**ASIF KHAN**

**Student ID: 15428**

**MS Software Engineering**

**Subject: Topics In Software Engineering**

**Spring Semester, Assignment, May - June 2020**

**Teacher Name: Dr. Fazal-e-Malik**

**Q1. Study some Case Studies for software maintenance and maintenance costs and write summaries of these case studies.**

**SOFTWARE MAINTENANCE AND ITS COST:**

The task is based on the reading of three studies. In order to avoid repetition in the basic theme, the important notes have been taken care of, and is merged (all of these from three case studies) into one, and of course have been tried to summarized too.

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. 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. Over the time, customer may ask for new features or functions in the software. Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification. If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.

**MAINTENANCE TYPES:**

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

**CORRECTIVE SOFTWARE MAINTENANCE:**

Corrective software maintenance is what one would typically associate with the maintenance of any kind. Correct software maintenance addresses the errors and faults within software applications that could impact various parts of your software, including the design, logic, and code. These corrections usually come from bug reports that were created by users or customers – but corrective software maintenance can help to spot them before your customers do, which can help your brand’s reputation.

**ADAPTIVE SOFTWARE MAINTENANCE:**

Adaptive software maintenance becomes important when the environment of your software changes. This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system. Sometimes, adaptive software maintenance reflects organizational policies or rules as well. Updating services, making modifications to vendors, or changing payment processors can all necessitate adaptive software maintenance.

**PREFECTIVE SOFTWARE MAINTENANCE:**

Perfective software maintenance focuses on the evolution of requirements and features that existing in your system. As users interact with your applications, they may notice things that you did not or suggest new features that they would like as part of the software, which could become future projects or enhancements. Perfective software maintenance takes over some of the work, both adding features that can enhance user experience and removing features that are not effective and functional. This can include features that are not used or those that do not help you to meet your end goals.

**PREVENTIVE SOFTWARE MAINTENANCE:**

Preventative Software Maintenance helps to make changes and adaptations to your software so that it can work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. These services can include optimizing code and updating documentation as needed.

Preventative software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

For all businesses and organizations, software maintenance is an essential part of the software development lifecycle. This isn’t something that one can skip or avoid. It is absolutely necessary for the success of your software and any evolution into the future. It is important to know that maintenance needs to go much further than fixing issues or bugs – that is only one steps of the software maintenance process.

Updating software environments, reducing deterioration, and enhancing what is already there to help satisfy the needs of all users are also included in the software maintenance examples.

**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 its 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.

Reverse engineering skills are also used to detect and neutralize viruses and malware, and to protect intellectual property. Computer programmers proficient in SRE will be needed should software components like these need to be maintained, enhanced, or reused. It became frightfully apparent during the Y2K crisis that reverse engineering skills were not commonly held amongst programmers. Since that time, much research has been under way to formalize just what types of activities fall into the category of reverse engineering, so that these skills could be taught to computer programmers and testers. To help address the lack of SRE education, several peer-reviewed articles on SRE, software re-engineering, software reuse, software maintenance, software evolution, and software security were gathered with the objective of developing relevant, practical exercises for instructional purposes. The research revealed that SRE is fairly well described and all related activities mostly fall into one of two categories: software-development-related and software-security-related. Hands-on reversing exercises were developed in the spirit of these two categories with the goal of providing a baseline education in reversing both Wintel machine code and Java bytecode.

**SOFTWARE MAINTENANCE COST DEFINED AND EXPLAINED:**

"Software maintenance cost is derived from the changes made to software after it has been delivered to the end user. Software does not “wear out” but it will become less useful as it gets older, plus there WILL always be issues within the software itself."

Software maintenance costs will typically form 75% of TCO. Software maintenance costs includes Corrective maintenance – costs due to modifying software to correct issues discovered after initial deployment (generally 20% of software maintenance costs)

Adaptive maintenance – costs due to modifying a software solution to allow it to remain effective in a changing business environment (25% of software maintenance costs)

Perfective maintenance – costs due to improving or enhancing a software solution to improve overall performance (generally 5% of software maintenance costs)

Enhancements – costs due to continuing innovations (generally 50% or more of software maintenance costs).

Software production and maintenance issues, costs estimation, project schedule and knowledge of the process have always been complicated cases in software engineering. Cost depends on the creation and maintenance of the software. Thus, continuous monitoring and control of maintenance costs, and software optimization, are really important. Taking into account this statistic, also leads to careful software maintenance to reduce costs. Software maintenance costs are rising and based on the estimations about 90% of the cost related to the software life is in the maintenance phase. The estimations show 50 percent increase over the past two decades.

Incomplete documentation and low maintenance could also be the factor to increase the cost. Therefore, the defect makes it difficult for the maintenance team to expand or rebuild the product. Because the production team members may have left the company, be retired or replaced by another person who are unaware of the production process.

Since quality improvement and reduced software lifecycle time are among rapid application development techniques, the use of common-sense approach in the production shows that using individual techniques is not a threat to high availability, acceptable performance and quality of projects.

The researchers introduced support and maintenance software to estimate the maintenance effort. In these researchers’ point of view support and maintenance software were a set of activities to support IT. Magne Jorgensen came to this conclusion that 43 to 44% of the estimations are mentally done by the experts and using such models results in the estimations complexity.

Hence the estimation techniques along with the knowledge of the project team, project manager and the president to design a predictive model for estimating the software. This model suggests that the maintenance plays an important role in the success of IT projects. Though the effective use of technology for estimating the time and cost is necessary but is not sufficient. To predict the exact time and cost, the management needs the knowledge, knowledge integration and sharing it.

**HOW MANAGE TO SOFTWARE MAINTENANCE COST:**

Lack of proper reporting or documentation and people who work on its development retire or leave without being replaced. Re-engineering the system is not an option because the system that needs to be rebuild could be extensive and not well documented.

The cost of maintaining software can be three times higher than the cost of development. Several businesses do not realize that the initial costs of designing and developing software are just a small part of the total cost of its ownership (TCO).

The cost of maintaining software is rising exponentially and it has been found that software maintenance could account for 70% of the total cost of the software. There are reasons behind this. While more and more software is being developed every day, it becomes hard to maintain them over time.

Moreover, businesses think that maintenance of the software is somehow related to bug fixing, that is not true. Every business evolves and they may need to make enhancements as the business grows. Enterprises may also need to update the existing software to run smoothly on newer versions of both mobile and desktop operating systems. Therefore, maintenance is more crucial than development itself and leads to higher costs.

maximize ROI with reduced software maintenance costs, you will be able to focus more on delivering new software rapidly. It results in the maximized ROI for the business. Automated monitoring tools for maintaining the performance of the software can help in reducing maintenance costs. Therefore, reduced operational costs will result in increased productivity and output. And Many organizations hire expensive onsite support team for the maintenance of software, that lead to high maintenance costs. You can choose to work with a partner who works remotely at cost-effective prices.

In conclusion, What the Informatics experts agree on for reducing maintenance costs, is that “with respect to some important factors such as accuracy in HIS projects feasibility, along with complete documentation and helping the design and implementation mechanisms in the maintenance phase, favorable results can be achieved in reducing the cost. “

Generally, we can conclude that for an accurate assessment and reduce the cost of software maintenance, software maintenance factors determining is essential. This will lead to the longer life of software. Evaluation of these factors and their influence on each of the maintenance costs, help the project manager in making decisions and planning, and is essential in the success of software maintenance. Project managers must consider these factors for success in their projects and decisions: