8-16-1996

How Software Evolves: An Exploratory Analysis of Software Change Histories

Sandra Slaughter
Greaduate School of Industrial Adminstration, Carnegie Mellon University, sandras@andrew.cmu.edu

Weelin Sim
Graduate School of Industrial Administration, Carnegie Mellon University

How Software Evolves: An Exploratory Analysis of Software Change Histories

Research-in-Progress
Sandra Slaughter
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213
email:sandras@andrew.cmu.edu
and
Weelin Sim
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

Introduction

Software maintenance is an expensive problem for organizations. On a life cycle basis, more than three fourths of the investment in software occurs after the software has been implemented (Arthur, 1988). Thus, there is considerable interest in improving performance in software maintenance. However, software maintenance is an organizational activity that is notoriously difficult to perform and manage effectively (Swanson and Beath, 1989). In particular, it is difficult to plan for maintenance and to manage maintenance requests. In part, this is because managers have very little a priori knowledge about the demand for maintenance work and about the ease with which the work can be accomplished.

If managers could forecast the demand for changes to a system, they could plan and allocate workforce, do more effective change management, and estimate the need for re-engineering or replacement of systems (Martin and McClure, 1983). For example, the need for re-engineering could be signaled by large amounts of corrective maintenance. The need for replacement could be signaled by unusually large numbers of changes as well as growing user dissatisfaction, and increasing maintenance effort. Such indicators are consistent with theories of systems entropy (Checkland, 1981; Weinberg, 1975; Boulding, 1956) and economic models of software replacement (Gode, Barua and Mukhopadhyay, 1990) that predict the point at which it is more costly to update than to re-write the software. However, signals in the form of past patterns of maintenance are not normally made available to managers due to the effort required to accumulate, report and analyze detailed change data.

In this study, we examine changes to software over time. Although the lifetime of installed software can extend across decades, many studies of software maintenance have been cross-sectional in nature due to difficulties in collecting longitudinal data (Kemerer, 1995). With this study, we intend to open the "black box" of software maintenance and gain an understanding of how software evolves. We investigate the following questions:

- what are the antecedents of software change
- are there certain subsets of systems that change most frequently, and if so, why
- do changes to software follow "predictable" patterns over time
- can the point of software entropy or replacement be predicted based upon the number and type of changes

Methodology

To examine these questions, we have collected historical data on the types of changes made to an information systems portfolio in a commercial organization. The data have been extracted from change "histories" that were logged by programmers for each software module in the portfolio. The change histories were kept for 23 different COBOL business systems, including approximately 4,000 software modules, and beginning in the early 1970's (when many of the systems were originally written) until 1993. The kind of data available in the histories includes the original software module creation date and author, the function of the software module, the person making the change, the date of the change, and a description of the change. We have also collected data relating to the size, complexity, and development characteristics of the software modules to assess whether these factors are associated with the likelihood of change.

To identify the number and patterns of changes to the systems, we employ a content analytic approach (Krippendorff, 1980). We have developed a coding scheme that relies on the standard industry categorization for software maintenance activities (ANSI/IEEE, 1983). The coding scheme is applied to the change histories to count the number of changes made to software modules on a monthly basis and to classify the changes according to the standard maintenance categories of corrective, perfective and adaptive work. We have also allowed for finer distinctions within the three main categories. For example, we sub-classify adaptive work into data related and logic related changes.

Because the coding of change histories for thousands of modules is a non-trivial effort, we have selected several systems with rich change histories for our exploratory analysis. Two coders have been given the change history coding scheme and its description. To assess coding reliability, several modules have been randomly sampled from the systems selected for the exploratory analysis. The change histories from these modules were used by the coders to achieve sufficient inter-coder reliability regarding the categories of maintenance work. Initially, the coders independently coded a random sample of two modules each. After the independent coding, the Cohen coefficient of agreement for nominal scales (Cohen, 1960) was computed to assess the relative agreement between the coders. Differences in coding were resolved, and the coders independently coded two additional modules. After another round of independent coding, agreement between the coders was 100 per cent with regards to maintenance categories. Subsequently, the change histories for software systems in the pilot analysis were divided between the coders, and were coded independently.

We analyze the antecedents of software change using correlation and regression analysis. The number and pattern of changes over time for each system will be analyzed graphically and using ARIMA (autoregressive integrated moving average) models. ARIMA models are flexible and are widely used in time series analysis to identify systematic patterns in data (Box, Jenkins, and Reinsel, 1994). The particular form of the ARIMA model for each system will be initially selected based upon the autocorrelation and partial autocorrelation functions, and coefficients of the model will be estimated using maximum-likelihood procedures. Model specification will be checked in multiple ways. Residuals will be examined to ensure that the autocorrelation and partial autocorrelation functions are not significantly different from zero, and the Box-Ljung Q statistic (Ljung and Box, 1978) will be calculated to ensure that the residuals are without pattern. The Akaike information criteria statistic (Akaike, 1974) and the Schwarz Bayesian criterion statistic (Schwarz, 1978) will be calculated to assess goodness of fit for the model. If necessary, alternative forms of the ARIMA models will be estimated and assessed in a similar manner.

Preliminary Results

We have finished the coding for several systems and are in the process of analysis. Our data indicate that software modules that have received corrective maintenance once are highly likely to be repaired repeatedly over their entire lifetimes. Only a small percent (about 10-20% of the modules in the systems) received corrective work, but those modules were continually repaired. This has an interesting implication for software management: it may be more cost effective to identify and re-write error-prone software

modules as soon as possible, because it is likely that these modules will require constant and expensive repairs over their lifetimes.

Another pattern that we have discerned relates to the amount of modification. Most software modules in the applications we studied were modified less than twice over their lifetimes. However, a small number of modules were frequently modified. We are examining data on function, size, complexity, and development characteristics to analyze why some modules receive frequent modification. Such a pattern is consistent with the systems concept of *stress localization* whereby systems have relatively independent subsystems. This allows changes to be localized and minimized by confining them to parts of the system, rather than having global effects on the system. We are creating "stress" diagrams for each system that can be used to identify the "hot spots" in the application. Such an analysis could be useful to managers who wish to identify areas of applications that are most likely to change.

We are in the process of applying ARIMA models to our data. ARIMA analysis will help us to determine whether there are systematic patterns in the series of software changes such that a mathematical model can be built to explain the past behavior of the series and forecast its future behavior.

Implications and Contributions

Our study has the potential to make important and interesting contributions relating to the management of software maintenance. Results will indicate how software evolves over time, and whether that evolution is consistent with theories of systems entropy and with mathematical models of time series. From a managerial perspective, information on software change histories can be used by software managers to more effectively plan for software maintenance. Knowledge of change history patterns can be used to improve change request management, to plan workloads for software maintainers, and to make decisions regarding software re-engineering and replacement.

References

Akaike, H., "A New Look at the Statistical Model Identification," *IEEE Transactions on Automatic Control*, AC-19, 1974, pp. 716-733.

ANSI/IEEE Standard 729, *An American National Standard IEEE Standard Glossary of Software Engineering Terminology*, 1983.

Arthur, L., *Software Evolution*, New York: John Wiley, 1988.

Boulding, K., "General Systems Theory - The Skeleton of Science," *Management Science*, April 1956, pp. 197-208.

Box, G., G. Jenkins, and G. Reinsel, *Time Series Analysis*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1994.

Checkland, P., *Systems Thinking, Systems Practice*, New York: Wiley, 1981.

Cohen, J., "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, 20:1, 1960, pp. 37-46.

Gode, D., A. Barua and T. Mukhopadhyay, "On the Economics of the Software Replacement Problem," *Proceedings of the 11th Conference on Information Systems*, December, 1990, pp. 159-170.

Kemerer, C., "Empirical Research on Software Complexity and Software Maintenance," *Annals of Software Engineering*, 1995.

Krippendorff, K., *Content Analysis: An Introduction to its Methodology*, Newbury Park, CA: Sage Publications, 1980.

Ljung, G. and G. Box, "On a Measure of Lack of Fit in Time Series Models," *Biometrika*, 65, 1978, pp. 297-303.

Martin, J. and C. McClure, *Software Maintenance: The Problem and Its Solutions*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

Schwarz, G., "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 1978, pp. 461-464.

Swanson, E.B. and C.M. Beath, *Maintaining Information Systems in Organizations*, New York: Wiley, 1989.

Weinberg, G., *An Introduction to General Systems Theory*, New York: Wiley, 1975.