8ème édition du congrès international pluridisciplinaire
QUALITA 2009 – Besançon (France)
Du 18 au 20 mars 2009

SOFTWARE ESTIMATION MODELS & QUALITY CRITERIA

ABRAN Alain [1], BUGLIONE Luigi [2]

[1 1] *École de Technologie Supérieure (ETS), 1100, rue Notre-Dame Ouest, Montréal, QB Canada H3C 1K3, téléphone : (514) 396-8632, Téléphone, e-mail : alain.abran@etsmtl.ca*

[2 1] *École de Technologie Supérieure (ETS) / Engineering.IT, Via R. Morandi 32, I-00148, Rome, Italie, Tel :(39) 06-83074472 , email : luigi.buglione@eng.it*

Résumé :

L'estimation des projets de développement de logiciels est un défi pour la plupart des organisations de développement de logiciels; c'est également un défi pour leurs clients qui subissent des débordements considérables des budgets, des retards importants par rapport aux échéanciers, moins de fonctionnalités que promises et avec des niveaux de qualité questionnables.

L'estimation et la qualité des logiciels sont deux préoccupations importantes pour les gestionnaires et les ingénieurs logiciels: il n'y a pas un projet de logiciel qui n'a pas besoin d'être estimé. La disponibilité de modèles et d'outils d'estimation n'est plus un problème, et certains sont même disponibles gratuitement sur le web. La vraie question est: ceux-ci sont-ils fiables? Est-ce que l'industrie du logiciel sait mieux estimer aujourd'hui qu'il y a 30 ans. Comment déterminer la qualité d'un modèle d'estimation logiciel? Quelles sont les connaissances disponibles pour évaluer l'estimation des outils mis à la disponibilité de l'industrie?

Cet article présente une vision intégrée de nombreux concepts théoriques et modalités pratiques nécessaires par des professionnels et des gestionnaires pour les aider à comprendre les fondements de l'évaluation des modèles d'estimation logiciels, ainsi que des améliorations à ceux-ci. L'approche adoptée dans le présent papier est de proposer des critères de vérification pour chacune des étapes d'un processus d'estimation des logiciels liés au processus d'estimation, la qualité de les entrées (directs/indirects) et sorties de ces modèles ainsi que la qualité des études publiées par le les constructeurs et les modèles de propagation d'erreurs à travers les différentes étapes d'un processus d'estimation.

Abstract:

Software project estimation is a challenge to most software organizations, and to their customers who endure software development projects significantly over budget, with significant delays in schedules, less functionality than promised and with unknown levels of quality.

Software estimation and software quality are two of the most prevalent issues facing software managers and software practitioners: there is not a software project which does not need to be estimated. Availability of estimation tools and techniques is not the problem anymore and some are even freely available on the web. But the real issue is: how good are they? Is the software industry better today than 30 years ago at estimating software projects? How to figure out the quality of software estimation models? What knowledge is available to assess the estimation tools available to industry?

This paper presents an integrated view of the many of the theoretical concepts and practical procedures needed by professionals and managers to help them understand the fundamentals of the evaluation of software estimation models, and of improvements to them. The approach taken in this paper is to propose verification criteria for each of the steps of a software estimation process related to the estimation process, the quality of its direct/indirect inputs, outputs for such models as well as the quality of studies published

by the models builders and of the propagation of errors through the various steps of an estimation process.

Mots clés: Modèles d'estimation, Mesures de taille, Dépôts, International Software Benchmarking Standards Group (ISBSG).

Keywords: Estimation Models, Size Measures, Repositories, Statistical Control Parameters, International Software Benchmarking Standards Group (ISBSG).

1. Introduction

Software estimation and software quality are two of the issues that software managers and software practitioners (as well as their users and organizations) face. There is not a software project that does not need to be estimated, and there is not a manager, at any level in an organization, who has not requested a fixed and accurate software effort estimate based on some fuzzy requirements, and wanted it yesterday! Unfortunately, it is common practice in the software industry to seek a single magic number to which everybody, including the estimator, has to commit at the peril of their own professional career. This is not what estimation is about. Software estimation tools have been around for the past 3 decades, and are typically multi-variable estimation models from a variety of sources:

- books and research papers
- vendors (these tools typically take a black-box approach, where neither the internal mathematical equations nor the datasets used to build them are available for independent review)
- the Web (no-fee estimation software)

The availability of estimation tools and techniques is no longer the problem. The real issue is, how good are they?

While important management decisions with significant financial impacts are taken on the basis of such software estimation models, little is done in software organizations to verify the quality of such models. By contrast, in day-to-day life (as well as in engineering and in most scientific fields), the quality of products and services is a major concern. For example, before purchasing a car, we read consumer reports or specialized magazines to compare prices, features, suggestions, and pros & cons, and consider many possible variables – both quantitative and qualitative -- which we consider relevant for making a decision. No one, of course, would rely exclusively on the advertising material produced by car vendors. Do software managers and practitioners carry out this type of due process before using a software estimation model or technique? Do they behave like wise consumers? Is the software industry better today at estimating software projects than it was 30 years ago?

Software estimation models based on statistical techniques consist of a set of mathematical operations. The use of such techniques for estimation purposes requires several types of data verification, from inputs to outputs. Figure 1 presents a generic estimation model using a process view.

Figure 1. An estimation model as a process

Researchers and estimation tool vendors are hard at work building software estimation models, but are today's proposed estimation models and approaches better suited to the task than those of 30 years ago? How can the quality of software estimation models be determined? What knowledge is available to assess the estimation tools available to industry?

This paper presents an integrated view of many of the theoretical concepts and practical procedures needed by professionals and managers to help them understand the fundamentals of evaluating software estimation models. The approach taken in this paper is to propose verification criteria for each of the steps of a software estimation process. The paper is organized as follows: section 2 proposes a procedure for verifying the quality of the direct inputs to the estimation models; section 3 sets out criteria for the derived inputs to the estimation models and criteria for the quality of the outputs of those models; section 5 presents an evaluation of the quality of estimation models by their builders, as well as independent evaluation of the models; section 6 provides some additional comments concerning the propagation of errors through the various steps of an estimation process; and section 7 presents a summary.

2. Verification of the Direct Input Parameters

Looking at estimation models as processes (Figure 1), the first step is to verify the quality of the input parameters to these models. Any estimation technique (from simple to multiple regression analyses, to neural networks, to case-based reasoning (CBR), etc.) uses the data that are fed into it, on the implicit assumption that such data are well-defined, accurate, and reliable. However, such an assumption is rarely justified in practice, and it is the responsibility of the users of the estimation tools to ensure the quality of the data fed into the estimation model as input parameters: The colloquial expression 'garbage in – garbage out' (GIGO) properly summarizes the starting point for this series of data quality verification steps. Examples of these steps are the following:

- **Verification of the data definitions**: Each input parameter should be clearly defined[1] and understood, including its scale type (nominal, ordinal, interval, ratio), and each requires distinct relevant statistical techniques [11].
- **Verification of the quality of the data collected** – see upcoming ISO 25012 [12].
- **Verification of the uncertainty about the data collected**: Is the information used for the measurement of the input complete, unambiguous, coherent, and stable? If not, what is the impact of uncertainty and how can the corresponding risk be mitigated in the estimation process.

In addition, the building (and use) of statistical techniques requires that the data parameters in the inputs meet a number of conditions, such as:

- a normal distribution in regression techniques
- identification and removal of significant outliers

The existence of a normal distribution and of outliers can be verified through statistical tests on these inputs, as well as by graphical analysis [4, 8].

[1] Refer to [25, chapter 4] for examples of well-defined definitions of data fields for software benchmarking and estimation models.

3. Verification of Derived Inputs to Estimation Models

Input verification also consists in verifying the so-called derived inputs to estimation models. In the literature, most software estimation models take as inputs the number of lines of code (LOC) and functional size in Function Points (FP), as well as a number of costs parameters.

Lines of code: Typically, this number is not derived from measurement, since the software has not yet been built. It must therefore be 'estimated', thereby introducing additional uncertainty into the estimation process. The quality of the output of the estimation process will be highly dependent on the quality of this estimated LOC input.

Functional size: A number of estimation tools indicate that they take functional size as input, but, since relatively few estimation models had, until quite recently, been built using FP directly as their main size input, a conversion step was required between FP and LOC. The term used for proposing conversion ratios between the two types of inputs is 'backfiring'. Both practitioners and researchers must be extremely careful when dealing with such conversion factors, as there are currently almost no reliable, documented studies on the comparability of the two measures across programming languages. Moreover, published conversion factors do not provide any documentary evidence of their origin or characterization from a statistical viewpoint (e.g. unknown number of data points in samples used, unknown statistical deviations from the published averages, unknown presence (or absence) of outliers within the datasets, unknown indication of potential causes of significant variation from the published averages, etc.). Without such information, those conversion ratios are merely numbers, without any intrinsic quality as valuable pieces of information for decision-making purposes. Similarly, there is little in the way of recommendations to support a professionally sound use of such numbers.

Other derived inputs: In parametric estimation models, users should gain an understanding of the strengths and weaknesses of the variables used as inputs to these models, in particular when such variables are transformed prior to being used, as in the COCOMO-like models. For example, each of the 'cost drivers' used as inputs in COCOMO I and II is:

(a) first described as a 'nominal variable', and then
(b) broken down into 5 'ordinal' categories (from 'very low' to 'extremely high').

Finally, next to each category within a cost driver, an impact factor is assigned an 'impact'; that is, a transformation of these inputs into fraction of 'days per size unit'. This means that these input cost drivers are no longer direct inputs to the estimation models, but rather 'estimation sub-models' themselves! Most of these transformations (or estimation sub-models) are neither documented nor supported by publicly available empirical data. Because the quality of such estimation sub-models is unknown, these models can but provide a weak basis for the estimation models themselves: each of these transformations is a 'black box' for which it is not possible to analyze the original input data or to determine their level of quality.

4. Analysis of the Outputs of the Estimation Models

After verifying the quality of the inputs, the next step is to analyze the outputs obtained by the estimation models. There exist multiple statistical criteria for assessing whether or not a model reflects a dataset and its ability to predict the behaviour of the dependent variable [1-3, 9-10, 13-16]. Some of the criteria most often used are:

(a) **Coefficient of determination (R^2)**: This describes the percentage of variability explained by the predictive variable in regression models. Its range is between 0 and 1: the closer it is to 1, the stronger the relationship between the independent and dependent variables.

(b) **Error of an estimate**: There are various possible views on errors: the Mean Relative Error (**MRE**), which gives an indication of the divergence between estimated and actual values as a percentage in absolute terms, and where values close to 0 are desirable; the Mean Magnitude of Relative Error (**MMRE**), which is the MRE applied to a whole dataset; the Root of the Mean Square Error (**RMS**) and its Relative RMS (**RRMS**).

(c) **Predictive quality of the model**: the prediction level of an estimation model is given by $PRED(l) = \frac{k}{n}$, where k is the number of projects in a specific sample of size n for which MRE $\leq l$. In the software engineering literature, an estimation model is generally considered good when PRED(0.25)=0.75[2].

(d) **P value statistical parameter**: This parameter expresses the statistical significance of the coefficient of the independent variables. Commonly, P values lower than 0.05 are considered significant.

The above criteria are not, however, sufficient to claim that such models are good enough from a statistical viewpoint. Estimation models built using statistical regression analysis techniques require that additional conditions be met, such as:

(a) large enough datasets: typically, at least 30 data points for each independent parameter included in the model;
(b) a normal distribution of the input parameters (dependent and independent parameters);
(c) no outlier which unduly influences the model.

When any one of the above conditions is not met, care should be exercised. For instance:

(a) with fewer than 15 to 20 projects, authors should not venture into broad generalizations about their models;
(b) with 4 to 10 data points, models should be considered as merely anecdotic and without statistical strength.

5. Evaluation of Estimation Models

5.1. Evaluation by model builders

The classical example of an evaluation by a model builder is the evaluation performed by Boehm for his COCOMO I model [5]. Table 1 shows the performance of the three versions this model, as documented by the author himself.

	MRE	**PRED(0.25)**
Basic	610%	25%
Intermediate	584%	68%
Detailed	608%	70%

Tableau 1. COCOMO I – Quality evaluation on some statistical criteria

A revised COCOMO II version was produced in the late ’90s [6,7]. The main additions include more cost drivers, as well as some updates to previous cost drivers, and a few extra features, such as the use of

[2] Of course, project managers and users would like much better levels of prediction, but these expectations are typically far beyond the state of both research and practice!

backfiring FP for determining the estimated software product size. The design of COCOMO II was not based on additional data collection and statistical analyses, but rather, as reported by Boehm and his colleagues, on opinions from various domain experts. A few subsequent studies, including one by the authors themselves on 161 projects, have attempted to verify the model's performance, but results to date have been inconclusive [19]. The revised COCOMO II model being based not on empirical data, but on expert opinion, should be considered a 'theoretical' model, and how good it is remains to be demonstrated.

5.2. Independent review of evaluations by model builders

The evaluations published by model builders are always interesting, but they are not necessarily complete. Independent evaluations are the most useful, and challenging, of course, because of the need for a dataset large enough for both replication studies using the same quality criteria and for exploring criteria not looked into by the model builders. The independent evaluation of the COCOMO I model is the easiest to perform, since its author has extensively documented his model, including the original dataset on which it was built. COCOMO I was built on the basis of a sample of 63 projects, and includes three models (Basic, Intermediate, Detailed) with 4, 18, and 72 input parameters respectively. The number of projects required for statistical meaningfulness for each of these 3 models would be 120 projects (4 independent parameters x 30 data points) for the Basic model, 540 projects (18 parameters x 30 projects) for the Intermediate model, and 2,160 projects (18 x 4 project phases x 30) for the Detailed model. Therefore, the published sample of 63 projects is barely sufficient for the Basic model and certainly not enough for the Intermediate and Detailed ones. Consequently, the reported performances achieved in terms of R^2 for the Intermediate and Detailed versions of COCOMO I are not statistically significant, since the sample size for Intermediate and Detailed models is not large enough. Therefore, COCOMO I users should not rely on the reported performance of either the Intermediate or the Detailed COCOMO I models.

6. Error Propagation in Software Measurement and Estimation

In science, the terms *uncertainty* and *error* do not formally refer to mistakes, but rather to the uncertainty that is inherent in all measurements and can never be completely eliminated. A part of an estimator's effort should be devoted to understanding such uncertainty (error analysis), so that appropriate conclusions can be drawn from variable observations. In science and engineering, numbers without accompanying error estimates are suspect and possibly useless. This also holds true in software engineering – in estimation, for every input parameter, and subsequent transformation/manipulation, the corresponding uncertainty should be recorded. Therefore, an estimator should have a good understanding of the consequences of the application of some formula (algorithm) to derive further quantities, and the same is true of estimation models. In science, this analysis is usually denoted as error propagation (or propagation of uncertainty) [17-18].

The percent error on the output of a mathematical formula with multiple parameters is not the simple sum of the percent errors on these parameters, and will depend on the function itself. Table 2 shows the uncertainty of simple functions, resulting from independent variables A, B, and C, with uncertainties ΔA, ΔB, and ΔC, and a constant c which is precisely known.

Applying the same concepts to software engineering, and in particular to parametric estimation models such as COCOMO, a general conclusion would be that the more we introduce additional cost drivers into an estimation model, the more sources of uncertainty are introduced into the estimation process. Very few estimation model builders have considered this in their search for a hope-for-all cost driver formula: when due care is not exercised, an unreasonable propagation of errors may result.

Function	Function uncertainty
$X = A \pm B$	$(\Delta X)^2 = (\Delta A)^2 + (\Delta B)^2$
$X = cA$	$\Delta X = c\,\Delta A$
$X = c(A \times B)$ or $X = c(A/B)$	$(\Delta X/X)^2 = (\Delta A/A)^2 + (\Delta B/B)^2$
$X = cA^n$	$\Delta X/X = \lvert n \rvert\,(\Delta A/A)$
$X = \ln(cA)$	$\Delta X = \Delta A/A$
$X = \exp(A)$	$\Delta X/X = \Delta A$

Tableau 2. Examples of formulae for calculating a function uncertainty [17]

7. Summary

Software project estimation is a challenge for most software organizations, as well as for their customers, who often face significant cost over-runs and substantial delays in their software development projects. Not only must customers frequently accept less functionality than promised, but also unknown levels of quality. Researchers are hard at work building software estimation models, but are today's proposed estimation models and approaches better suited to the task than those of 30 years ago?

This paper has looked into how to determine the quality of software estimation models based on the knowledge we have to assess the estimation tools available to industry. We have presented an integrated view of the many theoretical concepts and practical procedures needed by professionals and managers to help them understand the fundamentals of evaluating software estimation models. The criteria identified have been applied to the direct input parameters, to the derived input parameters, and to the outputs of the estimation models. Examples we have presented include both information provided by model builders and an analysis of the limitations of such self-assessments. The criteria have been illustrated with examples from the COCOMO models using criteria used by the model builders themselves, as well as additional criteria which must be taken into account for a more comprehensive evaluation. Further work is being carried out to develop verification procedures and to prepare detailed examples for each of the criteria we have identified here.

A proper data-driven process for building estimation models constitutes a quality-driven process for improving estimates over time and for allowing an organization to move towards higher maturity and capability levels.

Références

[1] Abran A. and Robillard P.N. (1996), *Function Point Analysis: An Empirical Study of Its Measurement Processes*, IEEE Transactions on Software Engineering 22(12): 895-909.

[2] Abran A., Gil B., and Lefebvre E. (2004), *Estimation Models Based on Functional Profiles*. International Workshop on Software Measurement, IWSM/MetriKon, Kronisburg (Germany), Shaker Verlag, pp. 195-211.

[3] Abran, A., Ndiaye, I., and Bourque, P. (2007), *Evaluation of a Black-Box Estimation Tool: A Case Study*, in a special issue entitled Advances in: Measurement for Software Process Assessment in the journal Software Process Improvement and Practice, vol. 12, no. 2, March-April 2007, pp. 199-218.

[4] Abran, I., Silva, L., and Primera, L., (2002), *Field Studies Using Functional Size Measurement in*

Building Estimation Models for Software Maintenance, in: Journal of Software Maintenance and Evolution: Research and Practice, vol. 14, 2002, pp. 31-64.

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

[6] Boehm, B., et al. (1997), *COCOMO II Model Definition Manual*, Version 1.4, University of Southern California.

[7] Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Bradford, K.C., Steece, B., Brown, A.W., Chulani, S., and Abts, C. (2000), Software Cost Estimation with COCOMO II, Prentice Hall, New Jersey.

[8] Bourque, P., Oligny, S., Abran, A., and Fournier, B. (2007), *Developing Project Duration Models*, Software Engineering, Journal of Computer Science and Technology, vol. 22, no.3, May 2007, pp. 338-347.

[9] Conte S. D., Dunsmsore D. E., and Shen V. Y. (1986) *Software Engineering Metrics and Models*, Menlo Park: The Benjamin/Cummings Publishing Company, Inc., 1986.

[10] Ferens, D. V. (1999), *The Conundrum of Software Estimation Models*, IEEE AES Systems Magazine, March, 1999, pp. 23-29.

[11] ISBSG, *Data Collection Questionnaire*, International Software Benchmarking Standards Group -- ISBSG, www.isbsg.org, accessed September 9, 2008.

[12] ISO, Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) -- Data Quality Model, International Organization for Standardization, Geneva (Switzerland), Final Committee Draft, Oct. 2007, p. 18.

[13] Kemerer, C.F., 1987, *An Empirical Validation of Software Cost Estimation Models*, Communications of the ACM 30(5): 416-429.

[14] Knaff, F. J. and Sacks (1986), J., *Software Development Effort Prediction Based on Function Points*, Proceedings of COMPSAC'86, Ill.

[15] Jensen, R. W., Putnam, L. H., and Roetzheim, W., *Software Estimating Models: Three Viewpoints*, in Crosstalk, The Journal of Defense Software Engineering, February 2006. Web: http://www.stsc.hill.af.mil/crosstalk/2006/02/0602JensenPutnamRoetzheim.html.

[16] Y. Miyazaki and K. Mori (1985), *COCOMO Evaluation and Tailoring*, 8th International Conference on Software Engineering -- ICSE, 292-299.

[17] Santillo L. (2006), Error Propagation in Software Measurement and Estimation, Proceedings of IWSM / Metrikon 2006, Potsdam (Germany), November 2-4, 2006, Shaker Verlag, ISBN 3-8322-5611-3, pp.371-382.

[18] Taylor, J. R. (1982), *An Introduction to Error Analysis*, The Study of Uncertainties in Physical Measurements, University Science Books, 1982.

[19] Bradford Clark Sunita Devnani-Chulani Barry Boehm, *Calibrating the COCOMO II Post-Architecture Model*, METRICS 1998, ICSE 2008, Kyoto, 1998