'Why COCOMO Works' Revisited or Feedback Control as a Cost Factor

J F Ramil
Dept. of Computing, Imperial College
London SW7 2BZ
tel +44 (0) 207594 8216 fax +44 (0) 20 7581 8024
ramil@doc.ic.ac.uk http://www-dse.doc.ic.ac.uk/~jcf1

Abstract

The achievement of accurate software cost estimation based only on a few factors is a long-standing goal in software engineering. Work in this area is exemplified by a number of *algorithmic* approaches that have been proposed over the years. COCOMO is one of the most frequently quoted of such approaches. Evidence emerging from observational and simulation studies suggest that feedback mechanisms play an important role in determining software process behaviour, its dynamics and performance. Thus, the presence of feedback mechanisms, and in particular *feedback control* may have a significant influence on software project cost and interval performance, but none of the current algorithmic cost estimation approaches appears, at least explicitly, to account for such influence. Why, in spite of this, do algorithmic approaches provide satisfactory estimates? Why did they work? This paper discusses some possible answers, that at the present must only be taken as hypotheses. The paper provides suggestions for further investigation of the problem.

Keywords: Cost Factors, Feedback, Feedback Control, Software Cost Estimation, Software Process Dynamics, Software Process Performance

1 Introduction

Algorithmic cost[1] estimation approaches [boe81] involve models based on historical data that reflect software project cost performance and other attributes. Total cost and interval of the project, the size of the developed software and several *cost factors* are, for example, systematically recorded. Mathematical models are fitted to these data. The models are then used to predict the cost of current and future projects. Current state of software cost estimation suggests that there is still room for its improvement. This becomes apparent when one considers that the reported predictive accuracy of approaches such as COCOMO is in the order of 30 percent [cla98]. One could attribute this perceived lack of accuracy to many reasons. For example, the historical data sets may not tend to reflect aspects of current software development. The latter are exemplified by the move towards object orientation, component-based and reused-based development involving commercial off the shelf COTS systems. Thus, the need to generate new algorithmic approaches adequate to the current software engineering practices [boe95]. In this paper a different point of view is assumed. It is asked whether such lack of accuracy may be due to a different reason: the existence of features that determine to a great extent software process performance and that have not been yet captured by current estimation approaches. Software processes, probably with the exception of the most *primitive* [pau93], can be regarded as complex multi-loop, multi-agent, multi-level *feedback* systems, as stated in the FEAST hypothesis [leh94]. The feedback role does not appear to be, at least explicitly, considered in current algorithmic cost estimation approaches. This topic appears to deserve attention. This paper discusses some of its aspects.

One can find interesting similarities between the software cost estimation problem as treated in the algorithmic approaches and what is termed *system identification* in control engineering [nor86]. The latter refers to methods for estimating the parameters of a mathematical model that will reflect system's input and output behaviour. The precautions required in control engineering during *system identification* of the system of interest, generally termed *plant,* if the latter operates under a feedback control law have been studied since the seventies [sod75, gus98]. If feedback is not disconnected while identification is performed, or accounted for in appropriate way, one may either facing difficulty to obtain the correct values of certain system parameters. When feedback control is embedded in the system being identified one will end up with a model reflecting, not only the plant, but the *aggregated* system formed by the plant and the control. That is, the estimated model will reflect *closed loop* behaviour, as the operation under a feedback control law is termed. It will not reflect, *open loop* behaviour. The closed loop system may behave very differently to the alone plant. In many cases application of feedback control may result in close

[1] In this paper wherever the term *cost* is used, *cost and interval* is implied.

loop dynamics significantly different from the open loop dynamics. System identification is a well-established discipline within control engineering. We have chosen to discuss some aspects of its theory with the hope that it may contribute to understand feedback's role in cost estimation. Following sections try to draw what appear to be the most immediate and evident conclusions.

2 Software Cost Estimation as System Identification

Figure 1 depicts a system as seen in system identification. One seeks to express the output y (a scalar or a vector) as a function of the input u (a scalar or a vector) and of the system parameters , that is, vector Θ, expressed as

$$y = G(\Theta,u). \qquad (1)$$

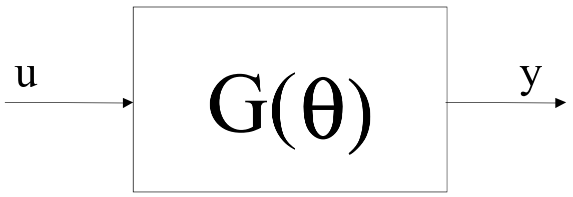

Figure 1 A simple representation of a general system with input u and output y

Θ may be determined as a function of y and u, i.e. $\Theta = \phi(y,u)$ by some appropriate system identification procedure [nor86,wel91,gus98]. Θ reflects, directly or indirectly, the properties of the system. In fact, system identification focuses on providing such procedures and in establishing their limits and the conditions of their valid application.

Consider an algorithmic cost estimation model, for example, the widely mentioned Intermediate COCOMO 81[2] cost estimation model [boe81] in which the development effort is a function f() given by

$$\text{development effort} = f(\text{mode},15 \text{ cost drivers},\text{size})$$
$$= k_1 * k_2 * ... * k_{15} * A * \text{size}^{B} \qquad (2)$$

where the development effort is in person-months, the size of the software is expressed in thousands of delivered source instructions (KDSI), the mode refers to one of the three development types (*organic, embedded, semidetached*) defined in the context of COCOMO. See [boe81] for details on the cost drivers and these modes. The multiplying factors $k_1,k_2,...,k_{15}$ are function of the cost drivers. Constants A and B are function of the development mode [boe81]. In a system identification context Eq. (2) can be recast as follows, in the form of (1):

$$y = \text{development effort}$$
$$u = \text{size}$$
$$\Theta = \{k_1,...,k_{15},A,B\}. \qquad (3)$$

One could argue, *inter alia,* whether some of the cost drivers must be considered as inputs or whether they must be considered as parameters. It cannot be provided here a detailed discussion of these and other related issues. The idea has been to provide a basis for the ensuing discussion on the impact of closed loop operation on parameter estimation.

3 Identification under Closed Loop

Software processes, as many other industrial processes, involve feedback mechanisms. Evidence to date comes from several sources, for example:

- study of metric data as, for example, observation of oscillatory behaviour in long-term growth trends of evolving software systems [bel72,leh85],
- study of phenomenology of industrial software evolution processes [leh85,94,fea00],
- simulation-based modelling of such processes by system dynamics [for61] techniques, for example, [abd91,cha99] and other simulation paradigms, as described in [kel99].

The evidence of the presence of feedback in software processes is difficult to deny. It follows that software processes *operate in closed-loop*. The data one normally gathers from such processes is closed-loop data. Algorithmic cost estimation approaches do not appear, at least explicitly, to account properly for neither the presence of feedback as a factor, nor for the use of closed-loop data.

One important aspect that is considered in system identification is whether the plant is under feedback control. In system identification one faces a problem of *identifiability* when it is not possible to estimate one or several of the system parameters. Figure 2 shows a system operating in closed-loop consisting of two major components, the plant, represented by a forward path component G', and a feedback control law H. In the rest of the paper H will be referred to as *feedback control.*

The problem is the following: if one seeks to identify the parameters of G' with knowledge only of input u and output y one may not succeed in estimating the parameters of G' alone. *Precautions are required during identification of a plant that is under feedback control* [so75, gau98]. If feedback is not temporarily disconnected, while input-output data is recorded, or accounted for in some appropriate way, one may end up with the a model reflecting, the aggregated closed loop behaviour of the system formed by the feedback control and the plant. Such system may behave different to the plant alone, the open loop plant. Application of feedback control may result in a close loop dynamics significantly different than the open loop one. For example, a given feedback control law may stabilise open-loop unstable plants. Another may destabilise an open loop stable plant.

[2] COCOMO 81 has been superseded by COCOMO II [cla98]. This paper's discussion applies also to COCOMO II and other algorithmic approaches.

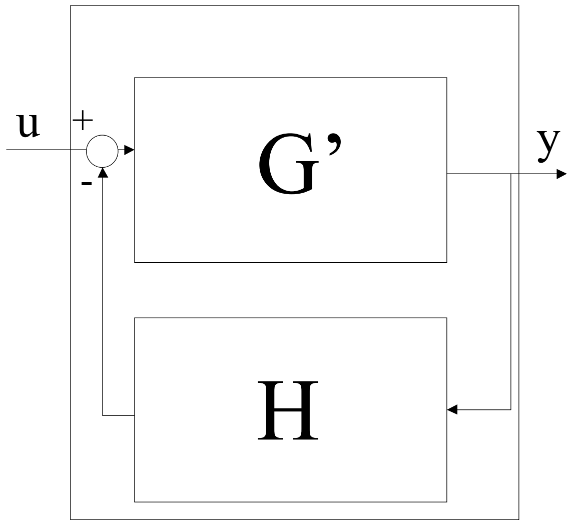

Figure 2 A closed-loop system in a black-box situation: only u and y are observable

Against to what one may initially think, the problem is not readily solved when the inside of the system is accessible to the investigator, so that, for example, she or he can measure the input to the plant of interest G'. This situation is depicted in figure 3 where u' represents such input. In this case, as well, special precautions are required to be taken if identifiability of all the relevant parameters of G' is to be achieved [wel91].

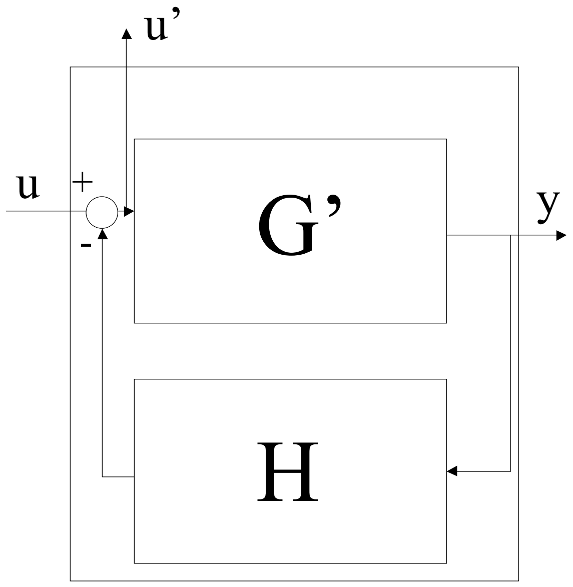

Figure 3 A closed-loop system in a white-box situation: variables internal to system are observable

What is to be concluded from the brief above remarks in the context of algorithmic software cost estimation? Consider first the fig. 3, white-box, case. This case implies that one would be able to distinguish between G' and H in the context of the software process. It would also imply that one can measure u'. Current algorithmic models, however, only appear to reflect the aggregated system, since no provision appears to be made for H. One may then ask: why do these approaches still work? A number of answers, at the present mere hypotheses, may be considered:

- *Feedback control is yet to be accounted for.* This would imply that at least part of the still *unexplained* variance in the project cost data might be due to the absence of factors accounting for the role of feedback control H.
- *Feedback as a homogeneising factor.* Feedback control, though being different in each project, may act as a regulating mechanism forcing the performance (of otherwise different processes) to converge. The latter appears to have been initially suggested in [leh96b].
- *Feedback control influences can be factored out.* This would assume that feedback control are somehow homogenous across projects and processes and hence can be safely factored out from the cost estimation procedures and algorithms.

Others might be possible. The topic appears to deserve wider empirical investigation. Initial suggestions in this regard are, for example:

- One theoretical possibility would identify and model separately G' and H from records of attributes of a process, by, for example, system identification methods. It does not appear, just to mention one of many challenges, that one can, in the context of industrial software processes, to provide conditions, such as *persistent excitation* in the input signals, or use the special methods mentioned in section 3 to achieve identifiability. Alternatives in this regard may be, however, explored.
- A second alternative would be, essentially, a white-box approach. It would try to identify G' and H by study of the process constituents and of the management control over such process, by using, for example, system dynamics [for61] methods and tools. System dynamic modelling procedures do not force a distinction between G' and H. The distinction between *process* feedback and feedback *control* must be imposed by the modeler, for example by using multi layers in the model. Such distinction may require imposing an extra discipline in the model building process.
- A third possibility would consist in finding out, by appropriate means, such as expert assessment of a process, where a given process and/or project stands in terms of its feedback control. The result could be given a set of appropriate levels or categories related to different project cost performances.

A discussion of the first possibility is left for the future. With regards to the second, experience of the authors in this regard indicates that in many cases feedback control actions may be undisciplined, irregular and mostly undocumented or unrecorded; in short, difficult to find and formalise [leh96a].

In fact, the achievement of a successful separation between the process and its control and of appropriate abstractions and representations of both, would lead, if and when achieved and applied, to a higher degree of process intellectual mastery. This seems still far from being achieved. The third suggestion relating to expert assessment is briefly discussed in the next section.

4 Feedback Control as a Cost Factor?

In the context of algorithmic software cost estimation, one may wish to understand how productivity, for example, may be related to the nature of H, the feedback control being applied to a given software process. One would like to do this without involving a detailed model of the dynamics of G' and H.

Under adequate feedback control the project would be likely to keep on track even under a significant external disturbances. Under inadequate feedback, the project may run *out of control* after the first bump in the road. Can feedback control be considered as a cost factor and hence relate process or project performance to the type or degree of feedback control being exerted? Both theoretical and empirical work might be required to understand the role of H in software processes and abstract its impact. One may, however, envisage a process assessment procedure, based on a procedure of systematic process observation and modelling that may lead to an overall assessment of the role of feedback control. This has already been done, for example, to assess the effect of process *maturity* [pau93] and later applied to investigate the relationship between such maturity and to software development effort [cla97]. Here it is not the intent to discuss the relationship(s) between feedback control and process maturity. The intention is to highlight that a systematic assessment procedure for the role of feedback control in software processes may be eventually achieved. Such procedure appears to be essential if the role of feedback control is to be understood, and even more, to be managed. Such procedure may have to involve assessment of attributes of the *global* software organisation [leh94]. This is due to the fact the software organisation is generally contained within a larger division or department and interacting with others, which are sources and sinks of process and control feedback mechanisms.

Such assessment procedure may involve the need to distinguish what are the major types of feedback control. One could use a classification of control systems in control engineering as a starting point [wel91]. Undoubtedly any classification will require to abstract appropriately the different types of feedback control in software and other industrial processes. Next, one would wish to distinguish between different feedback control adequacy levels, such as: *very low, low, medium, high, very high*. All this, will require means for assessment, based, for example, in a systematic procedure that would take into account, for example,

- organisational structure,
- structure and other attributes of the information network involving developers, users, supporters, marketeers, their managers and others,[3]
- number of hierarchy levels involved in management,
- size, structure, location and physical/cultural distance between teams involved,
- type of control procedures and mechanisms,
- its degree of automation or computer support and
- alignment between the project and the organisational structure
- alignment between the different models involved, for example, degree of model *clash* [boe98].

Work in the study of the role and impact of control in other not totally unrelated domains, as exemplified by [lee00][4], may prove to be helpful or at least provide a starting point for the study of feedback control as cost factor in software processes.

Undoubtedly feedback control also involves a cost of its own. In an ideal world one would like to quantify, not only the total project or process cost, but to be able to distinguish between the cost of the feedback control H and the cost of the plant side G'. An organisation would like to increase its degree of cost effectiveness by, for example, avoiding excessive managerial burden. In fact, this discussion does not intend to suggest that an increasing number of feedback control loops may lead to an improving process performance, however the latter is defined. On the contrary, *lightweight* feedback control mechanisms, for example those inspired in the ones in place in economic markets, may prove more advantageous than others which rely on a high degree of prescription, communication and strong dependency between the agents involved.

In summary, the author believes that it applies, in the context of feedback control, what Eilon wrote when referring to the role of organisational structures: "...It is not my intention to extol the virtues of one type of organisation as compared with another, but merely to underline the fact that structural choices may have far-reaching consequences for the enterprise." [eil79].

5 Final Remarks

This paper has argued that feedback control schemes appear to have an important role in determining process performance and, by implication, may have to be explicitly considered in algorithmic cost estimation to improve the predictive accuracy of the latter. The arguments presented suggest that feedback role must not be simply dismissed. The alternative, that is, to think of feedback as a

[3] Suggested by Dr. G. Kahen. Private communication.

[4] Reference kindly provided by Dr. G. Kahen.

homogenous mechanism that can be factored out of cost estimation, for example, is difficult to accept. Some suggestions for further study have been given. A better understanding of the role of humans as control agents (in the control theoretic sense) in software and business processes is needed. It is hoped that the feedback control abstraction may help in this regard. One has to recognised, however, that further progress in the understanding of this topic still may encounter many challenges, of theoretical, empirical and even ethical nature.

6 Acknowledgements

Grateful thanks are due to Dr. G. Kahen and to Les Barker of Matra-BAe for their comments on this paper. Thanks are also due to Profs. M.M. Lehman, D.E. Perry and W.M. Turski and to Dr. P Wernick, for having shared insights and views with the present author. Financial support from the UK EPSRC, through the FEAST/2 project, grant number GR/M44101 is gratefully acknowledged.

References

[abd91] Abdel-Hamid T and Madnick SE, Software Project Dynamics - An Integrated Approach, Prentice Hall, Englewood Cliffs, NJ, 263 p.

[bel72] Belady LA and Lehman MM, *An Introduction to Program Growth Dynamics*, in Statistical Computer Performance Evaluation, W. Freiburger (ed.), Academic Press, NY, 1972, pp. 503-511. Reprinted as chapter 6 in [leh85].

[boe81] Boehm B, *Software Engineering Economics*, Englewood Cliffs, N.J, Prentice-Hall, 1981, 767 p.

[boe95] Boehm B, Clark B, Horowitz E, Westland C, Madachy R and Selby R, *Cost models for future software life cycle processes: COCOMO 2.0*, Annals of Software Eng., Vol. 1, 1995 pp. 57-94.

[boe98] Boehm B, *Mini-Tutorial: Model-Integrated Software System Engineering (MISSE),* Proc. ICSE'98, Vol.2, April 19-25,1998, Kyoto, Japan, pp. 285 - 286

[cha99] Chatters BW, Lehman MM, Ramil JF, Wernick P, *Modelling a Software Evolution Process*, ProSim'99, Softw. Process Modelling and Simulation Workshop, Silver Falls, Oregon, 28-30 June 99, to appear as *Modelling a Long Term Software Evolution Process* in Software Process - Impr. and Practice in 2000

[cla97] Clark B, *The Effects of Software Process Maturity on Software Development Effort*, Unpublished PhD thesis, Univ. Southern California, August 1997

[cla98] Clark B, Devnani-Chulani S and Boehm B, *Calibrating the COCOMO II Post-Architecture Model*, Proc. ICSE'20, April 19-25, Kyoto, Japan, 1998, pp. 477 - 480

[eil79] Eilon S, *Management Control*, 2nd. Edition, Pergamon Press, Oxford, 1979,207 p.

[fea00] FEAST, *F*eedback, *E*volution and *S*oftware *T*echnology, FEAST/ 2 project, http://www-dse.doc.ic.ac.uk/~mml/feast

[for61] Forrester JW, *Industrial Dynamics*, MIT Press, Cambridge, Mass., 1961

[gus98] Gustafsson F and Graebe SF, *Closed-Loop Performance Monitoring in the Presence of System Changes and Disturbances*, Automatica, Vol. 24, No. 11, pp. 1311 - 1326, 1998

[kel99] Kellner MI, Madacy RJ and Raffo DM*, Software Process Simulation Modelling: Why? What? How?*, Journal of Systems and Software, Vol. 46, No. 2/3, April 1999, pp 91 -106

[lee00] Lee S and Han I, *Fuzzy Cognitive Map for the Design of EDI Controls*, Information & Management, 37 (2000), pp 37 - 50

[leh85] Lehman M.M. and Belady L.A., *Software Evolution - Processes of Software Change*, Academic Press, London, 1985, 538 p.

[leh94] Lehman M.M., *Feedback in the Software Evolution Process*, Keynote Address, CSR 11th Annual Workshop on Software Evolution: Models and Metrics. Dublin, 7-9th Sept. 1994, also in *Information and Software Technology*, sp. is. on Software Maintenance, vol. 38, no. 11, 1996, 681 - 686

[leh96a] Lehman MM, Perry DE and Turski WM, *Why is it so hard to find Feedback Control in Software Processes?* Invited Talk, Proc. of the 19th Australasian Comp. Sc. Conf., Melbourne, Australia, Jan 31 - Feb 2 1996, pp. 107-115

[leh96b] Lehman MM, *Why Cocomo Works*, Invited Talk (in absentia), COCOMO Symposium, October 1996

[nor86] Norton JP, *An Introduction to Identification*, Academic Press, London, 1986

[pau93] Paulk MC et al, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute Report CMU/SEI-93-TR-24

[ram99] *id.*, *Exploring the Relationship between Programming Effort and Software Evolution*, MPhil-PhD Transfer Report, Dept. of Computing, Imperial College, London, Sept. 1999, 22 pps. A revised version entitled *Effort Estimation from Change Records of Evolving Software*, and with Prof. Lehman as co-author has been submitted for publication in Nov. 1999

[sod75] Soderstrom T, Gustavsson I and Ljung L*, Identifiability conditions for linear systems operating in closed loop*, Int. J. Control, Vol. 21, No. 2, 1975, pp 243-255

[wel91] Wellstead PE and Zarrop MB, *Self-Tuning Systems - Control and Signal Processing*, Wiley, 1991, 579 pps