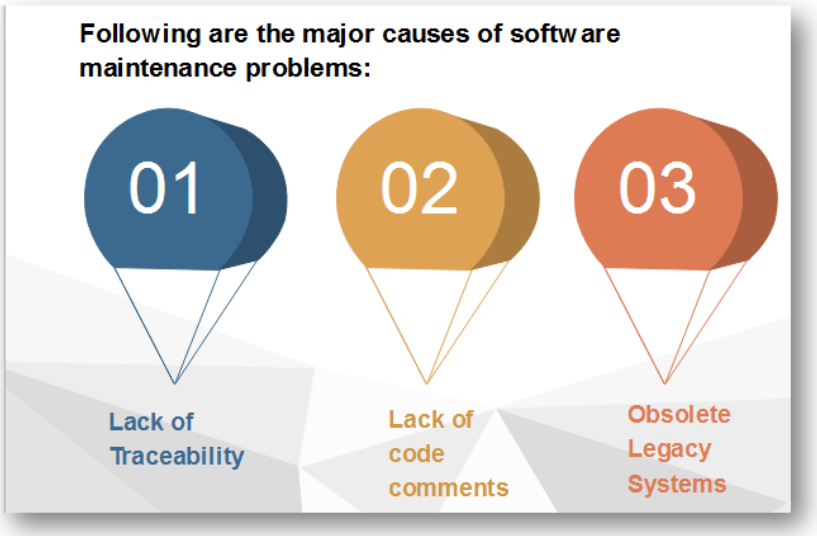
Phase 7 – Delivery

As and when the testing phase is cleared and the developers get a green signal from the third-party users, they deliver the software to the primary users.

Causes of Software Maintenance Problems

Lack of Traceability

- Codes are rarely traceable to the requirements and design specifications.
- It makes it very difficult for a programmer to detect and correct a critical defect affecting customer operations.
- Like a detective, the programmer pores over the program looking for clues.
- Life Cycle documents are not always produced even as part of a development project.



Lack of Code Comments

- Most of the software system codes lack adequate comments. Lesser comments may not be helpful in certain situations.

Obsolete Legacy Systems

- In most of the countries worldwide, the legacy system that provides the backbone of the nation's critical industries, e.g., telecommunications, medical, transportation utility services, were not designed with maintenance in mind.
- They were not expected to last for a quarter of a century or more!
- As a consequence, the code supporting these systems is devoid of traceability to the requirements, compliance to design and programming standards and often includes dead, extra and uncommented code, which all make the maintenance task next to the impossible.

Software Maintenance Process

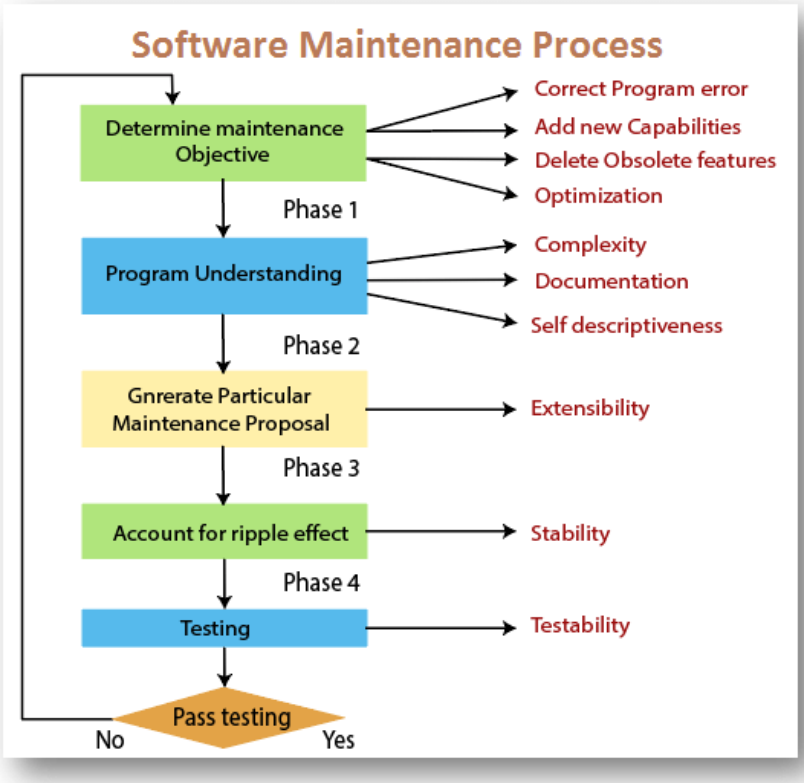
Program Understanding

00:00/05:19

The first step consists of analyzing the program to understand.

Generating a Particular maintenance problem

The second phase consists of creating a particular maintenance proposal to accomplish the implementation of the maintenance goals.



Ripple Effect

The third step consists of accounting for all of the ripple effects as a consequence of program modifications.

Modified Program Testing

The fourth step consists of testing the modified program to ensure that the revised application has at least the same reliability level as prior.

Maintainability

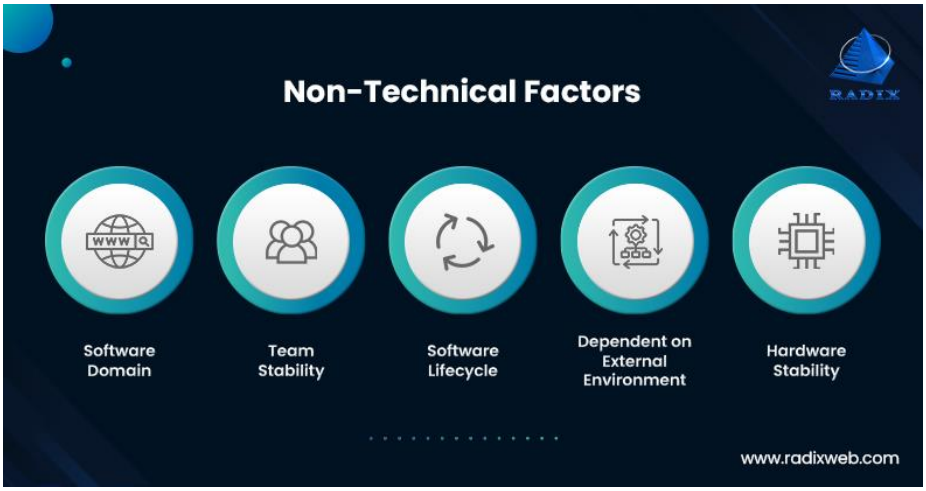
Each of these four steps and their associated software quality attributes is critical to the maintenance process. All of these methods must be combined to form maintainability.

Software Maintenance Cost

The cost of the specific software can be categorized into three different components: Software License, Software Maintenance, and Implementation Services.

But if we consider the cost of software maintenance, it is categorized into two parts.

1) Non-Technical Factors



The non-technical factors include:

Software Domain: When the domain of the software is well-defined, system requirements may not change over time. This will lead to a fewer chance of maintenance.

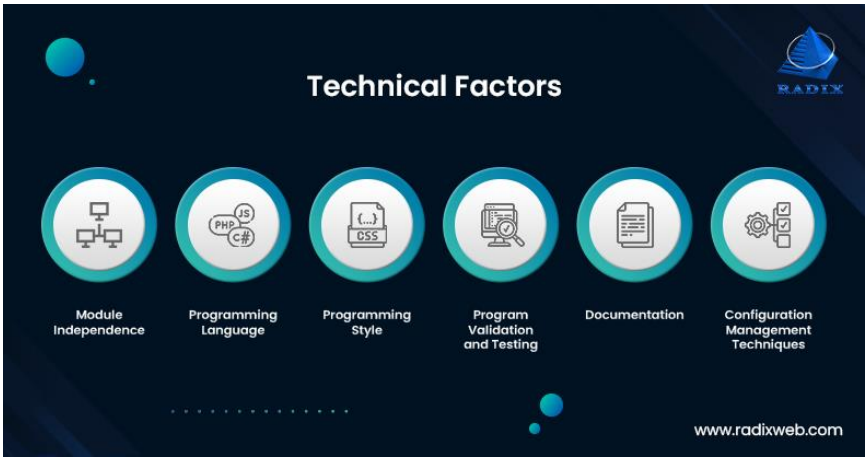
Team Stability: When the new team member joins the software development team, it takes some time to get his hands-on the software development process. Therefore, it becomes quite difficult to make changes in the software. This will add on a cost to the software maintenance.

Software Lifecycle: When software becomes obsolete, the original hardware is changed, and the conversion cost exceeds the rewriting cost.

Dependent on External Environment: When software is dependent on the external environment, it must be modified whenever the external environment changes.

Hardware Stability: Software maintenance expenses are reduced to zero if the software executes on a specific hardware configuration that does not change during the entire software lifecycle. However, this is a rare occurrence due to ongoing hardware development.

2) Technical Factors



The technical factors include:

Module Independence: The ability to update any software block in the system without impacting the others.

Programming Language: Software written in a high-level programming language is generally easier to understand than written in a low-level language.

Programming Style: The developer's writing method determines the ease of updating and understanding software.

Program Validation and Testing: The more time spent evaluating the design and testing the software, the fewer bugs it contains and the lower the cost of software maintenance. The cost of resolving errors is determined by the type of error. Errors in software requirements are the most expensive.

Documentation: A clear and complete documentation will bring down the maintenance cost.

Keys Strategies for Successful Software Maintenance

There are two main techniques or strategies that we should consider for successful software maintenance in software engineering.

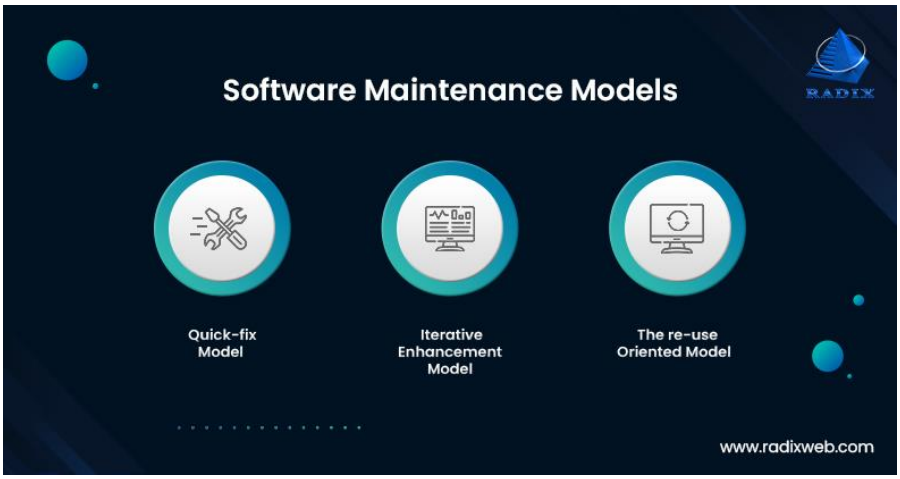
Here's explaining these two concepts in detail below.

Documentation – The process involves containing information related to code operation. In addition, it also includes solutions to problems that may occur in the near future. This makes the task of upgradation considerably easier.

QA (Quality Analysis) – Coming second in the list, the QA process can be integrated either before the launch or during the planning stage itself. This will make sure to deploy software without any errors or bugs. You can also get an insight into the necessary changes that need to be made.

As we explain software maintenance, its cost and the strategies to bring it down completely, it is worth describing the different models of this process too.

Case Tools / Software Maintenance Models



There are many software maintenance models, but the most important models are:

- Quick-Fix Model
- Iterative Enhancement Model
- Re-Use Oriented Model

Quick-Fix Model

Just how it sounds, the main aim of this model is to search for glitches and fix them as soon as possible. The main advantage of the quick-fix model is that it works very rapidly at a low price. The approach of this model is to modify the coding with minimal consideration.

Iterative Enhancement Model

The primary nature of the changes in this model is iterative. Based on the analysis of the existing system, the changes are incorporated in this model. It requires complete documentation of the software that is available before making any changes.

Re-Use Oriented Model

In this model, the part of the existing system that is in the position of using is identified; hence it is called the ‘re-use-oriented model’. After analyzing, this model goes through changes or enhancements as required.

Configuration Management

Keeping track of system documentation and ensuring uniformity is one of the costs involved. This means that good configuration management can help you save money.

The process usually costs up to two-thirds of the entire software process or even more than 50% of the SDLC processes on the whole.

However, on a positive note, costs for software maintenance can, in fact, be brought down by following the below steps.

- Adhere to functional programming principles
- Follow a transparent development process
- Don’t forget about (Re)documentation
- Hire experienced software developers
- Do not accumulate technical debt