A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts

Uttamjit Kaur
Department of Computer Science
GIMET
Amritsar

Gagandeep Singh
Department of Computer Science
GIMET
Amritsar

ABSTRACT
Software Maintenance and evolution are identifying by their huge cost and slow speed of implementation. Survey showed that around 60% of the maintenance effort was on the total cost of software. But after delivering the software to the client, the maintenance work begins. This paper suggests some issues and problems faced by software maintenance process. There are some issues of software maintenance i.e.: database size, system age, maintenance budget, system size, staff size or restructuring for change. This paper presents several ways to reduce cost and efforts involved in software maintenance. Software maintenance costs can be reduced significantly if the software architecture is well defined, clearly documented, and creates an environment that promotes design consistency through the use of guidelines and testing quality.

General Terms:
Software Maintenance, maintenance cost reduction.

Keywords
Software maintenance, issues and problems in software maintenance, cost and challenges in maintenance, maintenance cost reduction.

1. INTRODUCTION
Software does not get tired, it need to meet some new requirements that enhanced the software system. The changes made in the software system can be performed by software maintenance. Software Maintenance comes under process when a software team delivers a successful project to its client within a fixed time. Software Maintenance is the modification of a software product after delivery. The Modification can be done to improve performance, correct faults and to adapt the product to a modified environment [4].According to ISO, software maintenance used to check modification and documentation for a problem that need for improvement. To update the existing software product then changes can be preserving by holding its originality. To change the software after some operation performed as: Enhancement of capabilities, election of problem capabilities and optimization. The cost of software maintenance is rising more than 90% of the total cost of software [1].In addition, software maintenance serves the following:

1.1 Improving the software to support user requirement
User request for new requirement, to enhance the performance or functionality of the software. User desire some functions changes that can be accommodated.

1.2 Supporting Upgrades
Upgrades required when some changes in government regulations. The need for upgrades used to maintain software that exist in same category.

1.3 Providing Continuity of Service
Maintenance focus on recovering from failures such as hardware or software and changes in the operating system.

Three type of software maintenance considered: Corrective Maintenance used for emergency program fixes and routine debugging: Adaptive Maintenance used to making changes in response to technology changes: Perfective Maintenance used to improve documented and requested enhancement [3].Survey showed that around 75% of the maintenance effort was on the adaptive and perfective maintenance. Or 21% consumed by error correction [4].Maintenance costs depend on the Number of changes and the costs of change depend on the maintainability. Important issues of software maintenance are limited understanding. Limited understanding refers to how quickly a software engineering can understand where to make a change or correction. In software 40-60% of the maintenance effort is devoted to this task. Limited understanding is to produce documentation. Documentation is lacking or incomplete and the people who know the software leave or retire without being replaced. Other issues i.e. customer priorities, staffing, and cost estimation with some technical issues i.e. testing, maintainability. If we want to reduce the overall cost of software, the goal of development should be reduce the maintenance effort [1].

2. MAINTENANCE PROBLEMS
Maintenance costs are due to software improvement rather than corrections. Most problems of software maintenance are associated with the software development process [3]. In software process, software engineer develop the part of the software, which may maintaining the software, has to get to get with the detailed design and functioning of the source code. Maintenance cost and effort is very important part of a planning process to estimate them. Estimate helps to plan maintenance staff. Maintenance is not only concerned with technical issues, but also the department and organizational issues [3]. The problems may be either technical or organizational as follows:

2.1 Program Comprehensive
Maintenance engineer used to modify the behavior, functionality, understanding, adding new features and changing the system. The objectives of maintenance can be generated only when the attributes of system are understood. It is the central research problem; once the change and its impact have been understood, it is relatively simple to make it. It is a prerequisite of the change and it has been a subject of extensive research. It consumes more than half of all maintenance resources [8].

2.2 Change impact analysis
Change impact analysis is a major challenge faced by the maintenance process, to determine the effects of a new modification on other parts of the system. The software

changing can have impact on the rest of the system. In new modification, it involves the identification of the system's parts that need to be modified. Change impact analysis is the activity by which components that will be impacted by the change [8]. It indicates that how costly the change is going to be.

2.3 Change Implementation

Change implementation may consist of several steps, each one specific software component. If the visited components are modified, it may no longer fit with other components because it no longer properly interacts with other. Then secondary changes must be made in neighboring components. This process called **Change propagation.** Successful change starts and ends with consistent software, during the change propagation the software are often inconsistent [8].

2.4 Regression Testing

The process of testing a system after it has been modified is called regression testing. This type of testing is important to confirm that system is free of errors after it has been modified and also ensure that the rest of the system has not been affected by the modification. This testing can be reuse, if needed. The Software changes, when a new module is added as part of integration testing. If new modules are added then new control logic, new input/output and new data flow paths are established. The modification or change cause a problem with functions that worked previously. According to integration test strategy, regression testing is used to ensure that changes may not affect the software or it is the Re-execution of subset of tests that have already been conducted.
Successful tests can be conducted by, identify the errors and errors must be corrected. When software is corrected, the program, its documentation or data it supports is changed. Regression testing helps to ensure that changes do not add some additional errors. Regression testing can be done manually, by Re-execution the test cases of subsets or using automated with some Capture/playback tools. The Capture/playback tool used to capture the test cases by software engineer [7].
The regression test suite, the subset of tests to be executed by three different classes of test cases:

1. All Software functions that exercise the tests must represent sample.
2. New additional software functions are affected by the change.
3. Software components that are tests have been changed.

The numbers of regression tests become large, as the testing proceeds. The design of regression test suits must address one or more classes of errors in each program functions. Once the change has occurred, every test for program function become impractical and inefficient.

2.5 Programmer Time Availability

It refers to maintenance programmer shortage, programmer turnover and programmer increasing demand [3].

2.6 User Knowledge

User knowledge refers to problems caused by user expectations, lack of user training and understanding. Limited understanding means that how quickly a user can understand, where a change or correction to make in siftware.40-60% of the maintenance effort conducted by this task [2].Limited understanding used to produce poor documentation and other supporting descriptions. Survey showed shows that comments and source code are the important facts to understanding software to be maintained.

2.7 User Demand and Expectations

User expected to change their software quickly [3] and used foe enhancement the performance to the current systems. This type of requirement makes a MIS department heavy burden.

2.8 Relationship of software product and environment

According to the operational and organizational environment, software product changes its relationship. However, it is very important to choose those changes that are necessary for the software product.

2.9 Relationship of the software product and user

According to the new requirements from user, software product modified its relationship. Hence, it is very important to choose the Software those are necessary for the user to accept the changes after modification.

2.10 Relationship of software product and software maintenance team

According to the maintenance team members to keep track of the software product to change its relationship. Maintenance team analyzes the changes and effect on the software product.

2.11 Database Size

Database size is measured by number of data files and the number of characters in database. As per the required of the customer during the maintenance and modification in the software product may lead to the modification of the database. So it is not an easy task to modify database again and again [1].

2.12 Product Quality

It refers to quality of original design specification and quality of original programming. The unavailability of up-to-date systems documentation affects maintenance product quality.

2.13 System Age

It correlated with the use of corrective maintenance. Older systems which are entering the growth phase should have more problems with lack of user knowledge, product quality and programmer time [3].

2.14 Staff Size

Staff size describes the number of people who are engaged in the development process of software under application development. After delivering the project, customers need some changes and then the software team which is engaged in the other development projects. If software organization assigns the maintenance work to new developers or programmers then the organization needs to provide training to the programmer, which leads to the increment in time, cost and efforts for the maintenance [3].

2.15 Staff Turnover

It is survey that, when staff turnover is high, then maintenance is not properly performed in the software. The ratio of number

of individuals that leaves the organization during a specified period of time. Software product are replaced by new personnel who spend the maintenance effort in understanding the system [2].

2.16 Operating Environment

Operating environment include are hardware and software reliability, software failure, data integrity and documentation. A large system increases the errors and with large database, a greater amount of change in data and files, as well as a greater need for hardware and software upgrades. Hardware and software made advantage to technological advances [3].

2.17 System Size

It can be measured by number of program modules and number of source statement contained in the system. Large systems are more complex and harder to maintain.

2.18 Maintenance Budget

It used to measure the importance of maintenance as reflected in the budget. System with more resources devoted to maintenance should have fewer problems [2].

2.19 Documentation Quality

If the documentation is poor, then it becomes very costly to find any faults that are involved in the system. This indicates that documentation quality has a effect on maintenance effort [2].

2.20 Restructuring for Change

Changes can be restructure due to architecture do not support the change as the change are delocalized [8].

2.21 Development Experience of maintenance staff

It is very clear that, if the maintenance staffs have greater experience with system development, then there would be fewer problems with programmer and product quality.

2.22 Legacy Software

Software legacy was one of the major software problems, but it has become less recently. Legacy software is not so much a technological problem as an organizational and management problem. If current technology uses i.e. component, middleware, enterprise etc will provide the ultimate answer to all problem, then there will be no more legacy software there will be no more legacy software. Over fifteen years in technology solutions to legacy systems, much effort has been expended. This is clear from detailed and practice papers in the literature [8].

In the section on servicing, we carefully avoided introducing new terminology, and concentrated on the need for better methods for program comprehension, it support by code improve. The existing legacy system may now be the only source of information about the organization's business rules.

1. It is predict that modern technology can be use instead of existing legacy; maintenance can be cheaper/easier. It has been argued earlier that the maintenance for many new technologies is not understood.
2. High and low level design helps maintenance, but little detailed that this helps maintenance staff who have responsibility for only part of the system. If several representations of software exist, it is only the source code that is maintained other become inconsistent.
3. It is not clear that how much business rules may be of little value. It requires large amounts of time from highly skilled software engineers and experts.
4. Sometimes the source codes are not available, for many components and their behavior may be very expensive.

Sometimes Domain knowledge and Business rules are recorded, where the place of legacy code, and the development of a new modified system may have knowledge that encapsulated in the old system. In short, legacy software can be identified as" Large software system that we don't know how to handle it and encapsulate it but that are important to our organization". Legacy systems can be managed by a numbers of options. Some solutions are: leave the legacy system and building a replacement system; stop the system and its components can be use in anew replacement system; maintain the system and; change the system to give another period of life [2].

Modification can be identifying by complexity and reduction of size to Re-documentation, Re-engineering and Restructuring or wrapping and migration. Migration solution explore when we do not know what future software system will look like. It is to be expected that the problem is raised from addressing code to addressing components in distributed system.

3. HOW TO REDUCE MAINTENANCE EFFORT

There are several ways to reduce cost and efforts involved in software maintenance. Software maintenance costs can be reduced significantly if the software architecture is well defined, clearly documented, and creates an environment that promotes design consistency through the use of guidelines and testing quality are describe as:

3.1 (Re) Documentation

Maintenance cost can be reduced by (Re) documentation. Without documentation programmer spent 21.5% understanding of code. With documentation, we could save 12% of the cost of maintenance .Without documentation the good and bad programmer skill level disappeared.During the past decade, CARE has developed by Omnext, a concept that helps to cost reduction. CARE Stands for "Computer Aided (Re) documentation and evaluation. This concept can be used to improve the principal [6]. CARE may generate up to 50% of saving on the cost, impact analysis, documentation and testing. It also used to increases the quality of software. This concept comprises the technical and functional documentation. CARE helps to monitoring the software system's size and quality.

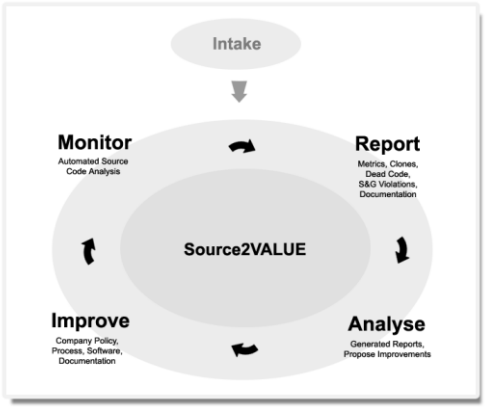

Figure 1: CARE [6]

1. **Intake**: A CARE environment is depending on the environments that have a software documentation and software evaluation [6]. The result, generated by the CARE must be analyzed, the standards, and the method of documentation and evaluation.

2. **Monitor**: CARE based on monitoring, in order to analyze the sources automatically or manually. The result generated by the CARE with quality and size. For documentation purposes, it may contain structure and relationship between application sources [6].

3. **Report**: The different types of reports can be generated from the object. These reports may contain the information about quality, size, and development [6]. For documentation purposes, technical documentation was developed and functional documentation that used to adapt the changes in the application.

4. **Analyze**: Analysis is used to carry out signals, quality, and productivity, and results generated for improvement. For documentation purposes, the functional documentation is also analyzed from the signals [6].

5. **Improve**: Improvement can be occurred by implementing and updating the functional documentation [6]. The updating can be carried out manually or by automatically that used for tools. If required, the CARE definition can be updated on the basis of advanced insight.

3.2 Decreasing Turnovers

Maintenance cost can be reducing turnovers by internal and external. Internal turnovers result from moving internally and external turnovers are the results of people moving out. To reduce maintenance cost through work scheduling and reducing number of internal moves through strict policy.

3.3 Eliminating Dead Code

Dead code means unnecessary code that can be removed without affecting the program that are never called.30% of the software in an older system can be dead code by eliminating dead code, we reduce the code size [6].

3.4 Reduce Complexity

Saving can be achieved by monitoring and improving the technical quality of a system. Maintenance costs affected by the existing software complexcity.25% of maintenance cost should be total life-cycle costs.

3.5 Testing Quality

The number of errors can be reduced by applying an effective testing strategy. With reduced errors, maintenance effort can be quite low. So better testing quality reduces maintenance effort.

3.6 Eliminating Bugs

Bugs in software are costly and difficult to find and fix. Techniques and tools have been developed for automatically finding bugs by analyzing source code [8].

3.7 Understand-ability

It is important that the maintenance gain a complete understanding of the structure, behavior and functionality of the system being maintained [10]. "If we have more information, the test will be smarter".

1. The design is well understood.
2. Dependencies between internal, external, and shared components are well understood.
3. Changes to the design are communicated.
4. Technical documentation is instantly accessible.
5. Technical documentation is well organized.
6. Technical documentation is specific and detailed.
7. Technical documentation is accurate.

Bach suggested the attributes of software configuration i.e., programs, data, and documents that can be used by software engineer [10] to develop a relevant testing. "Good" test have following attributes:

1. High probability-A good test has a high probability to finding an error. To achieve this, s understood by the tester to develop a blueprint of how the software fails. Example: If failure in a graphical user interfaces (GUI), then to recognize proper mouse position. Then the tester used to recognize an error in mouse position. Software being
2. No Redundancy-A good test is not redundant. Time of testing and resources are limited. One test cannot conduct the same purpose as other test. Every test must have a different purpose. For example, a module of the *Safe Home* software is designed to recognize a user password to activate and deactivate the system. In an effort to uncover an error in password input, the tester designs a series of tests that input a sequence of passwords. Valid and invalid passwords (four numeral sequences) are input as separate tests. However, each valid/invalid password should probe a different mode of failure. For example, the invalid password 1234 should not be accepted by a system programmed to recognize 8080 as the valid password. If it is accepted, an error is present. Another test input, say 1235, would have the same purpose as 1234 and is therefore redundant. However, the invalid input 8081 or 8180 has a subtle difference, attempting to demonstrate

that an error exists for passwords "close to" but not identical with the valid password.

3. Best of breed-A good test should be "best of breed". Time and resources are limited that migrate towards the execution for test. In such cases, the whole class of errors should be uncovering be used.

4. Simple and complex-A good test should be neither too simple nor too complex. Sometimes a group of tests combines to test in one test case; the mask errors can be associated with this approach. In general, group of tests should be separately executed,

3.8 Eliminating Cloned Code

Large software systems typically contain large redundant code. When applying new functionality, many programmers cannot copy and customize existing pieces of code [8]. Large software systems contain 10-25% redundant code. If redundant code were removed, spend the budget savings on new problems and enhancing the efficiency of the organization.

3.9 Software Reengineering

Re-engineering a software system can be used to better understand and maintain [8] it long been accepted within the software maintenance. Software Re-engineering can be used as:

1. It is used to improve software understanding.
2. It improves software itself, for reusability and maintainability.
3. It is an activity that absorbs the information regarding resources.
4. It helps to reduce an organization's evolution risk.
5. Its capability extends CASE toolsets.
6. It makes software easier to change.

Software re-engineering is not completely automated, tools process the complex Re-engineering. These can be used to help in moving a system to a new maintenance environment [8]. This is the activity that performs only by the human being. Success in software re-engineering requires more than tools. Few steps to be considered while planning Re-engineering project: Justification to project, it determine that which system enhanced by business value; Analysis, the application Re-engineered based on quality and value of business; Cost estimation, it is used for projects to estimate the cost; Cost-benefits Analysis, Costs and expected return are compared, and; Contracting, it determine the task identification and effort distribution.

3.9.1 Reverse engineering

Reverse engineering can be defined as "the process of analyzing a subject system to identify the system's components and their relationship and to create system in another form or at a higher level of abstraction" [9].The reverse engineering has its origins in the hardware. The companies want to understand the design of a hardware product for effort "secrets". These secrets can be understood easily when the design specification were obtained. But, during reverse engineering these documents are unavailable to the company.Succrssful reverse engineering derived more than one design and manufacturing specification to produce actual exiting of the product.

Reverse engineering for software is similar as hardware. In most cases, the program in reverse engineering are not competitor's but often done by company many years earlier. Therefore, reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code. Reverse engineering is a process of design recovery. Reverse engineering tools extract data, architectural, and procedural design information from an existing program.

3.9.2 Forward engineering

In an ideal world, using a automated "Reengineering engine", application can be rebuilt. The existing system fed into the engine, analyzed, re-structured and generated in a form of software quality. In the short term," engine" appear, but CASE included the limited subset that application are implemented using a specific database system. The tools are increasly used in reengineering. It is also called, Reclamation or renovation [9]. Forward engineering not only covers the information of design from old software, but uses its information to modify the existing system in an effort to enhance or increases the overall quality. It also used to improve the performance, by generating additional new functions.

3.10 Restructuring

Software Restructuring used to change the source code or data to ensure that the changes applicable to the future or effort can be handle by software [9]. In general, Restructuring does not change the overall program architecture. Restructuring mainly focus in the design detail rather than individual modules and structure within modules. If the Restructuring effort becomes beyond of its boundaries, then restructuring and software architecture become forward engineering. The benefits, when software is restructuring:

1. Higher quality has programs with better documentation, modern standards and less complexity.
2. Improving productivity, it makes case of learning and conflict among software engineers must be reduced.
3. Effort required to perform maintenance activities is reduced.
4. Easier to test and debug the software.

Restructuring occurs when the basic architecture of an application is solid, even though technical internals need work.

3.10.1 Code Restructuring

Code Restructuring is used to performed the same function of design that produces the higher quality than the original system. In general, code restructuring is a program logic using Boolean algebra applies a rule transformation that restructuring the logic [9]. The objective to produce the structural programming and other tools proposed by using restructuring techniques. A *resource exchange diagram* maps each program module and the resources (data types, procedures and variables) that are exchanged between it and other modules. The program architecture can be restructured to achieve minimum coupling among modules.

3.10.2 Data Restructuring

Before data restructuring can begin, a reverse engineering activity called analysis of source code must be conducted. All programming language contains data definition, interface descriptions, input/output and file descriptions are evaluated. The goal is to explore the data items, to get information on data flow, so that existing data structure can be understand in detail that have been implemented. This activity is sometimes called data analysis [9]. If the data analysis completed, then redesign of data committed. When restructuring moves beyond standardization and rationalization, physical modifications to existing data structures are made to make the data design more effective. This may mean a translation from one file format to another, or in some cases, translation from one type of database to another.

4. REFERENCES

[1] Bennett, K. H., "Legacy Systems: Coping with Success", IEEE Software, 12(1):19-23, 1995.

[2] Bennett, K. H., Ramage, M., Munro, M., "Decision Model for Legacy Systems", IEE Proceedings on Software, 146(3):153-159, 1999.

[3] "Challenges during Software product maintenance" Deepak Kumar, Parul 1 Assistant Professor, Directorate of Distance Education, Kurukshetra University Kurukshetra 2Shri Baba Mastnath Engineering College, Rohtak.

[4] International Journal of Technical Research and Applications e-ISSN: 2320-8163, www.ijtra.com Volume 2, Issue 6 (Nov-Dec 2014),"DEVELOPMENT OF A SOFTWARE MAINTENANCE COST ESTIMATION MODEL" 4TH GL PERSPECTIVE Mohammad Islam1, Dr. Vinodani Katiyar2 1Research Scholar, Shri Venkateshwara University,Gajraula, UP, India 2Professor, SRM, University, Lucknow, UP, India 1mohdislam3@gmail.com, 2drvinodani@gmail.com.

[5] "MAINTENANCE ISSUES IN SOFTWARE ENGINEERING" Praveen Chandra Kidambi Department of Computer Science Louisiana Tech University.

[6] Omnext white paper, March 2010 "How to save on software maintenance costs".

[7] "Problems and issues in application software maintenance management" Prashant Palvia the University of Memphis, Aaron Patula University of Minnesota, John Nosek Temple University.

[8] "Software Maintenance and Evolution: a Roadmap" K. H. Bennett V.T Rajlich Research Institute for Software Evolution Department of Computer Science University of Durham Wayne State University UK Detroit, MI 48202 DH1 3LE USA.

[9] Software Maintenance Gerardo Canfora and Aniello Cimitile, cimitile@unisannio.it University of Sannio, Faculty of Engineering at Benevento Palazzo Bosco Lucarelli, Piazza Roma 82100, Benevento Italy

[10] Top ten ways to reduce software costs PRACTICAL TIPS FOR SOFTWARE ASSET MANAGEMENTTop ten ways to reduce software costs PRACTICAL TIPS FOR SOFTWARE ASSET MANAGEMENT.

[11] W. Li and S. Henry. Maintenance Metrics for the Object Oriented Paradigm. In IEEE Proc. of the 1st Int. Sw. Metrics Symposium, pages 52{60, May 1993.