A Metrics-Based Software Maintenance Effort Model

Jane Huffman Hayes
Computer Science Department
Lab for Advanced Networking
University of Kentucky
hayes@cs.uky.edu
(corresponding author)

Sandip C. Patel
Computer Science Department
University of Louisville
scapte2@uky.edu

Liming Zhao
Computer Science Department
University of Kentucky
lzhao2@uky.edu

Abstract

We derive a model for estimating adaptive software maintenance effort in person hours, the Adaptive Maintenance Effort Model (AMEffMo). A number of metrics such as lines of code changed and number of operators changed were found to be strongly correlated to maintenance effort. The regression models performed well in predicting adaptive maintenance effort as well as provide useful information for managers and maintainers.

1. Introduction

Software maintenance typically accounts for at least 50 percent of the total lifetime cost of a software system [16]. Schach et al. found that 0 – 13.8% of changes made fall under the category of adaptive maintenance [25]. It is wise to design software that is maintainable since 18.2% of project time is devoted to adaptive maintenance [25].

Software maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment" [15]. Unfortunately, developers and managers underestimate the time and effort required to change software. A lack of validated, widely accepted, and adopted tools for planning, estimating, and performing maintenance also contributes to this problem.

This paper addresses the aforementioned problem of planning and estimating when change is required. A metrics-based method is introduced that uses a model for estimating adaptive maintenance effort. A study was performed to determine metrics that correlated closely with maintenance effort as measured in time (hours). These metrics were then used to build a model for estimating adaptive maintenance effort. Two simple regression models were built. The result, the Adaptive Maintenance Effort Model (AMEffMo), appears promising from initial results.

2. Related Work

2.1 Effort Estimation Models

The Constructive Cost Model (COCOMO) [3,5,7,28] supports the estimation of cost, effort, and schedule when planning a new software development project. Albrecht introduced the notion of function points (FP) to estimate effort [1]. Mukhopadhyay [19] proposes early software cost estimation based on requirements alone. Software Life Cycle Management (SLIM) [23] is based on the Norden/Rayleigh function and is suitable for large projects. Shepperd et al. [27] argued that algorithmic cost models such as COCOMO and those based on function points suggested an approach based on using analogous projects to estimate the effort for a new project.

In addition to the traditional off-the-self models such as COCOMO, machine-learning methods have surfaced recently. In [17], Mair et al. compared machine-learning methods in building software effort prediction systems. There has also been some work toward applying fuzzy logic to building software metrics models for estimating software development effort [10]. Recent attempts have also been made to evaluate the potential of genetic programming in software effort estimation [6]. Putnam et al. [24] argue that the relationship between the metrics size, effort, and time is nonlinear. Mendes et al. [18] suggest that stepwise regression provides better predictions than linear regression.

2.2 Maintenance Effort Estimation Models

Basili et al. [2] attempted to develop a model to estimate the cost of software releases that includes large enhancements. Gefen et al. [9] present a case study to estimate total lifecycle cost of a software system. The estimation techniques for large information systems use lines of code or function points to calculate project effort in person-months [3, 4, 23].

The Albrecht Function Point model assumes that effort is primarily related to the size of a change [20]. In [20], Niessink et al state that the size of the component to be changed (as opposed to the size of changes) is crucial. In [21], they state that the size of the change and the size of the component to be changed are equally important. Niessink et al. [20, 21] also emphasize the consistency of the process in maintenance effort data collection. Henry et al. found that the number of requirements changes that occur during maintenance can be used to improve effort estimates [14].

3. Metrics Identification

We hypothesize that the maintenance effort for a software application depends on measurable metrics that can be derived from the software development process. We first identified the metrics that could affect the effort required for maintaining an application to help managers use the right metrics. Next, we worked on establishing correlation between maintenance effort and the identified metrics.

Our first set of metrics was primarily taken from the results of two previous studies: [12] and [13]. The data was taken from four sources: CS 499 and CS 616 courses (taught at the University of Kentucky), the research performed in [13] using Together® [29] and the industry research data from [12].

Table 1 displays the results of correlation analysis: the higher the value of the coefficient, the stronger the relationship between effort and the metric. The results suggest that **percentage of operators changed** and the **number of lines of codes changed edited, added or deleted** (DLOC) were the most effective for predicting adaptive maintenance effort. Note that these will have to be estimated. Table 2 describes the metrics in Table 1.

Table 1. Correlation between metrics and effort.

	Metrics	Coefficient of Determination (R^2)	Significance level from ANOVA Regression
1	%Operators Changed	0.978	0.0002
2	LOC Delta -DLOC	0.779	0.00003
3	% Mod change/add	0.192	0.117
4	Noprtr	0.152	0.444
5	CF	0.108	0.525
6	CR	0.071	0.358
7	Hdiff	0.046	0.683
8	LCOM	0.034	0.725
9	AC	0.033	0.518
10	CC	0.032	0.581
11	TCR	0.011	0.741
12	PM	0.008	0.77
13	MP	0.006	0.79
14	Classes Changed	0.006	0.8
15	MI	0.003	0.874
16	HPVol	0.0008	0.929
17	Classes Added	0.0006	0.946
18	Heff	0.0004	0.969
19	LOC	0.00001	0.991

Table 2. Metric description.

	Metrics	Description
1	%Operators Changed	Percent difference in total number of operators in the application after maintenance
2	LOC Difference (DLOC)	Lines of code edited, added or deleted during maintenance
3	% Mod change/add	% Code modules changed during maintenance
4	Noprtr	Total number of operators
5	CF	Coupling factor
6	CR	Comment ratio
7	Hdiff	Halstead's difficulty
8	LCOM	Lack of cohesion in methods
9	AC	Attribute complexity
10	CC	Cyclomatic complexity
11	TCR	True comment ratio
12	PM	Perceived maintainability
13	MP	Maintainability product
14	Classes Changed	Number of classes modified
15	MI	Welker's[30] Maintainability Index
16	HPVol	Halstead program volume
17	Classes Added	Number of classes added
18	Heff	Halstead's effort
19	LOC	Total lines of code

4. Development and Validation of AMEffMo

We built an analytical model called Adaptive Maintenance Effort Model (AMEffMo) to predict adaptive maintenance effort in terms of person-hours. We took the top two metrics from Table 1 to build AMEffMo.

"A typical estimation model is derived using regression analysis on data collected from past software projects" [22]. We used 70% of our data to build the model and the remaining 30% to validate it. Table 3 lists the data availability and the average size of the applications constituting each dataset.

4.1 Our Approach

In addition to a variation of the "leave one out" approach to evaluate the model, we performed standard checks such as examining residues, coefficient of determination, significance level, performing a bias reduction technique [8], etc.

Table 3. Number of data points per data source.

	Data Source	Total data points	Data points with DLOC information	Data points with DNoprtr info	Avg size of apps (lines of code), min, max
1	CS 499	9	8[1]	0[2]	2688.2, 101, 6201
2	CS 616	10	10	8[3]	1934, 1192, 3425
3	Previous In-house Research Data [12]	8	8	0[2]	58.6, 35, 79
4	Industry Research Data [11]	6	6	6	2725.4, 238, 18952

4.2 Multiple Regression

First, we treated the percentage of operators changed and the change in the total lines of code (DLOC) as multiple variables for the data from CS 616. The multiple regression output showed that none of the independent variables related to effort. But the simple regression performed showed that the independent variable was strongly related to effort. Next, we performed simple regression using the LOC difference (DLOC).

4.3 Simple Regression

First, we used the data from the sources numbered 3 and 4 in Table 3. Table 4 indicates that our models were built based on small- to medium-sized maintenance tasks.

Table 4. Descriptive statistics data of DLOC and Effort Spent on Changes (in person hours).

DLOC		Effort Spent on Changes	
Mean	502.19	Mean	112.75
Standard Deviation	194.87	Standard Deviation	34.34
Minimum	0	Minimum	10
Maximum	6026	Maximum	1121
Sum	16070	Sum	3608

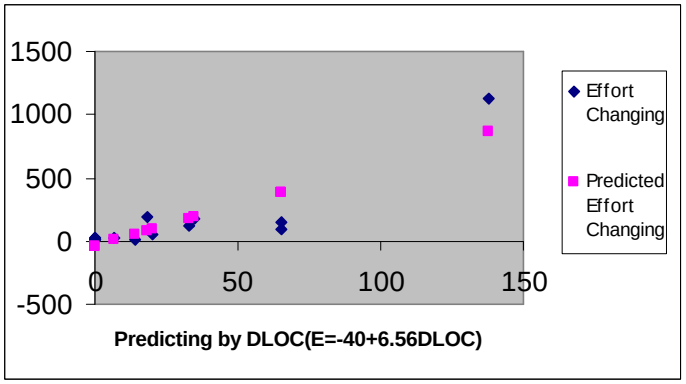
Using the least square method, we produced the following model:

$$E = -40 + 6.56\ DLOC \quad (1)$$

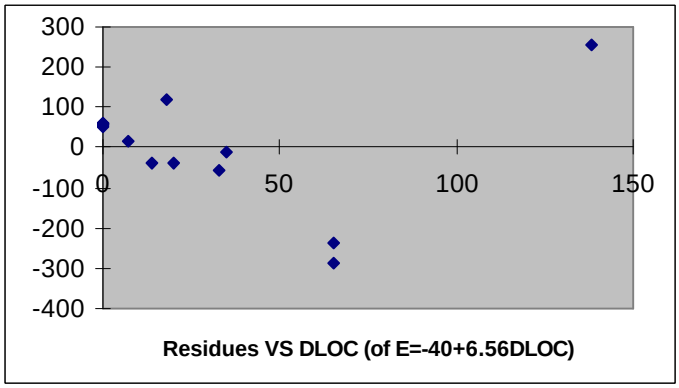
where E is maintenance effort in person-hours. This model shows a significant linear relationship between DLOC and effort spent on changes (Figure 1.a and 1.b) in which the R square value and significance are 0.78 and 0.000029 respectively.

Then, by using the course data to build the model and the rest of the data to validate, we obtained the following model:

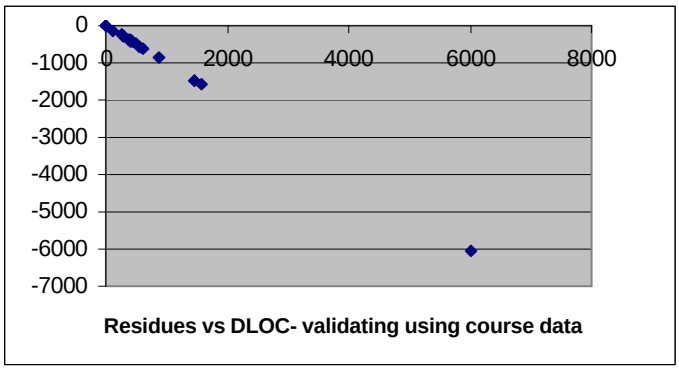
$$E = 78 + .01DLOC \quad (2)$$



(a)



(b)



(c)

Figure 1. (a) Predicting Change Effort by E = -40 + 6.56 DLOC in modeling process; (b) Residue Plot of predicting Change Effort by E = -40 + 6.56 DLOC in modeling process; (c) Residue Plot of validating E = -40 +6.56 DLOC using course data.

[1] Did not have enough information from one student team.

[2] Source code was unavailable for generating metrics.

[3] Two student teams used Visual Basic. Togethersoft does not calculate the number of operators for Visual Basic.

These two models have significantly different slopes and intercept constants, resulting in failure to validate on the validation dataset - the residue even increases as DLOC increases (Figure 1.c , Figure 2).

We built models from just one of the four data sets to investigate this further. It shows that the maintenance effort required in an industry environment is much more dependent on the DLOC than that in a university environment.

We expected a generic model to have a slope between that of the two preliminary models developed above. Considering the 70/30 ratio for building/validating data, we used the data sources numbered 1, 3, and 4 in Table 3 (a mix of industry and course data) for building the model and the data source numbered 2 (one set of course data) as the validation dataset. Before attempting to derive the final AMEffMo, we tried to identify some outliers and removed them. We received a very low statistical significance level of 1.8% for this model. Pleased with these values, we propose the following equation as our AMEffMo based on LOC difference (DLOC):

E = 63 + .1 DLOC **(3)**

When we validated AMEffMo as described above (Figure 3), we got a standard error of the estimate se of 63 with average effort ($\bar{y}$) equal to 74 (that is, standard deviation of the line of regression). We consider this error acceptable considering that AMEffMo was derived with good statistical results and was validated on a dataset that was very different from the one on which it was derived.

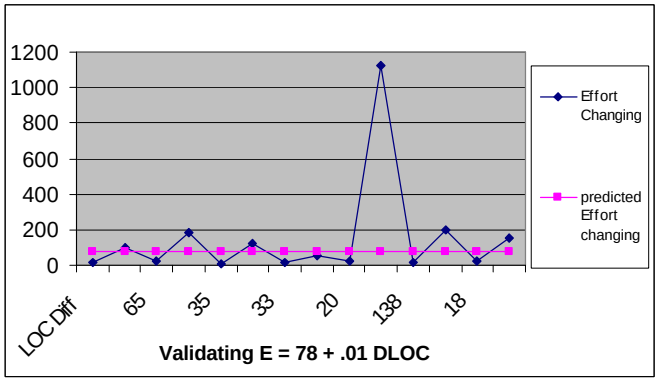


Figure 2. Line Fit Plot-Validating E = 78 +.01 DLOC using Industry and Industry-like data.

Next, we considered the other variable: number of operators changed (DNoprtr). Building on the complete data sets and excluding a few outliners, we propose the following model from all the data that can be tested by other researchers:

E = -124 + 7.5 DNoprtr **(4)**

This model has a significance level of 1.6% and coefficient of determination of .96, both well within acceptable limits.

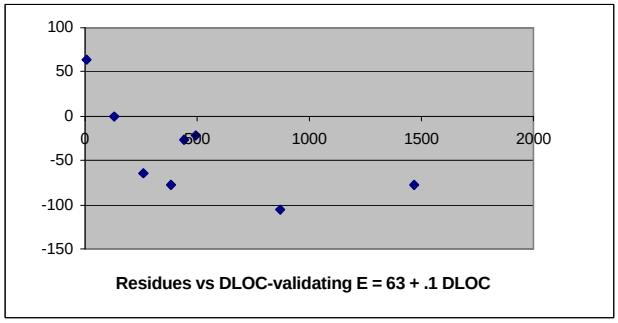


Figure 3. The residue plot- validating E = 63 + .1 DLOC on CS616 Data.

5. Conclusions And Future Work

In this paper, we have proposed a method for estimating the effort to perform adaptive maintenance based on the estimated number of lines of code to be changed and/or the number of operators to be changed. We first used our prior research to generate a list of possible metrics that could impact maintenance effort. We performed correlation analysis and ranked the above metrics. After several attempts we proposed two final models. A larger scale study with a variety of industry projects across diverse domains is required before any broad conclusions can be reached.

6. Acknowledgements

Our thanks to Perot Systems for their participation and support. We thank the students of CS 616 (January 2001), CS 499 (August 2001), Inies Raphael Michael Chemmannoor, and Senthil Karthikeyan Sundaram for their contributions. We thank Togethersoft for their donation of Together® to our research program.

7. References

[1] Albrecht, A. J. Measuring Application Development Productivity. *Proceedings SHARE/GUIDE IBM Applications Development Symposium*, Monterey, CA., Oct 14-17, 1979.

[2] Basili, V., Briand, L., Condon, S., Kim Y.-M., Melo, W. L. and Valett, J. D. Understanding and Predicting the Process of Software Maintenance Releases. *Proceedings of the 18th International Conference on Software Engineering*, Berlin, Germany, May 25-29, 1996, 464-474.

[3] Boehm, B.W. *Software Engineering Economics*. Prentice Hall, NY, 1981.

[4] Akiyama F. An example of software system debugging'', *Inf Processing*, Volume 71, 1971, 353-379.

[5] Boehm, B., Horowitz, E., Madachy, R Reifer, D., Clark, B.K., Steece, B., Brown, A.W., Chulani, S., and Abts, C. Software Cost Estimation with Cocomo II, Prentice-Hall 2000.

[6] Burgess, C.J. and Lefley, M. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, Volume 43, Number 14:863--873, December 2001.

[7] Conte, Dunsmore and Shen. *Software Engineering Metrics and Models*. The Benjamin/Cummings Publishing Company, Inc. 1996.

[8] http://www.esat.kuleuven.ac.be/sista/lssvmlab/tutorial/node59.html

[9] Gefen D., and Schneberger, S. L. The Non-Homogeneous Maintenance Periods: A Case Study of Software Modifications. *Proceedings of the International Conference on Software Maintenance*, pages, Monterey, California, November 4-8, 1996, 134-141.

[10] Gray, A., and MacDonell, S.G. Applications of fuzzy logic to software metric models for development effort estimation. *Proceedings of the 1997 Annual Meeting of the North American Fuzzy Information Processing Society - NAFIPS*. Syracuse NY, IEEE (1997) 394-399.

[11] Hayes, J. Huffman. Energizing Software Engineering Education through Real-World Projects as Experimental Studies. *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET)*, Covington, KY, February 2002.

[12] Hayes, J. Huffman, Mohamed, N., and Gao, T. The Observe-Mine-Adopt Model: An Agile Way to Enhance Software Maintainability. *Journal of Software Maintenance and Evolution: Research and Practice*, Volume 15, Issue 5, Pages 297 – 323, 1 Oct 2003.

[13] Hayes, J. Huffman. Evaluating the Reliability of Maintained Applications. University of Kentucky Computer Science Department *Technical Report TR 367-03*, January 2003.

[14] Henry, J., Blasewitz, R., and Kettinger, D. Defining and Implementing a Measurement-based Software Maintenance Process. *Software Maintenance: Research and Practic*e, Volume 8, Number 2:79–100, 1996.

[15] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990.

[16] Lientz, B.P., and Swanson, E.B. *Software Maintenance Management*, Addison-Wesley Publishing Company, 1980.

[17] Mair, C., Kadoda, G., Lefley, M., Phalp, K. Schofield, C., Sheppard, M., and Webster, S. An investigation of machine learning based prediction systems. *The Journal of Systems and Software*, Volume 53, Number 1, 2000, 23-29.

[18] Mendes, E., and Mosley, N. Comparing effort prediction models for web design and authoring using boxplots. *Proceedings of the 24th Australasian conference on Computer science*, Gold Coast, Queensland, Australia, 2001. ACM International Conference Proceeding Series, 125-133.

[19] Mukhopadhyay, T., and Kekre, S. Software Effort Models for Early Estimation of Process Control Applications. *IEEE Transactions on Software Engineering*, Volume 18, Number 10, 1992, 915-924.

[20] Niessink, F., Vliet,H. Van. Predicting Maintenance Effort with Function Points, International Conference on Software Maintenance (ICSM '97), October 01 - 03, Bari, Italy, 1997.

[21] Niessink, F., and Vliet, H. Van. Two Case Studies in Measuring Software Maintenance. *Proceedings of the International Conference on Software Maintenance*, Bethesda, Maryland, Nov. 16-20, 1988, 76-85.

[22] Pressman, R.S., *Software Engineering A Practitioner's Approach*, McGraw-Hill, 2001.

[23] Putnam, L. A general empirical solution to the macro software sizing and estimating problem. IEEE Transactions on Software Engineering., Volume 4, Number 4, , April 1978, 345-61.

[24] Putnam. L.H. and Myers, W. What We Have Learned. *Crosstalk*, April 2000.

[25] Schach, S.R., Jin, B., Wright, D. R., Heller, G.Z., and Offutt, J. Determining the Distribution of Maintenance Categories: Survey Versus Empirical Study. *Empirical Software Engineering*.. Accepted per revision by Kluwer Publications, September 2002.

[26] Schofield, C. Non-Algorithmic Effort Estimation Techniques. *Talbot CampusPoole, BH12 5BB* ESERG: TR98-01, Department of Computing, Bournemouth University England.

[27] Shepperd, M. J., Schofield, C., and Kitchenham, B. Effort Estimation Using Analogy. *Proceedings of ICSE-18*, IEEE Computer Society Press, Berlin, 1996.

[28] http://sunset.usc.edu/research/COCOMOII/index.html

[29] Togethersoft, http://www.togethersoft.com/

[30] Welker, K.D. and Oman, P.W. Software Maintainability Metrics Models in Practice, *Journal of Defense Software Engineering*, Volume 8, Number 11, November/December 1995, 19-23.

.