software maintenance

Introduction

Maintaining software is not less important than software development. It keeps solutions athletic to deal with developing technology and the business environment.

Conventionally, IT service providers suggest their consumers to go for software maintenance services for the enhanced and steady performance of the system. When it comes to software, 60% costing is abide for maintenance and from total software maintenance cost, 60% is for solution enhancement.

Content table

What is Software Maintenance?

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updations done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

- **Market Conditions - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain**

bookkeeping, may trigger need for modification.

- **Client Requirements - Over the time, customer may ask for new features or functions in the software.**
- **Host Modifications - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.**
- **Organization Changes - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.**

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 correcting 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**

Why Software Requires 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.

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.

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.

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 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.

Configuration Management Techniques: 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.

Why? This is because if, for instance, the software is old, the maintenance cost will be high too. So, it may happen that developers may not be 100% successful in targeting the exact issues. This will result in enormous hours being spent at work.

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**

Abiding by these steps will support you in bringing down the software maintenance costs to a great extent.

Now you may have got an idea about the costs you may have to incur to maintain software. Now, you must be wondering what strategies to follow to make the software maintenance process easier.

To throw light on this area, we present below the following approaches. This

will make the process seamless and help you save costs.

Reduce the software maintenance cost and increase its stability

Key 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.

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.

Conclusion

Software maintenance is not an option; it is a must. You can consider your vehicle, for example. If you don’t maintain your vehicle, it may cause many problems often. Also, the bill for the unperformed vehicle maintenance will cost you higher than the regular inspection or maintenance. Similarly, if you don’t take software maintenance is seriously in your business plan, there will be a lesser scope for optimum business growth.