SOFTWARE MAINTENANCE

Name:Ankita Sadhukha

Roll No.:18700120132

Stream:CSE Sem5th Sec:3

Year:2020-2024

INTRODUCTION:

Software Maintenance is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software application after delivery to correct faults and to improve performance.

Need for Maintenance –

Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.

Categories of Software Maintenance –

Maintenance can be divided into the following:

1. Corrective maintenance: Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

2. Adaptive maintenance: This includes modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

3. Perfective maintenance: A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

4. Preventive maintenance: This type of maintenance includes modifications and updations to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

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 it's 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.

Why Reverse Engineering?

- Providing proper system documentatiuon.
- Recovery of lost information.
- Assisting with maintenance.
- Facility of software reuse.
- Discovering unexpected flaws or faults.

Used of Software Reverse Engineering –

- Software Reverse Engineering is used in software design, reverse engineering enables the developer or programmer to add new features to the existing software with or without knowing the source code.
- Reverse engineering is also useful in software testing, it helps the testers to study the virus and other malware code .

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 softwares, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced softwares 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.

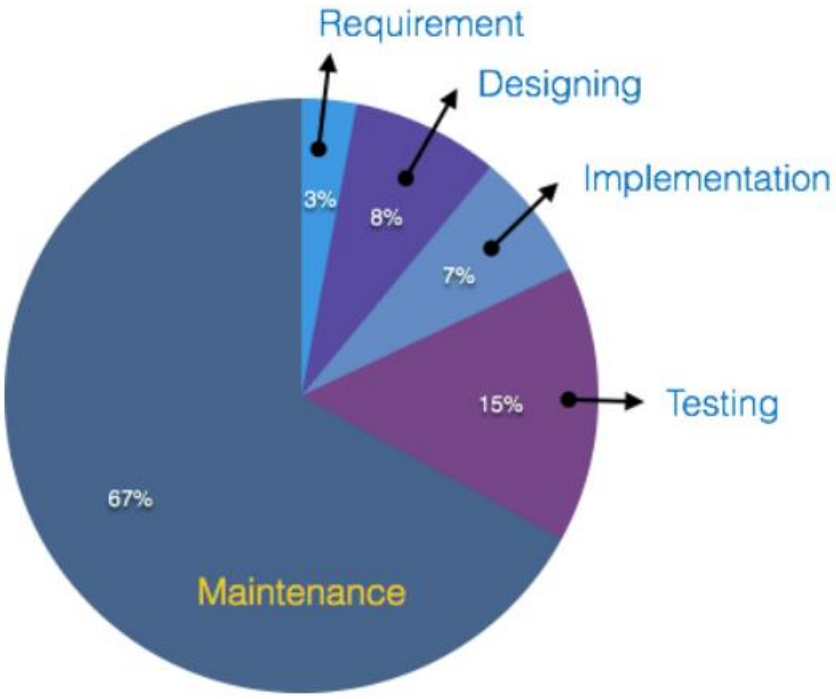
• Changes are often left undocumented which may cause more conflicts in future. Software-end factors affecting Maintenance Cost

• Structure of Software Program

• Programming Language

• Dependence on external environment

• Staff reliability and availability



Maintenance Activities

IEEE provides a framework for sequential maintenance process activities. It can be used in iterative manner and can be extended so that customized items and processes can be included.

These activities go hand-in-hand with each of the following phase:

• Identification & Tracing - It involves activities pertaining to identification of requirement of modification or maintenance. It is generated by user or system may itself report via logs or error messages.Here, the maintenance type is classified also.

• Analysis - The modification is analyzed for its impact on the system including safety and security implications. If probable impact is severe, alternative solution is looked for. A set of required modifications is then materialized into requirement specifications. The cost of modification/maintenance is analyzed and estimation is concluded.

• Design - New modules, which need to be replaced or modified, are designed against requirement specifications set in the previous stage. Test cases are created for validation and verification.

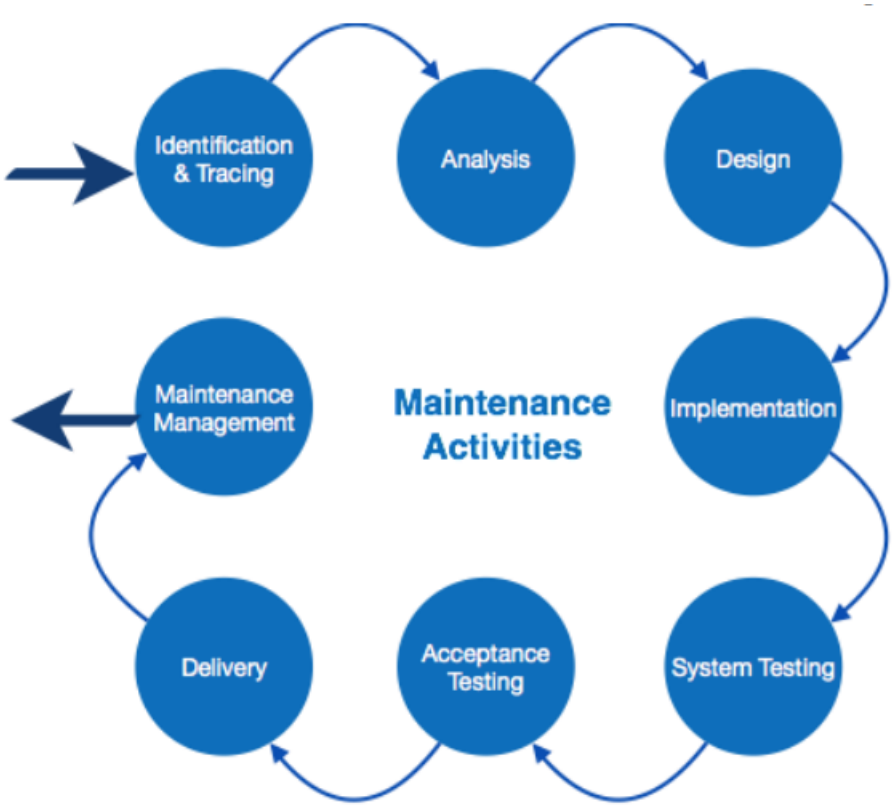
• Implementation - The new modules are coded with the help of structured design created in the design step.Every programmer is expected to do unit testing in parallel.

• System Testing - Integration testing is done among newly created modules. Integration testing is also carried out between new modules and the system. Finally the system is tested as a whole, following regressive testing procedures.

• Acceptance Testing - After testing the system internally, it is tested for acceptance with the help of users. If at this state, user complaints some issues they are addressed or noted to address in next iteration.

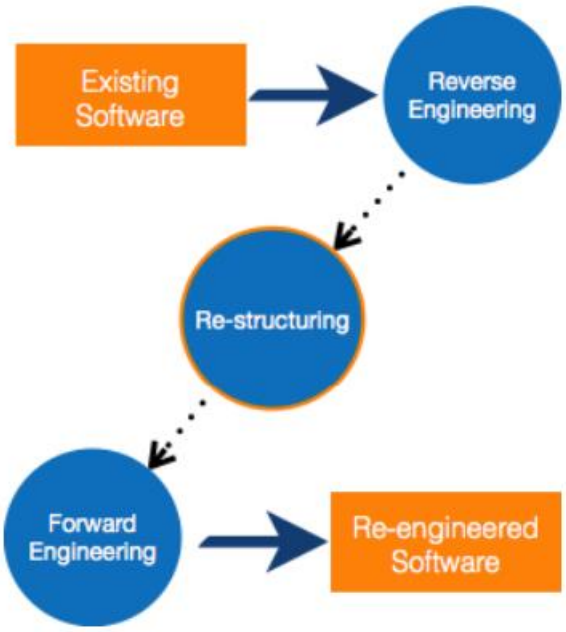
• Delivery - After acceptance test, the system is deployed all over the organization either by small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered. Training facility is provided if required, in addition to the hard copy of user manual.

• Maintenance management - Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi-version or patch management.



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 objectoriented programs. • Re-structure data as required.
- Apply Forward engineering concepts in order to get re-engineered software.

There are few important terms used in Software re-engineering

Reverse Engineering

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

CONCLUSION

Testing is a crucial element of the maintenance phase of software. The more software you already have developed, the harder it is to ensure that bug fixes and enhancements don't cause additional headaches. Good software development technique can minimize the likelihood that a code change will have unexpected repercussions, but can't reduce the chances to zero. Testing should be frequent and thorough, which means using as much automation as possible.

Unit testing is becoming increasingly common, and is even being picked up by development phase programmers in the form of Extreme Programming. GUI testing is harder, and you'll have to come up with policies that balance the required frequency and thoroughness with your own sanity.

A well-designed test bed and test set will save you a lot of apologies to your customers, who don't enjoy updating software and certainly don't want to do it more than once to fix any particular problem. Using maintenance testing techniques can save time, money, and dignity.

REFERENCES

- *"ISO/IEC 14764:2006 Software Engineering — Software Life Cycle Processes — Maintenance". Iso.org. 2011-12-17. Retrieved 2013-12-02.*
- ^ *Soleimani Neysiani, Behzad; Babamir, Seyed Morteza; Aritsugi, Masayoshi (2020-10-01). "Efficient feature extraction model for validation performance improvement of duplicate bug report detection in software bug triage systems". Information and Software Technology. **126**: 106344. doi:10.1016/j.infsof.2020.106344. S2CID 219733047.*
- ^ Pigoski, Thomas M., 1997: Practical software maintenance: Best practices for managing your software investment. Wiley Computer Pub. (New York)
- ^ Eick, S., Graves, T., Karr, A., Marron, J., and Mockus, A. 2001. Does Code Decay? Assessing Evidence from Change Management Data. IEEE Transactions on Software Engineering. 27(1) 1-12.