

Applying Project Cost Management to Software Maintenance Projects

Lakshminarayanan Ramanujam, PMP

Computer Sciences Corporation India Pvt Ltd



Leveraging project management for excellence, growth and transformation



Contents

1.1	Abstract	3
1.2	Keywords	3
1.3	Introduction	3
1.4	Main body of the paper	4
1.4.1	Maintenance task types	4
1.4.2	Legacy code amount	4
1.5	Key challenges	5
1.6	Methodology/process followed	5
1.6.1	Activities	7
1.7	Critical success factor	8
1.8	Quantified benefits to business	8
1.9	Lessons learnt	8
1.10	Conclusion	8
1.11	References	9
1.12	Author(s) Profile	11



1.1 Abstract

Software maintenance is the modification of a software product after delivery to correct faults (fix bugs), to improve performance or other attributes, or to adapt the product to a modified environment.

In other words “the process of modifying existing operational software while leaving its primary functions intact”. It is different from hardware maintenance due to the fact that software does not experience any physical wear and tear

Annual software maintenance cost in USA has been estimated to be more than \$70 billion (1995) Enormous amount of time and effort being spent on maintenance of software, In 1990 there were an estimated 120 billion lines of source code being maintained. There are at least 200 billion lines of COBOL-code still existing in mainframe computers alone (Gartner Group). Maintenance of software is highly expensive and complex.

[Jones02] states that, “In 2001 more than 50% of the global software population was engaged in modifying existing applications rather than writing new applications” Cost of software maintenance is estimated at 50% of the total life cycle cost; (Van Vliet [2000]) savings in this area is bound to have a huge impact.

This paper is an attempt to apply the project management principles, particularly, the project cost management methods/tools to software maintenance projects, and further suggest exploring the models to accurately estimate software maintenance cost.

The main aim is to insist the project cost management tools/methods (in particular EVM) to software maintenance projects using various maintenance process/models, in order to manage/control the same.

1.2 Keywords


EVM, software characteristics, Quality attributes of software, Software Maintainability Index, Cyclomatic complexity

1.3 Introduction

Martin and McClure [MM83] define software maintenance as, “Changes that have to be made to computer programs after they have been delivered to the customer or user.” It is necessary to estimate maintenance effort and costs as part of the planning process. Maintenance costs amount for a significant portion of the overall project life-cycle cost). Moreover estimate of maintenance effort helps to plan adequate maintenance staff.

Maintenance plays an important role in the life cycle of a software product. It is estimated that there are more than 100 billion lines of code in production in the world. A majority of it is unstructured, patched and not well documented. Maintenance can alleviate these problems.

However, estimating the costs for the Software maintenance projects are complex, time consuming, lacks accuracy, though there are different models available to



estimate the effort required to maintain software, depending on the technical characteristic of the software like syntactic, semantics, architecture, languages used etc the appropriate model to be applied, also different methodologies, like water fall, spiral, agile, scrum, etc. Each aspect plays important role in estimating the effort required to maintain.

Another aspect that needs to be studied is the process of maintaining the software, Also, software maintenance depends on various quality attributes of software like maintainability, analyzability, changeability, reliability, efficiency, etc. we will focus on maintainability to make certain decisions on software maintenance.

In general there are 4 types of maintenance to be performed on software, corrective, adaptive, perfective & preventive, Depending on the characteristic of software, the type of maintenance performed will vary, we will focus on corrective maintenance.

We will be using the IEEE 1219 software maintenance standard for identifying the tasks involved in corrective maintenance, apply EVM (earned value method) for the identified tasks.

1.4 Main body of the paper

Few interesting facts on software maintenance cost

- ◆ Annual software maintenance cost in USA has been estimated to be more than \$70 billion (Sutherland, 1995; Edelstein, 1993).
- ◆ E.g. in USA, the federal government alone spent about \$8.38 billion during a 5-year period to the Y2K-bug corrections.
- ◆ At company-level, e.g. Nokia Inc. used about \$90 million for preventive Y2K-bug corrections.

1.4.1 Maintenance task types

- ◆ About 65% of maintenance was found to be perfective by Lientz & Swanson (1981).
- ◆ About 75% of maintenance costs are spent for providing enhancements (in the form of adaptive and perfective maintenance) (Martin, 1983; Nosek & Palvia, 1990; van Vliet, 2000).
- ◆ Studies of software maintainers have shown that approximately 50% of their time is spent in the process of understanding the code that they are to maintain (Fjeldstad & Hamlen, 1983; Standish, 1984).

1.4.2 Legacy code amount

- ◆ In 1990 there were an estimated 120 billion lines of source code being maintained (Ulrich, 1990).
- ◆ In 2000 there are already about 250 billion lines of source code being maintained, and that number is increasing (Sommerville, 2000).
- ◆ An average Fortune 100 company maintains 35 million lines of code (Müller et al., 1994).

- ◆ These companies add in average 10% each year only in enhancements (Müller et al., 1994).
- ◆ As a result, the amount of code maintained doubles in size every 7 years (Müller et al., 1994).
- ◆ Older languages are not dead. E.g. 70% or more of the still active business applications are written in COBOL (Giga Information Group).
- ◆ There are at least 200 billion lines of COBOL-code still existing in mainframe computers alone (Gartner Group).

1.5 Key challenges


Several technical and managerial problems contribute to the costs of software maintenance. Few of the management challenges are justifying the ROI, staffing, standardizing the process across the organization etc. The other technical challenges are effort estimation, limited understanding, non availability of documentation, performing impact analysis and testing. For example, whenever a change is to be made to a piece of software, it is important that the maintainer gains a complete understanding of the structure, behavior and functionality of the system being modified. It is on the basis of this understanding that modification proposals can be made. As a consequence, maintainers spend a large amount of their time reading the code and the accompanying documentation to understand its logic, purpose, and structure. Available estimates indicate that the percentage of maintenance time consumed on program comprehension ranges from 50% up to 90%. When there is such a huge variance, the estimations may not be accurate. Code comprehension is frequently compounded because the maintainer is rarely the author of the code. Software designed with maintainability in mind greatly facilitates impact analysis.

1.6 Methodology/process followed

The approach is a combination of software engineering tool (SMI – Software Maturity Index) and project cost management tool (EVM) Maintenance effort estimation techniques range from the simplistic level of effort method, through more thoughtful analysis and development practice modifications, to the use of parametric models in order to use historical data to project future needs

We have chosen specifically software maintenance project as complexity is high when compared with new software development projects, moreover, models/methods are available to make relatively good estimate of the effort of a new development project like FPA (Function Point Analysis) , COSMIC functional size, use case points, etc which cannot be directly applied to software maintenance projects. Maintenance is important especially in case of software which has long life, large, complex and critical to customer.

Software maintenance is often quite demanding (despite a common view of it being routine work) especially in case of maintaining large legacy system, these software systems are typically hard to maintain, but cannot be replaced because of their great business value and application domain knowledge that they contain, thus they tend to have long life. Software maintenance is further complicated in case of large tasks of adaptive maintenance, that is modifications made due to changing technical and functional environments.



In our approach, we firstly review using software engineering methods/tools to decide whether it is economically viable to maintain a given software or not, and then to measure the size., later use project cost management toll EVM (Earned Value Method) to estimate the cost. A good example of importance of maintenance is Y2K bug, which is said to be the single most expensive maintenance done in the history.

Software life time & rewrite strategies - One of the central questions is in making a well informed and correct decision regarding whether to undertake software maintenance.

Any software development organization has vital interest in reducing the spending on software maintenance and this is not a surprise as bulk of the life cycle cost is not consumed by new software development but spent on software maintenance that is in continuous up keeping, adaption and bug fixing of existing software. Many software systems that are deployed in large companies today are 20 years or older.

In general there are 4 types of maintenance to be performed on software, corrective, adaptive, perfective & preventive, Depending on the characteristic of software and the requirements the type of maintenance performed will vary.

There are various types of processes or models (more and more studies are being conducted on this as the process is difficult & complex) to address maintenance of software. Quick fix model or ad-hoc approach, Boehm's model (based economic models), Osborne's model (based on metrics), iterative enhancement model (based on iterations) and reuse oriented model (based on reuse of software)

All the four types of maintenance are applicable to software maintenance, however depending on the characteristic of the software, maintenance process used may differ.

Metrics based approach to measure maintainability:

In software engineering, the ease with which a software product can be modified in order to: * correct defects * meet new requirements * make future maintenance easier, or * cope with a changed environment; these activities are known as software maintenance (cf. ISO 9126).

The software maturity index (SMI) provides an indication of the stability of a software product (based on the changes that occur for each release of the product). The following information is determined:

MT = the number of modules in the current release

Fc = the number of modules in the current release that have been changed


Fa = the number of modules in the current release that have been added

Fd = the number of modules from the preceding release that were deleted in the current release

The software maturity index is calculated in the following manner:

$$SMI = [MT - (Fa + Fc + Fd)] / MT$$

As SMI approaches 1.0, the product begins to stabilize. SMI may also be used as metric for planning software maintenance activities. The mean time to produce a release of a software product can be corrected with SMI and empirical models for maintenance effort can be developed.



In traditional software cost models, costs are derived simply based on required effort. Empirical estimation models provide formula for determining the effort based on statistical data of similar projects.

One of the quality attributes of software system is maintainability, A metrics based approach helps to arrive at project effort and cost estimates, Especially in software maintenance projects, the arriving at the effort is a very complex task, There are several variable that contribute to the effort estimation, more over several attempts have been made to estimate the project effort/cost estimates. If maintainability is low than the cost of maintaining is high and similarly the cost will be less (when compared with maintainable software) for software for which the maintainability is low.

Decisions on whether to go for maintenance or for new development, similarly decisions on whether to release a soft product/application can be made based on maintainability.

LOC (Lines of Code), McCabe's Cyclomatic complexity, software maintenance Index, decomposition or *Factor-Criteria-Metric* (FCM) approach are few metrics that pertains to maintainability. These metrics help in making a decision whether to maintain software or go for new development, based on the cost of ownership. Also software releases can be decided based on software maintainability Index.

Earned Value Management (EVM) helps project managers to measure project performance. It is a systematic project management process used to find variances in projects based on the comparison of worked performed and work planned. EVM is used on the cost and schedule control and can be very useful in project forecasting.

Application of project cost management tool (EVM) "Includes the processes required to ensure that the project is completed with the approved budget" - PMBOK Guide .


EVM when used in projects have several benefits like Preventing scope creep, Improving communication and visibility with stakeholders, Reducing risk, Profitability analysis, Project forecasting, Better accountability, Performance tracking .

[Walter H. Lipke] The progress reporting, using the EVM indicators, is an excellent method of providing project status. It portrays to the customer's measures of cost, schedule, and technical performance in a very concise, understandable, and meaningful way. The indicators from EVM, cost performance index (CPI) and schedule performance index (SPI), provide a means for statistically managing the software engineering process., The use of CPI and SPI was evolved into statistical applications. The applications are useful in predicting project outcomes and have been shown beneficial in project planning.

In order to apply EVM, the project activities are to broken down as tasks, the activities/process are based on the IEEE STD 1219 & ISO/IEC 14764 - for software maintenance projects.

1.6.1 Activities

- ◆ Modification request
- ◆ Classification & Identification
- ◆ Analysis
- ◆ Design

- 
- ◆ Implementation
 - ◆ System testing
 - ◆ Acceptance testing
 - ◆ Delivery

The efforts for the above activities are to be assigned based on empirical data or based on the application characteristic

1.7 Critical success factor

Identifying the correct WBS and accurate effort estimation are the key success factor for software maintenance project, Also, understanding process variations and improving overall process capability. The next level of quantitative management where empirical methods are used to establish process predictability, thus enabling better project planning and management.

1.8 Quantified benefits to business


The integration of work, schedule, and cost using a Work Breakdown Structure, The cumulative Cost Performance Index as an early warning signal. The Schedule Performance Index as an early warning signal. The Cost Performance Index as a predictor for the final cost of the project. Thus better project measurement & control.

1.9 Lessons learnt

The process and the tasks vary for different types of software maintenance - corrective, adaptive, perfective & preventive, Choosing a model or approach that works for one type may not work with other, for example, the tasks relating to the corrective maintenance is different than the perfective maintenance, the process for perfective is more like development. Similarly the tool also varies depending on the type of maintenance, Projects should carefully evaluate and use the right approach for the type of maintenance.

1.10 Conclusion


One of the most critical problems that project managers encounter is the management of the performance on their project; this problem is multifold with respect to software maintenance projects, due to the characteristics of the software. EVM is a process that has been used for years by organizations to measure performance and health of the project. It has been embraced by project managers around the globe with good success. To apply EVM successfully, the effort estimates needs to be accurate. Due to complex characteristics of software, the effort estimations with respect software maintenance projects may not be accurate and the variance is high. Though there are several estimation models available, the models, tools need to be carefully evaluated depending on technical characteristics of the software. While software maintenance index can be used to make a release decisions, it can also be used as an indication to decide on financial viability of maintenance of the software maintenance projects, depending on the availability of the data. The earned value method can be applied to software maintenance projects, using the activities/tasks based on the IEEE 1219



standard for corrective maintenance.. However the effort estimation of the maintenance activity depends on the quality attributes of the software like maintainability, analyzability, changeability, reliability, efficiency, etc and also the technical attributes like architecture, tools/methodology etc.

1.11 References

1. Gerardo Canfora and Aniello Cimitile (2000) Software Maintenance
2. Eastwood, A. (1993). "Firm fires shots at legacy systems". Computing Canada 19 (2), p. 17.
3. Edelstein, D. (1993). "Report on the IEEE STD 1219 – 1993 – Standard for Software Maintenance". ACM SIGSOFT Software Engineering Notes 18 (4), p. 94.
4. Erlikh, L. (2000). "Leveraging legacy system dollars for E-business". (IEEE) IT Pro, May/June 2000, 17-23.
5. Fjeldstad, R. & Hamlen, W. (1983). "Application program maintenance-report to our respondents". Tutorial on Software Maintenance, 13-27. Parikh, G. & Zvegintzov, N. (Eds.). IEEE Computer Soc. Press.
6. Jussi Koskinen, Henna Lahtonen, Tero Tilus (2003) SOFTWARE MAINTENANCE COST ESTIMATION AND MODERNIZATION SUPPORT ELTIS-project
7. Lientz, B.P. & Swanson, E. (1980). "Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations". Addison-Wesley: Reading, MA, 214 p.
8. Lientz, B.P. & Swanson, E. (1981). "Problems in application software maintenance". Communications of the ACM 24 (11), 763-769.
9. Lipke, Walter H. and Mike Jennings,(2000) "Software Project Planning, Statistics, and Earned Value," **CrossTalk**, December 2000: 10-14.
10. Lipke, Walter H., (2002) "Statistical Process Control of Project Performance," **CrossTalk**, March 2002: 15-18.
11. Martin, J. (1983). "Software Maintenance: The Problem and Its Solution". Prentice Hall, 472 p.
12. McKee, J. (1984). "Maintenance as a function of design". Proceedings of the AFIPS National Computer Conference, 187-193.
13. Moad, J. (1990). "Maintaining the competitive edge". Datamation 61-62, 64, 66.
14. Müller, H., Wong, K. & Tilley, S. (1994). "Understanding software systems using reverse engineering technology". The 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences Proceedings (ACFAS), 26 (4), 41-48.
15. Nosek, J. & Palvia, P. (1990). "Software maintenance management: changes in the last decade". Journal of Software Maintenance: Research and Practice 2 (3), 157-174.
16. Port, O. (1988). "The software trap – automate or else". Business Week 3051 (9), 142-154.

- 
17. Robert A. Hanna, (2009) "Earned Value Management Software Projects," smc-it, pp.297-304, Third IEEE International Conference on Space Mission Challenges for Information Technology
 18. Seacord, R., Plakosh, D. & Lewis, G. (2003). "Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices" (SEI Series in Software Engineering). Addison-Wesley..
 19. Standish, T. (1984). "An essay on software reuse". IEEE Transactions on Software Engineering SE-10 (5), 494-497.
 20. Sutherland, J. (1995). "Business objects in corporate information systems". ACM Computing Surveys 27 (2), 274-276.
 21. ISO/IEC 14764:2006 Software Engineering — Software Life Cycle Processes — Maintenance
 22. IEEE Computer Society – SWEBOK – software engineering body of knowledge (www.swebok.org)
 23. Burt Swanson, The dimensions of maintenance. Proceedings of the 2nd international conference on Software engineering, San Francisco, 1976, pp 492 — 497
 24. Abran, Alain (2008). Software Maintenance Management. New York: Wiley-IEEE. ISBN 978-0470-14707-8
 25. Gopalaswamy Ramesh; Ramesh Bhattiprolu (2006). Software maintenance : effective practices for geographically distributed environments. New Delhi: Tata McGraw-Hill. ISBN 9780070483453.
 26. Grubb, Penny; Takang, Armstrong (2003). Software Maintenance. New Jersey: World Scientific Publishing. ISBN 9789812384256.
 27. Lehman, M.M.; Belady, L.A. (1985). Program evolution : processes of software change. London: Academic Press Inc. ISBN 0-12-442441-4.
 28. Page-Jones, Meilir (1980). The Practical Guide to Structured Systems Design. New York: Yourdon Press. ISBN 0-917072-17-0.
 29. PMBOK_Guide_Fourth_Edition, Project Management Institute

1.12 Author(s) Profile



Lakshminarayanan Ramanujam PMP, CISA, CISSP is currently working as senior project manager with Computer Sciences Corporation India Ltd, have over 12 years of project management experience in IT application & Infrastructure projects. He is active member of PMI, ACM, IEEE computer Society, ISACA, ISC2, DSCI.

He has earlier worked with IISc, HAL, Digital Equipement Corporation, Silverline Technologies. He specialises in application maintenance, Infrastructure and information security projects.

E-mail: lramanujam@csc.com