International Journal of Information Technology and Management

Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

IMPORTANCE OF SOFTWARE MAINTENANCE

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

Importance of Software Maintenance

Anshul Kapil[1] Dr. S. B. L. Tripathy[2]

[1]Dept. Of Computer Science, Research Scholar of OPJS University, Churu, Rajasthan -India

[2]Assistant Professor, G.L. Bihani S.D. College, Ganga Nagar, Rajasthan

Abstract – In the late 1970s, a famous and widely cited survey study by Lentz and Swanson, exposed the very high fraction of life-cycle costs that were being expended on maintenance. An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resource and cost estimates. A new decision should be addressed for the developing of every new system feature and its quality objectives. The software maintenance, which can last for 5–6 years or even decades after the development process, calls for an effective plan which can address the scope of software maintenance, the tailoring of the post-delivery/deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs.

----------♦----------

INTRODUCTION

Software management is the art and science of planning and leading software projects. It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled. Software management has the capacity to help plan, organize, and manage resource pools and develop resource estimates. Depending on the sophistication of the software, it can manage estimation and planning, scheduling, cost control and budget management, resource allocation, collaboration software, communication, decision-making, quality management and documentation or administration systems. Today, numerous PC-based software management packages exist such as Frame bench-podio etc. and they are finding their way into almost every type of business.

There are several common activities which apply to most software management processes.

- **Versioning**.

Same developers use a version control system such as subversion to keep track of the changes to source code, especially when there are multiple developers working on the same software.

- **Building**.

In this process, the software is build or compiled. If any kind of error is detected in this stage then that error is removed from the source code.

- **Testing**.

Usually considered part of "the build", there are often automated tests for software and sometimes also manual tests to verify that the software was built/installed/packaged/configured correctly. Sometimes, it is possible to skip the tests. Sometimes, it is hard to run them or interpret the results. So, testing is done after the compilation of the software.

- **Packaging**.

Once software has been built then next step is to share it with others. To this end, the software is packed up into some kind of "archive format".

- **Configuring**.

Source code is often written to be able to function in a variety of environments, such as on different operating systems or even on different kinds of hardware. Some software may also function differently depending on what other software is installed on the system already. Lastly, users may wish to customize how the software behaves or is installed. So in this process, the software is configured in such a way that it can run in any kind of environment or any specified environment.

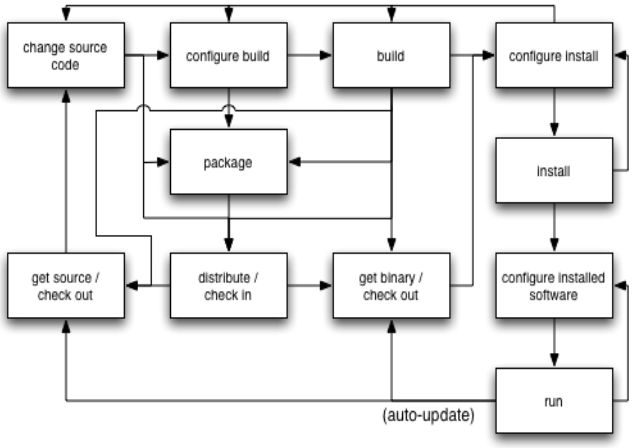
- **Installing**. After software has been built and configured, the next procedure is to install the software into specific locations on the target system,

register it with the operating system as a service, or something similar.

- **Distributing**. After the packaging of software, the next procedure is to publish it on the internet, a cd-rom or similar medium, or by submitting our changes to a centralized or decentralized version control system.

REAL LIFE SOFTWARE MANAGEMENT

A rather generic flowchart which incorporates many common processes is:



In this chart, each and every step could probably be skipped or replaced by a "no-op", and some arrows are followed more commonly than others.

For example, most software developers tend to follow an iterative process when writing code. They will "switch into development mode", change some source, rebuild and test the change, change some more source, rebuild and test, check in the changes, rebuild and test, create a distribution package, "switch to production mode", get the distribution, build and test it, do some configuration, rebuild, then install.

SOFTWARE MAINTENANCE:

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes. A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that the majority, over 80%, of the maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system. More recent studies put the bug-fixing proportion closer to 21%.

Software maintenance and evolution of systems was first addressed by Meir M. Lehman in 1969. Over a period of twenty years, his research led to the formulation of Lehman's Laws. Key findings of his research include that maintenance is really evolutionary development and that maintenance decisions are aided by understanding what happens to systems and software over time. Lehman demonstrated that systems continue to evolve over time. As they evolve, they grow more complex unless some action such as code refactoring is taken to reduce the complexity.

The key software maintenance issues are both managerial and technical. Key management issues are: alignment with customer priorities, staffing, which organization does maintenance, estimating costs. Key technical issues are: limited understanding, impact analysis, testing, and maintainability measurement.

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanisms must be developed for evaluation, controlling and making modifications.

So any work done to change the software after it is in operation is considered to be maintenance work. The purpose is to preserve the value of software over the time. The value can be enhanced by expanding the customer base, meeting additional requirements, becoming easier to use, more efficient and employing newer technology. Maintenance may span for 20 years, whereas development may be 1-2 years.

REVIEW OF LITERATURE:

Bateman classified maintenance programs as reactive, preventive and predictive maintenance. Preventive and predictive maintenance represent two proactive strategies by which companies can avoid code breakdowns. In reactive maintenance, the software is allowed to run until failure and then the failed software is repaired or replaced [Paz and Leigh, 2004]. Though under reactive maintenance, temporary repairs may be done in order to return software to operational condition and permanent repairs made later time (Gallimore and Penlesky, 2008). Proactive maintenance is a strategy for maintenance whereby breakdowns are avoided through activities that monitor code and undertake minor updations to restore code to operational condition. These activities, including preventive and predictive maintenance, reduce the probability of unexpected software failures. Preventive maintenance is often referred to as use-based maintenance where maintenance activities are undertaken after a specified period of time (Herbaty, 1990; Gits, 1992). Weil (2008) added another approach in his description of the maintenance spectrum by including Total Productive Maintenance (TPM). TPM is an aggressive maintenance approach that seeks to improve software performance while continuing to avoid software failures applied to all the project as the ultimate objective of any factory is to have a highly efficient integrated system and not brilliant individual module [Oechsner *et al.* 2002].

McKone *et al.* [2001] described positive impacts of TPM practices. Further, software performance can be measured with four different basic parameters i.e. cost, quality, delivery, and flexibility that are extended in some studies including several measures [Skinner ; Hayes *et al. 200*8; Schroeder 2003; Miller and Roth, 2004; Ward *et al. 200*5].

OBJECTIVES OF THE STUDY:

The objectives of present study are :

1. To study the implementation process of software.
2. To study the maintenance process of software.
3. To study the efficiency of processes involved in software management.

RESEARCH METHODOLOGY:

The survey showed that around 75% of the maintenance effort was on the first two types and error correction consumed about 21%. Many subsequent studies suggest a similar magnitude of the problem. Studies show that contribution of end user is crucial during the new requirement data gathering and analysis. And this is the main cause of any problem during software evolution and maintenance. So software maintenance is important because it consumes a large part of the overall lifecycle costs and also the inability to change software quickly and reliably means that business opportunities are lost.

Not only are error-prone modules troublesome, but many other factors can degrade performance too. For example, very complex “spaghetti code” is quite difficult to maintain safely. A very common situation which often degrades performance is lack of suitable maintenance tools, such as defect tracking software, change management software, and test library software. Below describe some of the factors and the range of impact on software maintenance.

Importance of Software Maintenance

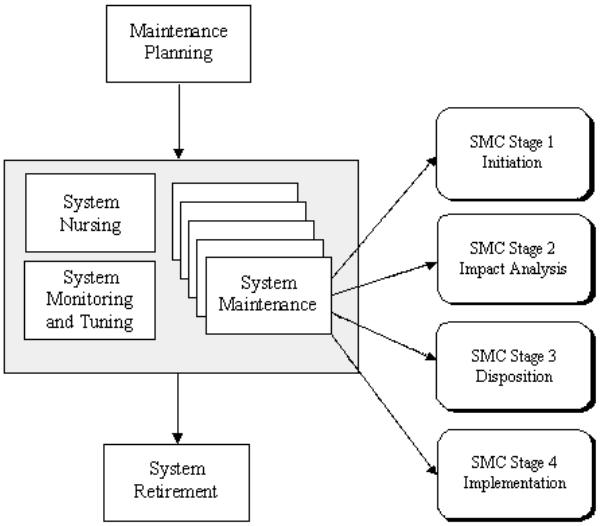
In the late 1970s, a famous and widely cited survey study by Lentz and Swanson, exposed the very high fraction of life-cycle costs that were being expended on maintenance. They categorized maintenance activities into four classes:

- **Adaptive** – modifying the system to cope with changes in the software environment.
- **Perfective** – implementing new or changed user requirements which concern functional enhancements to the software.
- **Corrective** – diagnosing and fixing errors, possibly ones found by users.
- **Preventive** – increasing software maintainability or reliability to prevent problems in the future.

System Maintenance

System Maintenance of an application system commences when the system is accepted and put into production. On-going monitoring and maintenance of the system have to be performed during the System Maintenance Cycle (SMC) so as to ensure the performance and functionality of the system match with the business needs of the users. Detailed information is provided in the Guidelines on System Maintenance Cycle.

System Maintenance Cycle of an application system is depicted below:-



Maintenance Planning:

A maintenance plan should be prepared in the Project Closure Stage and submitted to Project Steering Committee (PSC) for approval. It details

- the role assignments of SMC organization;
- the procedures of maintenance activities;
- the required deliverables for SMC; and
- the maintenance resources and facilities delegated for SMC.

Review of the plan should be conducted as need arises and properly approved.

BIBLIOGRAPHY:

- Ahmed Awad E. Ahmed, and Issa Traore , “A Study on Software Management Processes”, Proceedings of 6th IEEE Information Assurance Workshop, pp. 452- 453, 2005.
- Abdalla M. Elramsisi, Fareed Zaghlool, Tharwat O. S. Ahanafy and Abdou Saad El Din Moustafa, “Neural Approach Modeling Scheme for the Software Management”, New York Science Journal, Vol. 3, No. 12, pp. 142-149, 2010.
- Akila, M., Suresh Kumar, S., “Improving Feature Extraction in Software Management”, Proceedings of the International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011), pp. 891–898, 2011.
- Ali, H., Wahyudi, Salami, M., “Essential Components of Software Management”, Proceedings of 5th International Colloquium on Signal Processing & Its Applications, pp. 198–203, 2009.
- Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi, “Modified Levenberg-Marquardt Method for Neural Networks Training”, World Academy of Science, Engineering and Technology 6, pp. 46-48, 2005.
- Anil Jain, Karthik Nandakumar, Arun Ross, “Study on Software Management”, Pattern Recognition, Vol. 38 , pp. 2270 – 2285, 2005.
- Aqlan. A. M, W. F. Abd El-Wahed and M.A. Abd El-Wahed, “A Study on Maintaince processes of Software ”, 6th International Conference on Informatics and Systems, Faculty of Computers & Information-Cairo University, Cairo-Egypt, pp. 110-117, 2008.
- Azevedo, G.L.F., Cavalcanti, G.D.C., Carvalho Filho, E.C.B., “Analysis of Software Management Processes”.
- Boehm, B.W. The high cost of software. Proc Symp on High Cost of Software, Monterey, Calif., pp. 27--40.
- Boehm, B.W., Brown, J.R., and Lipow, M. Quantitative evaluation of software quality. Proc. 2nd Int. Conf. on Software Eng., pp. 592-605.
- Gildersleeve, T.R. *Data Processing Project Management.* Van Nostrand Reinhold, New York.
- Implications of Using Modular Programming. Guide No. 1, Hoskyns Syst. Res., J. Hoskyns and Co., London.
- Khan, Z. How to tackle **the systems maintenance** dilemma. *Canadian Data Syst.* , 30-32.
- Kosy, D.W. Air Force **command and** control information processing : Trends in software technology. U.S. Air Force Proj. RAND, RAND Corp., Santa Monica, Calif., p. 70.