Implementation of a new Size Estimation Model

Aruna Chouhan[1] Mohsin Sheikh[2]
[1]MTech Scholar(CSE) [2]Asst. Professor
[1, 2]MITM,Indore

Abstract— In this paper, we present a comparison between the COCOMO size estimation and a proposed size estimation model. Our experimental results show that the proposed model is providing more accurate size. It will help in accurate effort and cost estimation. Ultimately it will result in increase in overall productivity. Size estimation is a very popular task. We also explain the fundamentals of size estimation.

I. INTRODUCTION

Software engineering cost (and schedule) models and estimation techniques are used for a number of purposes. These include:

- Budgeting
- Tradeoff and risk analysis
- Project planning and control
- Software improvement investment analysis

II. NEED OF SOFTWARE SIZE & EFFORT ESTIMATION

Small Projects are very easy to estimate and accuracy is not very important. But as the size of project increases, required accuracy is not very important. But as the size of project increases, required accuracy is very important which is very hard to estimate. A good estimate should have amount of granularity so it can be explained. Since the effort invested in a project is one of the most important and most analyzed variables. So the prediction of this value while we start the software projects, it helps to plan any forthcoming activities adequately. Estimating the effort with a large value of reliability is a problem which has not been solved yet.

III. PROPOSED ESTIMATES

COMPUTING THE SIZE OF NEW MODULES

The module is newly added to the system, thus its size is simply the KLOC added to the preexisting code. It does not consider the effect of module checking and understanding. It is denoted by KLOC(NEW) ---Eq(A)

IV. COMPUTING THE SIZE OF ADOPTED MODULES

This size is computed using the size of preexisting modules to be adapted and MA factor. Deleted statements are subtracted from number of lines. It is defined as follows:

EKLOC(ADAPTED) = AKLOC * MA ---Eq(B)

Where
FA = 0.4 * MD + 0.3 * MC + 0.3 * MI
MD – % of Modified Design
MC – % of modified code
MI - % of modified integration and testing
MA = (DAA + FA + UM)/100
Where
DAA - the degree of Assessment and Assimilation
FA – Factor of Adjustment
MA – multiplier Adjustment
UM – measure of unfamiliarity
AKLOC is the KLOC of code adapted.

V. VALUES OF DAA INCREMENT

DAA Increment	*Level of Adjustment*
0	None
2	Basic module search and documentation
4	Some module Test and Evaluation (T&E), documentation
6	Considerable module T&E, documentation
8	Extensive module T&E, documentation

VALUES OF UM

UM Increment	*Level of Unfamiliarity*
0.0	Completely familiar.
0.2	Mostly familiar
0.4	Somewhat familiar
0.6	Considerably familiar
0.8	Mostly unfamiliar
1.0	Completely unfamiliar.

VI. COMPUTING THE SIZE OF REUSED MODULES

These modules are not modified, so MD, MC, UM all are zero. It is defined as follows:-

EKLOC (REUSED) = RKLOC * MA Eq(C)

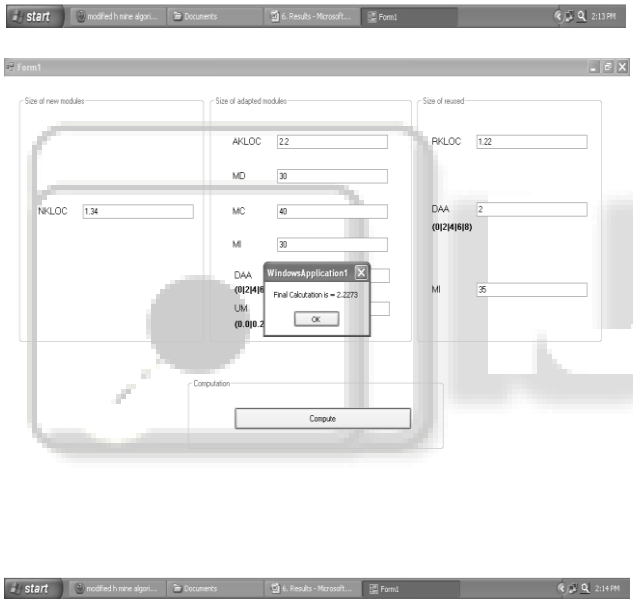
Where
RKLOC is KLOC of reused modules.
MA = (DAA + 0.3 * MI)/100
Finally,

EKLOC = KLOC(ADDED) + EKLOC(ADAPTED) + EKLOC(REUSED)

VII. IMPLEMENTATION

We have proposed a unique method for measuring the size of software reuse & maintenance. This method allows the maintainer to measure the size of both software reuse and maintenance work using the same parameters and formulas. This unique method gives actual size of complete reuse and maintenance work based on source code

delivered. It will result in improving software estimation accuracy.



VIII. COMPARISION TABLE

Modal	New kloc	Adapted kloc	Reused kloc	Estimated size
Cocomo	1.34	2.2	1.22	4.72kloc
Our proposed model	1.34	2.2	1.22	2.2273 kloc

From above table it is clear that the proposed model provides more accurate size estimates for maintenance.

IX. CONCLUSION

From our proposed model, it is clear that we consider all three aspects of maintenance: adapted code, new code, and reused code while computing the size of the software. On the other side the COCOMO II model, just takes the KLOC into consideration. Hence our proposed model gives more accurate values of effort and schedule estimates. Because these two depends up on the value of size.

REFERENCES

[1] Abran A., St-Pierre D., Maya M., Desharnais J.M. (1998), "*Full function points for embedded and real-time software*", Proceedings of the UKSMA Fall Conference, London, UK, 14.

[2] Albrecht A.J. (1979), "Measuring Application Development Productivity," Proc. *IBM Applications Development Symp*, SHARE-Guide, pp. 83-92.

[3] Albrecht A.J. and Gaffney J. E. (1983) "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, November

[4] Basili V.R., Briand L., Condon S., Kim Y.M., Melo W.L., Valett J.D. (1996), "*Understanding and predicting the process of software maintenance releases,"* Proceedings of International Conference on Software Engineering, Berlin, Germany, pp. 464–474.

[5] Boehm B.W. (1981), "*Software Engineering Economics",* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[6] Boehm B.W., Abts C., Chulani S. (2000), "*Software development cost estimation approaches: A survey*," Annals of Software Engineering 10, pp. 177-205.

[7] De Lucia A., Pompella E., and Stefanucci S. (2005), *"Assessing effort estimation models for corrective maintenance through empirical studies",* Information and Software Technology 47, pp. 3–15

[8] IFPUG (1999), *"IFPUG Counting Practices Manual - Release. 4.1,"* International Function Point Users Group, Westerville, OH

[9] IFPUG (2004), *"IFPUG Counting Practices Manual - Release. 4.2,"* International Function Point Users Group, Princeton Junction, NJ.

[10] Jeffery D.R., Ruhe M., Wieczorek I. (2000), *"A comparative study of cost modeling*

[11] *techniques using public domain multi-organizational and company-specific data",* Information and Software Technology 42 (14) 1009–1016.

[12] Jones T.C. (2008), *"Applied Software Measurement: Global Analysis of Productivity and*

[13] *Quality",* 3rd Ed., McGraw-Hill.

[14] Nguyen V., Deeds-Rubin S., Tan T., Boehm B.W. (2007), *"A SLOC Counting Standard,"* The 22nd International Annual Forum on COCOMO and Systems/Software Cost Modeling. DOI = http://csse.usc.edu/csse/TECHRPTS/2007/usc-csse-2007737/usc-csse-2007-737.pdf

[15] Park R.E. (1992), *"Software Size Measurement: A Framework for Counting Source Statements,"* CMU/SEI-92-TR-11, Sept.

[16] Symons C.R. (1988) *"Function Point Analysis: Difficulties and Improvements,"* IEEE Transactions on Software Engineering, vol. 14, no. 1, pp. 2-11

[17] UKSMA (1998) MkII Function Point Analysis Counting Practices Manual. *United Kingdom Software Metrics Association*. Version 1.3.1rs