A Review Of Mitigating Cost Concerns To Enhance Maintainability in SDLC

Najamul Saqib¹, Mir Mohammad Yousuf²

¹ Research Scholar, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, India,
²,4 Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, India.

Abstract -- Software maintenance is an indispensable phase of software development life cycle and incurred the maximum effort amongst all stages. The work maintainability relies on planning and execution of adjacent phases. Software maintenance is a never-ending process from the deployment of software until it is no longer produced or used. A wide variety of models are used by software maintenance personals for estimating the cost of software. To make any decision and benchmark these models need proper and complete information. There are various problems and issues that are faced during maintenance such as size of database, maintenance budget, size of system, complexity of software, size of staff etc. This paper proposes various ways for reducing cost in software maintenance. If the design, documentation and software architecture are explained clearly software maintenance costs can be reduced significantly.

Keywords: software maintenance, factors affecting maintenance, Techniques to reduce maintenance cost.

I. INTRODUCTION

Maintenance means to retain any condition or situation. In order to make any system work perfectly maintenance is needed. Software products also need maintenance like other things and objects in use. Different types of enhancements are needed to make software systems work successfully as life, needs and the operating environments always change and also client’s requirements and mind always change.

The phase of system post-delivery where the system need continuous modifications is called software maintenance. This is done to enhance the performance and correct the errors. software maintenance is a process of developing a system after it has been delivered (Ian Sommerville) [1] and ISO/IEC 12207 defines software maintenance as one of the phases of software development life cycle [2]. ISO/IEC 14764 All activities that provide cost-effective support to software system [3]. The activities such as modification is carried out in post-delivery phase and maintenance planning is performed in pre-delivery phase. According to ISO, problem that requires improvement software maintenance is used to check documentation and modification for that.

The various types of the maintenance are

i. Corrective maintenance: Modifications performed while detecting bugs and errors in a system to make system work efficiently.[8][23]

ii. Perfective maintenance: modifications performed to improve software system functionalities, abilities and performance.[8][23]

iii. Adaptive maintenance: Enhancement of software in changing environment to keep it usable after delivery.[8][23]
iv. **Preventive maintenance**: prevention and correction of faults that might become effective faults in a software product after delivery. [8][23]

The importance of software maintenance can be discussed in the below mentioned parameters

i. **Cost of maintenance**: Compared to the early development phases studies have shown that maintenance of software system is very costly and takes about 70% of total cost of project. The most important phase in development is software maintenance as shown in Fig (1).[4]. Testing is covered in programming and integration phases.

![Fig 1. Estimated relative cost of development software phases (Schach (1999)).](image)

Some studies have found that major software systems maintenance contributes 80% of total project cost. It is also reported that software maintenance cost has increased by 50% over past Three decades [5] as shown in Fig (2).

![Fig 2. Development of maintenance cost for software as a percentage of total cost (Floris and Harald, 2010).](image)

i. **Distribution cost maintenance**: Different types of maintenance phases have distributed cost with different percentages. Corrective maintenance takes 65% out of total maintenance cost (Ian Summerville) as shown in Fig 3.
Maintenance effort distribution.

Fig 4. Shows more important metric of maintenance distribution.

There is an immense need of the software maintenance due to the following reasons

i. Error fixing: Error fixing is significant aspect of software maintenance. A smooth run of software is important. It should be handled on preference basis. This method includes detection and correction of errors in code. These errors arise in operating systems, hardware or any part of software. Without compromising other software functionalities, it needs to be done [8].

ii. Enhancement of capabilities to improve environment: Improving the various functionality and features to make system more adaptable to changing environment. Various aspects that affect systems smooth workflow are enhanced such as software tools, operating patterns, compilers etc. Enhances the system performance using advanced technological solutions and services daily [8].

iii. Remove incomplete functions: Functions that occupy space and are not in use anymore disrupts the systems performance. Therefore, it is necessary to remove incomplete functions. Using latest tools and technologies, these interface and coding components are removed and replaced with new features [8].

iv. Improvement of performance: To deal with new demands system performance is improved. Re-engineering as well as software and coding limitations are a part of software maintenance. It avoids bugs in solution. This is not an operational capability, but exists to avoid harmful activities such as hacking [8].

II. LITERATURE SURVEY

<table>
<thead>
<tr>
<th>S. No</th>
<th>Title</th>
<th>Findings</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Software Maintenance practical: Practice for managing Software investment.</td>
<td>The author has observed that work done in software maintenance is less as seen in software development. Also, literary works does not illustrate the maintenance phase. The day-to-day services provided by software maintainers are based on different situations and functionalities. Some maintenance related activities are also defined by international standards organization Such as SEEBOK.</td>
<td>Pigoski et al. (1996).</td>
</tr>
<tr>
<td></td>
<td>Software Maintenance Maturity Model (SM\textsuperscript{mm}): The Software Maintenance Process Model.</td>
<td>Author have covered various internal and external issues regarding software maintenance from cost and effort viewpoint respectively, that involves high maintenance cost, poor maintenance service. Poorly programmed and designed software, prioritizing request for change are included in effort point of view. In fact, proper documentation is lacking there.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Maintenance: A Tutorial.</td>
<td>The author has listed different challenges faced by software maintainers in current scenario’s such as current challenges in strategic unity, technical and operational challenges etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Maintenance Effort Estimation: A Neural Network Approach.</td>
<td>This paper has observed gaps in the cost estimation model of software and its significance. It has categorized various factors that comes under cost estimations such as size, effort, project span etc. The paper have also listed various causes that leads to maintenance problems such as use of model for estimating cost of software, calculation of LOC, appropriate estimation etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Software Maintenance”, Article.</td>
<td>This article has defined maintenance activities as an ocean to describe and showcased herculean problems and costs related to it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamics of Software Maintenance.</td>
<td>The author has stated that it is uncontrollable task to evaluate and effort with high accuracy. In addition to that he has found certain factors by which accuracy can be achieved such as complexity and functionality of system. The author has also explained that maintenance phase is relatively different from development phase and inherit different features and emphasis much on estimation models.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Research on Software Maintenance Cost of Influence Factor Analysis and Estimation Method.</td>
<td>The author has explained different kinds of software maintenance and examined the effects of various technical and non-technical factors on software cost. In addition, provided more advanced version of cost estimation model from basic model and identified cost factors that affect maintenance load.</td>
<td></td>
</tr>
</tbody>
</table>

April et al. (2005)  
Bennett et al. (2000).  
Shukla et al. (2008)  
Gerardo Canfora et al. (2010)  
Pankaj Bhatt et al. (2004)  
Y. Ren et al. (2011)

The researcher has performed work on estimating the effort of component-based software development. They have taken many things into account such as estimation by researchers, degree of acceptance, data used by them and life cycle activities performed with any validation. The objective is to provide better understanding of this approach.

**T. Wijayasiriwardhane et al. (2011)**

### 10. Which Factors Affect Software Projects Maintenance Cost More?

The author has identified various factors that are responsible for software maintenance cost and listed them according to their order of priority and suggested some ways to reduce the cost. 32 factors have been investigated and further categorized into six types with their priorities. The author has proposed that maintenance cost can be reduced and software reliability can be enhanced by taking account of key elements such as proper documentation, proper functionality of IT projects and encouraging designers in maintenance phase.

**S. M. H. Dehaghani et al. (2013)**

### III. COST CONCERNS OF MAINTENANCE

There are number of cost concerns that affect software maintenance. These are categorized in two types: Technical and Non-technical [16].

#### Technical concerns:

i. Software complexity: More time is required to update and read a code if more complex software structure is there, resulting in increased maintenance load [16].

ii. Advancement in Human Ability: The load of maintenance can be reduced by complete analysis of system. If the programmers have good knowledge of programming then they are able to create easily understood and read software. Programs written in high-level languages are more easily understood than programs written in low-level language. Also programming styles or methods contributes to its understandability and thus, its ease of modification. Advanced testing methods leads to reduction of software errors [16].

iii. Documentation quality: Maintenance staff must provide complete description of software system in a document which helps other people to know the system in a better way. This will also help in meeting changes in users’ requirements or system environment and detection and correction of errors to improve the system. If documentation quality is simple, complete and precise, maintenance cost will be reduced [16].

iv. Managing configuration techniques: One of the important ways of maintenance cost is to keep record of all system documents and making sure they are accurate. It comprises of identifying configuration,
progress reports, modification control, program configuration etc. Effective management of configuration techniques helps in reducing maintenance costs [16].

v. Quality of components: In component-based software’s, quality depends on the individual software’s performance. Minimum performance should be expected when some component is integrated or adapted or incorporated into a system [16].

Non-technical concerns:

i. Application experience: When system requirements are well specified through software development application less maintenance is needed. Furthermore, when software is in new area and requirements for adaptive maintenance changes enormously, it becomes very difficult to understand the demand [18].

ii. Staff Stability: For maintenance of any component developer is responsible and if no more software engineer needs to waste more time to learn the full code then maintenance cost will be reduced. Maintenance cost will also reduce if implementation of any system maintains that system [18].

iii. Application Lifetime: In software applications, programming errors occur when there is a greater modification of maintenance load is concentrated [19]. Maintenance becomes more difficult when previous practices are used to run the software and are unable to control the integrity and flexibility of huge load resulting is maximum maintenance cost.

iv. External Environment: If a software depends on its external environment, it needs to be changed as climate changes. For example, business rules, workflow, reports etc. are some of the external environment or some of the soft environment. The exterior environment must be made appropriately overdependence software if environment changes.

v. Customer’s needs: Customer is the main body of understanding software. When software is running its functionality and performance must be better and for better performance and functionality requirements must be better, leading to increased maintenance load [20].

IV. FUTURE TECHNIQUES TO REDUCE MAINTENANCE COST

Here are various techniques to reduce future maintenance cost. These techniques either improve the maintenance or provide strategies for development to reduce cost of software maintenance in future. These strategies include defensive programming. Software re-engineering and restructuring, software reverse engineering, usage of top languages and code producers for software development and designing software with less error. Although re-engineering, restructuring and defensive programming are used to enhance the maintainability of currently existing programs, others are used for strategic development to produce effective programs. These techniques are described as [21].

i. Software re-engineering and restructuring: Programs that are unstructured are difficult to maintain and understand. As compared to structured programs they have less complexity. Such structures contain tightly connected modules and cannot be separated. Programmers must go through the entire application to understand unstructured programs.

Program restructuring includes splitting programs into top-down order of smaller modules with conditional, iterative and simpler construct sets. Every module has one entry and exit point. These are number of automated
CASM tools available of software restructuring such as Cobol Structuring Facility (IBM), Record (Language Technology), superstructure (Group Operations, Inc.) and Retrofit (Catalyst, Peat Marwick) [21][22]. These tools help in reducing maintenance cost that are result of unstructured programs and arise from weak programming methods and maintenance performed under pressure. Restructuring allows modification easy and reduces maintenance cost [21].

Software re-engineering is defined as modification of code and data structures. Re-engineering is larger than re-structuring software. It is used to achieve both corrective and enhancement maintenance [21].

ii. **Software reverse engineering**: It enables programmers to code and use automated tools in the way they like. It may be difficult for programmers to write correct codes and adhere software engineering principles. Existing programs and their description of databases and conversion of design from implementation to configuration level are included in reverse engineering. There are CASE tools integrated with experts’ systems for assistance in reverse engineering [21].

iii. **Defensive programming**: Glass and Noiseux suggested an method called as defensive programming for enhancing maintainability of codes. They identified five programming practices for error detection in future such as use of assertions, flagging harmful practices, test trails, margins, commentary. These practices help to understand and test the system. These represent preventive maintenance and reduce future maintenance cost [21].

V. **CONCLUSION**

Software maintenance represents up to 90% of entire lifetime of software development cycle and incur maximum effort among all phases. Organizations are unaware of importance of software maintenance. To reduce maintenance cost in long run better development tools and techniques are needed. This paper discusses various practices to enhance software maintainability such as software re-engineering and restructuring, reverse engineering and defensive programming. These practices can be performed by an organization in future and still they are working on them. Future efforts are therefore needed for developing maintenance techniques that integrate several facilities.

VI. **REFERENCES**


[5]. Floris P, Vogt Harald H. How to save on software maintenance costs, Omnext white paper, SOURCE 2 VALUE, 2010


