A Step Towards Software Preventive Maintenance

Yogesh Singh[1], Bindu Goel[2]
University School of Information Technology,
Guru Gobind Singh Indraprastha University,
Kashmere gate, Delhi (INDIA)

[1]ys66@rediffmail.com,
[2]bindu_delus@yahoo.com

Abstract

In a world interwoven economically, increasing dependence on critical software applications either in transaction processing (banking, Government services etc) or manufacturing automation (automobiles, pharmaceuticals, chemicals) has accentuated the economic impact aging software can have. Software with increased usage operating in unforeseen conditions at throughput much higher then initial expectations can degrade fast leading to higher altered form and dramatic drop in performance indicators. In the current paper we have tried to analyse the issues governing software maintenance and how preventive maintenance which is still considered a very evolving field in the context of software engineering can help the software product age usefully. We have also attempted to address the above in the essence of how it is done for hardware preventive maintenance which is a better understood and commercially accepted concept. Finally we suggest model for the preventive maintenance integrated within software life cycle.

Keywords: Software maintenance, maintainability, hardware preventive maintenance, documentation.

Introduction

In today's risk sensitive business and economic environment, software has become the single most valuable corporate asset in any business, industrial or government organisation. Software driven activity forms the most critical business continuity element in any organisation's disaster management. So it is of utmost importance that more research effort is put in obtaining a better understanding of software problems .And how they can be avoided through regular maintenance regime thus avoiding the avoidable software failures. With the increasing maturity in the information technology sector, more and more software systems are entering the maintenance phase thus requiring disciplined and scheduled maintenance effort.

Software maintenance accounts for more effort than any other software engineering activity. Although it has until very recently been a neglected phase in the software engineering process, maintainability is a key goal that guides its steps. Maintainability of software is the degree, to which it can be understood, corrected, adapted and/or enhanced [1]. Maintenance activity can originate from any kind of failures i.e. processing failure, performance failure or implementation failure. Four types of maintenance that are performed on software are [2]:

a) ***Corrective maintenance*** acts to correct errors that are uncovered after software is in use
b) ***Adaptive maintenance*** is applied when changes in the external environment precipitate modifications to software
c) ***Perfective maintenance*** incorporates enhancements of existing system functionality or improvement in computational efficiency that are requested by customers
d) ***Preventive maintenance*** improves future maintainability and provides a basis for future enhancements.

The costs of the maintenance processes are not distributed evenly across all categories. Studies by Leintz and Swanson [3] show that 50% of the total maintenance efforts can be attributed to perfective maintenance, 25% for adaptive maintenance, whereas only 21% of the total efforts are attributed to corrective maintenance and 4% for preventive maintenance shown in fig 1.

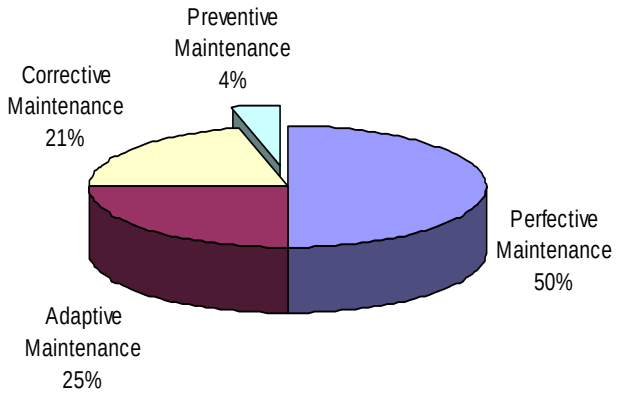

Fig 1: Distribution of maintenance effort

There has always been a debate on the classification of these types of maintenance. This is partly due to the fact that many activities are difficult to classify, for example, some organizations separate out software reengineering from perfective maintenance. In addition, it is not unusual that while performing adaptive maintenance one finds a defect, or perhaps decides that some perfective rewriting is necessary to add a new feature [4]. Even when these categories are reasonably well defined, adaptive and perfective work often overlaps corrective work. The two types, corrective and adaptive, along with perfective get the most attention.
The focus in the current paper is on preventive maintenance in software systems. This activity of preventive maintenance is relatively rare because the pressure during the development phase renders the preventive activity to a job of minimum importance. If one considers the probability of a software unit needing change

and the time pressures under which the change is requested, it makes lot of sense to anticipate change and prepare accordingly in advance. Comparison can be drawn to human ageing where preventive measures are taken in due course as aging is assumed to be an inevitable feature. Similarly the organization should undertake preventive measures (preventive maintenance) during the design phase itself without waiting for software to become ineffective. Hence, this is the process of changing software in order to improve its future maintainability or to provide a better platform for future enhancements. Miller [5] defines Preventive maintenance as application of today's methodologies to yesterday's systems to support tomorrow's requirements.

Preventive maintenance has also been seen as a key element of enterprise risk management. Risks are assessed and identified and necessary mitigating steps taken to prevent any unnecessary loss in the future.

Current state of the field

There exists inconsistency in the way preventive maintenance is defined. It is interpreted in different manner by different researchers [6, 7]. Much of this inconsistency is attributed by the current definition of maintenance and its categories as defined by IEEE [8-10] (fig 2).

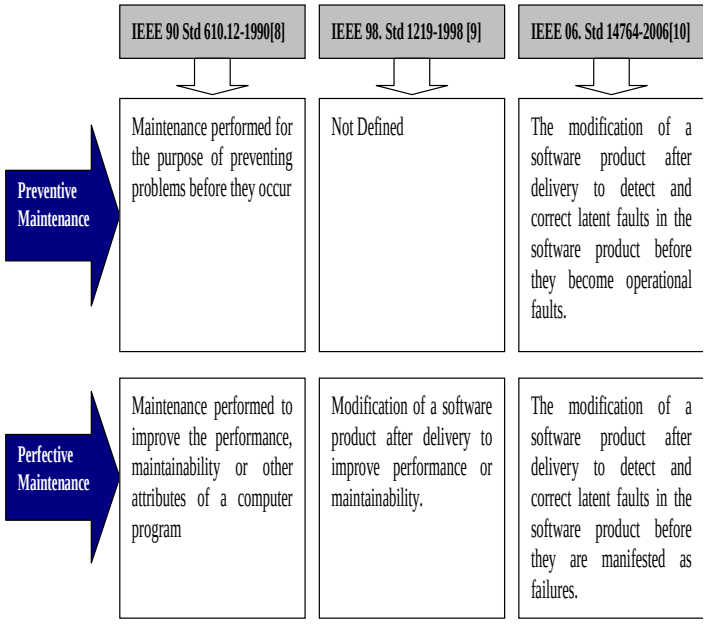

Fig 2: IEEE definitions of maintenance categories

The latest standard [10] defines the activities and tasks of software maintenance, and provides maintenance planning requirements. It does not address the operation of software and the operational functions, e.g., backup, recovery, system administration, which are normally performed by those who operate the software.

Preventive/perfective maintenance is an anticipatory activity as it is performed by anticipating or forecasting problems which might occur in future. Since both of them are anticipatory and not reactive to detected errors/changes/enhancements, both of them can be grouped under preventive maintenance activities. In other words preventive maintenance is done to modify software in order to improve the maintainability. The primary product attributes that contribute to software maintainability are

- Making the software less complex (modularity) or
- Easier to interpret (Clarity) or
- Good internal documentation of the source code as well as appropriate supporting documents.

Till date preventive maintenance has been taken seriously in the context of hardware maintenance (like lubrication of parts or routine checkups) while for software systems such concept of preventive maintenance is still in an exploratory phase [11]. It is right to state that while undertaking hardware maintenance repair or replacement of identified faulty component is accomplished with ease, the same activity can't be undertaken so easily during software maintenance as:

- The source of failure in software is design faults while the principle source in hardware is physical deterioration. Once the design fault is fixed the life span is expected to be very high. While the same life curve cannot be drawn for the hardware as it does not follow any predictable physical deterioration pattern.
- In hardware preventive maintenance a scheduled replacement of the hardware component can be done thus preventing it to fail. To do so estimates of parameter called mean operating time between failures (MTBF), acceptable number of failures etc. is done.
- Non availability of Off the shelve software components/modules like spare parts in hardware maintenance.
- Ripple impact of change in any software component/module on other elements of the software while the faulty component replacement does not have any ripple effect in hardware maintenance
- In software systems the applications are highly customized thus each site implementation is unique. While in case of hardware the machinery are more or less standardized across applications.

The maintenance in software means dealing with ageing software. Classically, software aging has two main symptoms increased failure rate, and decreased service rate. In [12] two types have been identified: *Software product ageing* and *software process execution ageing*. Software product ageing is degradation in software code and documentation quality by continual maintenance. Software process execution ageing is the degradation in performance characteristics of a software system through continuous running. Preventive maintenance for software systems represents a proactive approach to operational software fault-tolerance, and aims at counteracting the aging effect. [13]

Classification

Preventive maintenance is not a classification factor. However, preventive software maintenance is an important activity that should be defined in each organization, taking into account the local planning and working procedures. Thus, the practical definition is: Preventive software maintenance refers to all activities that are prepared and decided upon regularly, for example annually, in co-operation between the client and the Maintainer organizations, and are based on the joint analyses of the present condition as well as the forecasted needs of the software.

In this section we have tried to identify the kind of activities which can be done in preventive maintenance. Broadly many organizations classify maintenance into two major activities as either **repair or enhancement**. This is true from practical point of view .In preventive measures the future faults are avoided today. Although the preventive maintenance is a scheduled activity but any kind of maintenance request can be understood as an example of the following classification:

- Preventive corrective maintenance
- Preventive adaptive maintenance
- Preventive perfective maintenance

Preventive corrective maintenance: This activity is initiated by any kind of defect/error in the software. Correction of a software problem may reinitiate the development cycle in the analysis phase, the design phase or the implementation phase. A defect can result from design errors, coding errors or implementation or the errors which were left undetected during testing. Repeated corrective maintenance in a fixed time constraint sometimes lead to program complexity and non consideration of ripple effects (effect on the other part of the system due to the change in one part) which renders the distortion in the logic of the software. Hence all the activities which arise due to an error are grouped under this type.

Preventive adaptive maintenance: It includes modification or enhancement in the software in order to adapt to a new operating system or interfaces. These changes can also be driven by the change in economic environment. The term economic environment refers to influence on the software due to change in business rules, government policies, and work patterns. Enhancement or adaptation of the software reinitiates development in the analysis phase.

Preventive perfective maintenance: This activity is done to enhance and improve processing efficiency or performance of the software. This includes clearing log files, back up files, to boost up the performance or change some non-functional features of the software such as response time, throughput and memory size to increase processing efficiency. It also includes modification in the software to improve the maintainability e.g. by making the software less complex, understandable and readable source code.

Emergency requests pertaining to any kind of upset in the software generally does not come under preventive maintenance. Any unscheduled modification performed to temporarily keep a system operational is a part of corrective maintenance.

The Software maintenance like software development requires a combination of managerial control and technical expertise. So apart from the above classifications of different kind of problem requests there are other economic, managerial factors to be considered before performing the preventing maintenance activity as discussed below:

- The costs spent on emergency maintenance or corrective approaches outweigh the cost incurred on preventive maintenance. i.e. preventive maintenance is cheaper and efficient approach than emergency maintenance.
- The preventive maintenance is generally scheduled for the time when the software is available i.e. there is no load on the system and thus typically results in lesser downtime and cost [13].
- Management issues: The management attitude plays a major role in setting the right pitch for the maintenance job [14]. The management should be able to assign priority to a request on the basis of its emergency and significance. Sufficient Training, tools and motivational environment should be provided to maintenance team. The team should not be discouraged by giving extrapolating development assignments as rewards and giving more bonuses to development teams. Also the time constraint should be realistic and achievable so that quality jobs are delivered.
- Expertise of the maintenance team also plays a significant role. The Technical expertise, functional knowledge of the business domain around the software under maintenance and also the programmer attitude are the key attributes.
- The sync between the IT teams of the organization and business users is important. In this the customer attitude also plays a role like how responsive the customer to the queries of the maintenance team.

Model of preventive maintenance

A model for the preventive maintenance is presented in the present section. The model (fig 3) is based on the classification presented in the paper in the preceding section. It is outlined on the basis of the development life cycle. It should be observed that software maintenance is a microcosm of the software development cycle.

.

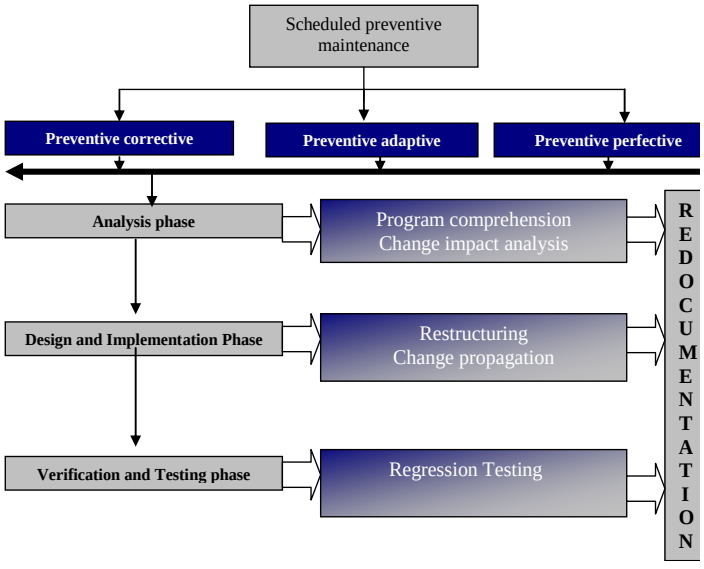

Fig 3: Model of preventive maintenance

A major distinction between development and maintenance is the set of constraints imposed on the maintainer by the existing implementation of the system. Information about system artifacts, relationships and dependencies can be obscure, missing, or incorrect as a result of continued changes to the system. This situation makes it increasingly difficult for the maintainer to understand the software system and the implications of a proposed change

Model outline

In this model we have outlined the life cycle of any type of maintenance request (MR). Maintenance request can be classified as per the classification suggested above. The preventive corrective maintenance can initiate either in analysis phase or the Design and implementation phase. While either preventive adaptive or perfective initiates in the analysis phase only. The request for a change may be a bug report or a request for additional functionality.

Analysis phase consists of two important activities: Program comprehension and Change impact analysis.

- Program comprehension is an important tool in the hands of the maintainer to understand the software. Multiple changes in the life of a software renders it a different code altogether. So Comprehension in the form of documentation and the self descriptiveness of the program contribute to the ease of understanding the program.
- Change impact analysis is the activity by which the programmers assess the extent of the change i.e. the other components that will be affected. This is called the ripple effect. Also in this the feasibility and the cost of the change is accessed. And once this preliminary phase establishes the feasibility then it is moved to next phase.

Design and implementation phase In this the two activities of restructuring and change propagation are undertaken.

- Restructuring includes the change in design structure to accommodate for the given change. Sometimes the given architecture does not support contemplated change, and then the software should be restructured first.
- Now the change in structure triggers the change in the neighbouring components. This process is called change propagation. The change propagation should end with the consistent software.

The phase again ends with the update of the program comprehension.

The software which is changed at last is verified and validated in the ***testing phase***. In this the modified program is tested to ensure its reliability. Regression testing is the process of testing the modified parts of the software and ensuring that no new errors have been introduced into the previously tested code. Again this phase is closed by the updating of the documents.

Final remarks

In the current paper we have tried to classify different change requests. We presented the domain state of software preventive maintenance and how this suffers from the multiple interpretations. And lack the objective understanding of maintainability. This kind of scheduled maintenance can actually lower the cost and effort required for the phase. In nutshell preventive maintenance is designing a software system that is easy to maintain and continuously upgrading a system to enable it to cope with current and future change.

1. Updating and issuing enhancements from time to time. The corrected releases should be announced well in time so that attention is given to maintainability.
2. One should periodically monitor system health and prevent system illness by checking the system maintainability level.
3. Significant resources could be saved at the upfront maintenance process level through providing:
    a. Ongoing user training in relevant systems and their operations
    b. Written recovery restart instructions and notifications about known problems
    c. Software rejuvenation: the software is periodically stopped and restarted in order to refresh its internal state. This prevents or at least postpones the occurrences of failures. Through software rejuvenation implies overheads; it prevents more severe (and therefore more costly) failures.
8. Maintain a trend analysis to account for predictable changes. And keep historical data from the past maintenance for future reference.

References

1. Pressman RS. Software Engineering (5th edition) McGraw-Hill Companies: New York NY, 2001.
2. Aggarwal K.K.and Singh Y. ,Software Engineering : Programs, documentation, operating procedures. New Age international publishers, 2005.
3. Lientz BP, Swanson EB. Software Maintenance Management: a Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations. Addison-Wesley Publishing Company: Reading MA, 1980.
4. Hatton, L. How Accurately Do Engineers Predict Software Maintenance Tasks? Volume 40, Issue 2, Feb. 2007 Page(s): 64 – 69.
5. Miller,J.C.,1979, Techniques of Program and System Maintenance,1981,ed. Parikh, G., Winthrop Publishers, 181-182.
6. Chapin N, Do We Know What Preventive Maintenance Is? In Proceedings, International Conference on Software Maintenance, IEEE, Computer Society Press in Los Alamitos CA, 2000.
7. Bennett,K.H .,Rajlich, V.T., Software Maintenance and evolution : a Roadmap,Proceedings of the conference on the future of Software Engineering table of contenets,Limerick.Ireland,2000, pp 73-87.

8. IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990 (1991 Corrected Edition). The Institute of Electrical and Electronics Engineers, Inc., 1994.
9. IEEE Standard for Software Maintenance, IEEE Std 1219-1998. The Institute of Electrical and Electronics Engineers, Inc. 1998.
10. IEEE Standard for Software Maintenance, IEEE Std 14764-2006. The Institute of Electrical and Electronics Engineers, Inc. 2006.
11. Kajko-Mattsson M, Can we learn anything from Hardware preventive maintenance? , 2001 IEEE.
12. Kajko-Mattsson M, Preventive Maintenance! Do We Know What It Is?, International panel, In Proceedings, International Conference on Software Maintenance, IEEE Computer Society Press in Los Alamitos CA, 2000.
13. Garg S, Puliafito A, Telek M, Trivedi K., Analysis of preventive maintenance in transactions based software systems. IEEE transactions on Computers, 47(1), pp. 96-107, 1998.
14. Bhatt Pankaj,Shroff Gautam, Misra Arun K., Dynamics of Software Maintenance,ACM SIGSOFT Softwatre engineering notes ,Volume 29,Number 5,September 2004.