Software Maintenance

Introduction

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. Amazon website launched, them created mobile App

Client Requirements - Over the time, customer may ask for new features or functions in the software.
Example Started with 5 modules in Amazon added new module of return of material

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. Example Created in JAVA then switched to python l

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. Example introduced for electric material and mobile phones

Categories of Maintenance

Traditionally, 4 types of maintenance have been distinguished, which are differentiated by the nature of the tasks that they include:

Corrective maintenance: The set of tasks is destined to correct the defects to be found in the software and that are communicated to the maintenance department by user himself.

Preventive Maintenance: Its mission is to maintain a level of certain services of software, programming the interventions of their vulnerabilities in the most opportune time. It is used to be a systematic character, that is, the software is inspected even if it has not given any symptoms of having a problem.

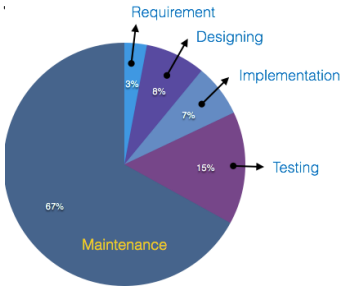
Perfective Maintenance: is concerned with implementing new or changed user requirements. It involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs. Examples of perfective maintenance include Re-organizing data sets within a database so they can be

Periodic maintenance (Time Based Maintenance TBM): the basic maintenance of software made by the users of it. It consists of a series of elementary tasks (data collections, visual inspections, cleaning, lubrication, retightening screws,…) for which no extensive training is necessary, but perhaps only a brief training. This type of maintenance is the based on TPM (Total Productive Maintenance).

Cost of Maintenance

Reports suggest that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle.

On an average, the cost of software maintenance is more than 50% of all SDLC phases. There are various factors, which trigger maintenance cost go high, such as:



- Real-world factors affecting Maintenance Cost
- The standard age of any software is considered up to 10 to 15 years.
- Older software, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced software on modern hardware.
- As technology advances, it becomes costly to maintain old software.
- Most maintenance engineers are newbie and use trial and error method to rectify problem.
- Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
- Changes are often left undocumented which may cause more conflicts in future.

Software Re- engineering

Software Re-engineering

When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.

Legacy software cannot keep tuning with the latest technology available in the market. As the hardware become obsolete, updating of software becomes a headache. Even if software grows old with time, its functionality does not.

For example, initially Unix was developed in assembly language. When language C came into existence, Unix was re-engineered in C, because working in assembly language was difficult.

Other than this, sometimes programmers notice that few parts of software need more maintenance than others and they also need re-engineering.

Re-Engineering Process

Decide what to re-engineer. Is it whole software or a part of it?

Perform Reverse Engineering, in order to obtain specifications of existing software.

Restructure Program if required. For example, changing function-oriented programs into object-oriented programs.

Re-structure data as required.

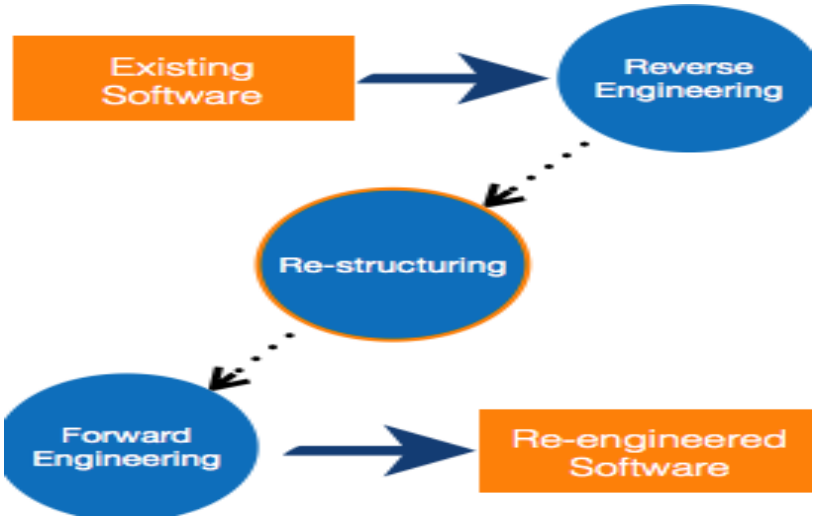
Apply Forward engineering concepts in order to get re-engineered software.

Software Reverse Engineering

Reverse Engineering

It is a process to achieve system specification by thoroughly analysing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analysing lower abstraction levels.

An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.



Software Restructuring and Forward Engineering

Program Restructuring

- It is a process to re-structure and re-construct the existing software. It is all about re-arranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code-restructuring and data-restructuring or both.
- Re-structuring does not impact the functionality of the software but enhance reliability and maintainability. Program components, which cause errors very frequently can be changed, or updated with re-structuring.
- The dependability of software on obsolete hardware platform can be removed via re-structuring.

Forward Engineering

- Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.
- Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering.