December 1999

Software Evaluation and Maintenance Cost,

Hsiang-Jui Kung
Ministry of National Defense, Hui-LienTung

Hui-Chung Chu
HuaFan University

Follow this and additional works at: http://aisel.aisnet.org/amcis1999

Recommended Citation
http://aisel.aisnet.org/amcis1999/257

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 1999 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Software Evaluation and Maintenance Cost

Hsiang-Jui Kung, Ministry of National Defense, kungh@tpts5.seed.net.tw
Hui-Lien Tung, University at Albany, ht1628@cnsvax.albany.edu
Hui-Chung Chu, HuaFan University, hcchu@huaan.hfu.edu.tw

Abstract
This paper tries to explore the relationship between software evaluation and maintenance cost. A research model and hypotheses are proposed. Field study methodology will be used to collect data. Future research is needed to validate the research model.

Problem Statement
Currently the global software industry has an annual sales of $300 billion (DeMarco, 1993). Software maintenance accounts for 60 - 80 percent of the budgets (Bohner, 1993). The misgiving that people get from this is that since maintenance accounts for a large chunk of the software cost, perhaps a better rationale would be to build software better in the first place and save the maintenance dollars.

The quality issues of software remain critical, as their impact on organizations, particularly in life-critical areas, becomes significant (Vollman, 1993). Although it is still controversial whether we should consider software as products or services, before any software system can be implemented, it is subjected to a evaluation process. An evaluation is generally performed from two different viewpoints - users (including manager), and developers (including internal audit) (O'Keefe, 1989).

Recently researchers have tried to link maintenance to quality assurance (Biggerstaff, 1993). The intention of this research is straightforward. There is a substantial literature that investigates software evaluation approaches or maintenance. Since software evaluation and maintenance are two important phases of software life cycle, this research is to study the relationship between evaluation and maintenance. The research problem is that:

Do software evaluation approaches correlate with maintenance cost?

Background
The impact of management information systems (MIS) within business and society as a whole, has increased enormously in the last two decades (Curtis, 1992). Automated systems have been used in many organizations. When computer systems fail because of software faults, the results can be nightmarish: traffic jams, medical misdiagnoses, botched accounting and inventory lists. Therefore it is critical that both software engineers and users need to evaluate software quality.

Software effectiveness can be evaluated from two levels: (a) developer efficiency; i.e., how well the developer in providing the system, and (b) user/organizational effectiveness; i.e., the success of accomplishing the organizational mission (Hamilton & Chervany, 1981). Software evaluation approaches that have been proposed include software metrics (Schneidewind, 1992), system usage (Lawrence & Low 1993), and user satisfaction (Yaverbaum & Nosek, 1992).

Software metrics are used by software engineers to quantitatively characterize the essential features of software. This aids the application of classification, comparison, and mathematical analysis. After identifying a number of useful metrics, software engineers measure software in an algorithmic and objective fashion. This ensures that the values of the selected metrics are consistent among different software products and are independent of the measurers (Conte, Dunsmore & Shen, 1986).

System usage can be an indicator of system success only when usage is voluntary, or there is no motivations (political, self-protection) for using the system other than its objective utility (Baroudi, Olson & Ives, 1986). The instrument used to measure system usage has three items: hours per week, frequency per week, and percentage of voluntary (Lawrence & Low, 1993). User satisfaction is a subjective measure of system success or effectiveness. Baroudi and Orlikowski (1988) developed an instrument and proved its reliability, content validity, and construct validity.

Software maintenance: Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment (ANSI/IEEE Standard).

Traditionally, software maintenance has been classified into three categories: corrective, adaptive or perfective (Swanson, 1976). All major maintenance efforts are distributed as: corrective (21%), adaptive (21%), or perfective (58%) (Abran & Nguyenkim, 1991).

Researchers today try to link software evaluation to maintenance. Cherinka, Overstreet and Sparks (1993)
proposed an integrated software maintenance environment in which the software maintainers and users worked hand-in-hand to make the maintenance work easier. Glass (1989) suggested that in order to make the transition to maintenance more satisfactory, personnel from maintenance could be represent the quality assurance team during development.

Slaughter (1995) links software design and development practices with software maintenance performance by using code complexity as an intermediate variable. Two organizations were chosen. Both organizations have a large investment in transaction processing applications written in COBOL and running on IBM mainframe computers. About 60-65% of the total application portfolios at the organizations in COBOL was included in Slaughter’s study. The study has shown that code complexity is a significant predictor of software maintenance effort, demonstrated that software design and development practices significantly influence code complexity, and provided empirical evidence to prove the link between software design practices and software maintenance.

Research Model and Hypotheses

The objective of this research is to examine the relationship between software evaluation approaches and annual maintenance cost in order to find a better evaluation approach to predict maintenance cost. The model (Figure 1) for this study tests that certain evaluation approaches influence maintenance costs. Dekleva (1992) suggests that user dissatisfaction is one of the top six software maintenance problems. Szajna (1993) suggests a positive relationship between system usage and user satisfaction, and indicates that practitioners use system usage as a surrogate of system effectiveness. If user dissatisfaction can be reduced (increase user satisfaction), maintenance problems should be lessened. Therefore, we hypothesize that:

H1: Greater user satisfaction will lead to lower maintenance cost.

H2: Higher system usage will lead to lower maintenance cost.

The corrective maintenance ratio of a software system can be an indicator of the software quality (Abran & Nguyenkim, 1991). Software metrics, when defined and used correctly, have been shown to be good indicators of software complexity (Conte, Dunsmore & Shen, 1986). As the complexity level of a software system increase, the code becomes difficult to understand, more likely to contain errors, and therefore more laborious to maintain (Lewis & Henry, 1989). The complexity metrics provide a means of measuring software complexity, and can be used to predict maintenance cost (Gill & Kemerer, 1992). This is investigated in hypothesis 3:

H3: Greater complexity metrics will lead to higher annual maintenance cost.

Methodology

The units to be analyzed in this research are large software application systems that are being used. The large software system has more than 100,000 lines of code excluding comments (Tamai & Torimitsu, 1992). Large implemented software systems will be studied. A method of combining case study and survey will be used in this research. Four data samples per unit will be collected in an one-year period (one data sample per quarter). The time series data collection enables us to focus on the issue of evaluation and maintenance, and overcome the poor internal validity of one shot case study, but at the limitation of external validity.

The variables measured in this study are complexity metrics, user satisfaction, system usage, and maintenance cost. Complexity metrics will be measured by a metric analyzer. User satisfaction will be measured by surveying randomly sampled end users of each software system. System usage and maintenance cost can be collected from archival records. Persons to be interviewed in this study are MIS manager, maintenance team leaders.

Conclusion and Future Research

The research model proposed by this paper provides a framework to better understanding the relationship between software evaluation and maintenance cost. This model needs empirical data to test the hypotheses. Some attempts to study software maintenance process are based on experiments involving small pieces of code, or are based on qualitative case study. Such evidence is valuable, but several researchers have noted that such results must be applied cautiously to the large-scale commercial application systems that account for most software maintenance expenditures (Gibson and Senn, 1989). Software maintenance management is as complex a job as new systems development, perhaps even harder because it covers a longer period of time. The benefit of empirical research has its strength in being fact-based. Due to the nature of software maintenance (evolution), a longitudinal field study combining qualitative and quantitative data collection methods is recommended to provide triangulation validating its findings.

References available upon request from the first author
Figure 1: Research model