

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

Juan F. Ramil  
Dept. of Computing,  
Imperial College of Science, Technology and Medicine  
180 Queen's Gate, London SW7 2BZ  
+44 (0) 20 7594 8216 fax +44 (0) 20 7581 8024  
<ramil@doc.ic.ac.uk>

## 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 diverse studies suggests that feedback mechanisms influence software process behaviour, its dynamics and performance, including software project cost performance and project duration, 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 do they work? This paper discusses some possible answers, that at the present must only be taken as hypotheses, and provides suggestions for further investigation of the topic.

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

## 1 Introduction

*Algorithmic* cost<sup>1</sup> estimation approaches [3] 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* or *drivers* 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. Judged by the popularity and frequent reference of models such as COCOMO [3,4,9], these approaches seem to work. There still is, however, room for their improvement. This becomes apparent when one considers that the reported predictive accuracy of approaches such as COCOMO II is in

the order of 30 percent of the actual values 52 percent of the time for effort [9]. Improvement of accuracy may be pursued in several ways. For example, the historical data sets may not reflect aspects of current software development in an adequate manner. The latter is exemplified by the move towards object orientation, component-based and reused-based development, and to projects involving commercial off-the-shelf COTS components. Thus, the need to adapt the existing approaches or to develop new ones that address the current software engineering practices [4]. Improved accuracy may also be achieved by investigating new factors. Such factors may significantly influence software process performance and may have not been captured by current approaches. *Feedback control*<sup>2</sup> could be one such factor.

*E-type* [16] software evolution processes are complex multi-loop, multi-agent, multi-level *feedback* systems [17]. *E-type* systems are those regularly used to solve a problem or address an application in a real world domain [16]. The statement is supported by observations collected since the seventies [2,16,17], most recently by the FEAST/1 and /2 projects [11] and by others [1,20], who have studied feedback dynamics at the project level.

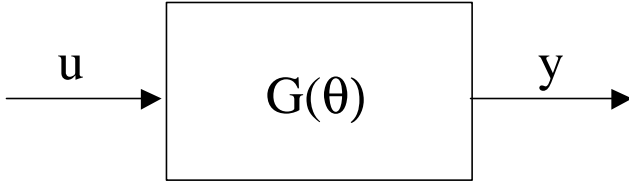
The feedback role does not appear to be explicitly considered in current algorithmic cost estimation approaches. The topic deserves attention and this paper discusses some of its aspects. One can find similarities between algorithmic software cost estimation problem and what is termed *system identification* [21]. Loosely defined, the latter provides theory and procedures to achieve mathematical models of system's input-output dynamic behaviour directly from input and output data. By referring to some of its concepts this paper derives implications of the presence of feedback for algorithmic cost estimation.

<sup>1</sup> In this paper wherever the term *cost* is used, *cost and interval* is implied. Cost and effort are considered synonyms.

<sup>2</sup> Feedback in the sense of "return of part of the output of a system to its source" Oxford Advanced Learners Dictionary, Oxford Univ. Press 1989

## 2 Software Cost Estimation seen as a System Identification Problem

Figure 1 depicts a system as generally 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  $\Theta$ , expressed as

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


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

$\Theta$  may be determined as a function of  $y$  and  $u$ , i.e.  $\Theta = \phi(y, u)$  by some appropriate procedure [13,21,25]. In fact, system identification focuses on providing such procedures and in establishing their limits and the conditions for their application.

Now consider an algorithmic cost estimation model, for example, the widely mentioned Intermediate COCOMO 81<sup>3</sup> cost estimation model [3] in which the development effort is a function  $f()$  given by

$$\begin{aligned} \text{development effort} &= f(\text{mode}, 15 \text{ cost drivers}, \text{size}) \\ &= k_1 * k_2 * \dots * k_{15} * A * \text{size}^B \end{aligned} \quad (2)$$

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. The multiplying factors  $k_1, k_2, \dots, k_{15}$  are function of the cost drivers. Constants  $A$  and  $B$  are function of the development mode [3]. See [3] for details regarding COCOMO's drivers and modes. Eq. 2 can be recast as follows, in the form of Eq. 1:

$$\begin{aligned} y &= \text{development effort} \\ u &= \text{size} \\ \Theta &= \{k_1, \dots, k_{15}, A, B\}. \end{aligned} \quad (3)$$

One could argue that some cost drivers are project features and hence part of  $u$  or whether they must all be considered as process related and hence part of  $\Theta$ . While this and other considerations may alter the final form of Eq. 3, it will not alter the view that algorithmic cost estimation, as generally pursued, is an attempt to *identify* (in the system identification sense) an appropriate model for the system (the software process). Such system operates under feedback control, with feedback not being properly accounted for. Even so, at first sight rather surprisingly, to a good degree such approaches seem to work.

## 3 Why do Algorithmic Cost Estimation Approaches such as COCOMO 'Work'?

*E*-type software evolution [16] processes involve feedback mechanisms at many levels [17] with evidence emerging from several sources, for example:

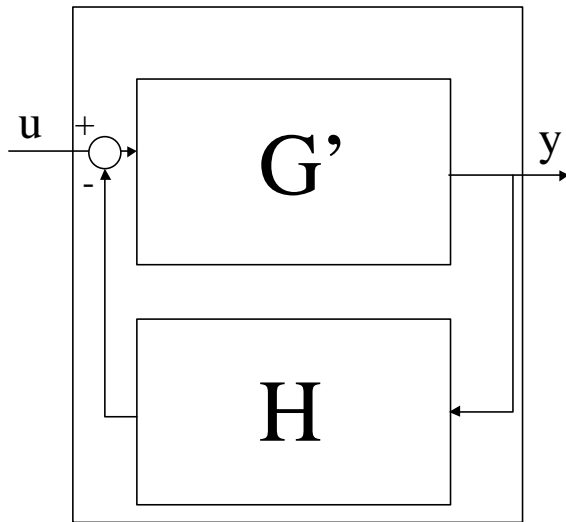
- study of metric data as, for example, observation of oscillatory behaviour in long-term growth trends of evolving software systems [2,16]
- study of phenomenology of industrial software evolution processes [11,16,17]
- simulation modelling of such processes using *system dynamics* (SD) [12], as in [7,11] and using other modelling techniques [14].

Such evidence suggests it as a fruitful topic of study, though it involves some challenges [18]. These influences may be particularly challenging to be subject of mathematical modelling in the less *mature* [22] processes [11], in small size software development (however small is defined) and/or where software development is a secondary or non-seriously pursued activity. It follows that, in general, software evolution processes *operate in closed-loop*, as the operation under a feedback control is termed. Feedback mechanisms may play also an important role not only in long-term evolution [7] but also in shorter-lived software projects. Some SD studies have addressed cost estimation issues at the project level, for example in [1,20]. However, SD is not widely applied in industry. On the other hand, algorithmic approaches, which appear to be much more popular as judged by the number of references, do not account, at least explicitly, for either the presence of feedback, or for the use of closed-loop data<sup>4</sup>.

Consider some of the consequences of feedback control seen from the point of view of system identification. The issues arising during *system identification* of a system of interest, generally termed *plant*, if the latter operates under a feedback control law, have been studied at least since the seventies [13,24], if not earlier. It is well known, for example, that if feedback is not disconnected while system identification is performed, or accounted for in appropriate way, one may face difficulties to obtain a model that is an appropriate representation of the system studied. When feedback cannot be disconnected from the plant, such as in some unstable processes, or when feedback is somehow *embedded* in the plant being identified, one will obtain a model reflecting, not only the plant, but the *aggregated* system formed by the plant and the feedback control. That is, the estimated model will reflect *closed-loop* behaviour. The closed-loop system may behave very differently to the open loop system. These observations become more evident when one considers figs. 2 and 3.

<sup>3</sup> COCOMO 81 has been superseded by COCOMO II [9]. This paper's discussion applies in general to all current algorithmic approaches.

<sup>4</sup> Implications of use of closed-loop data in statistical modelling are examined in [6].

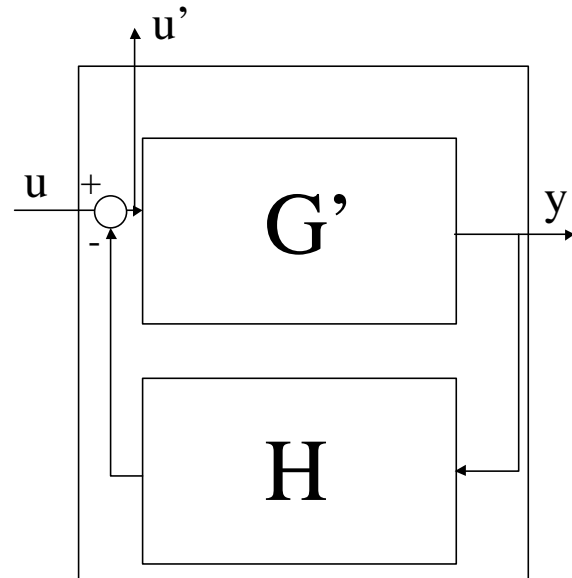


**Figure 2 A closed-loop system in a black-box situation: only  $u$  and  $y$  are observable**

Fig. 2 shows a system consisting of two components, the productive part, represented by  $G'$ , and a feedback control  $H$ , representing, for example, management controls. As mentioned above and due to the presence of  $H$ , when one seeks to obtain a dynamic model and the parameters of  $G'$  with knowledge only of  $u$  and  $y$  it may not be possible.

Suppose now that the inside of the system is accessible to the investigator, so that, for example, she or he can measure  $u'$ , the immediate input to  $G'$ . This situation is depicted in figure 3. Still in this case precautions may have to be taken if proper estimation of an appropriate dynamic model of  $G'$  is to be achieved [25].

Instead of considering such precautions and analysing whether they would be the basis to *new* cost estimation approaches, one may ask what is to be concluded from the above observations in the context of *current* algorithmic software cost estimation. Consider first the case of fig. 3 in which one is able to distinguish between  $G'$  and  $H$ . This case implies that one is able to measure not only  $u$ , the input to the software process, but also  $u'$ , the input to the forward paths of such process. Current algorithmic models, however, only appear to be based on measures of the *aggregated* system, since no provision appears to be made for either  $H$  or  $u'$ . So, at least in principle, fig. 3 can be regarded as not representative of current practice. On the other hand fig. 2 appears to be a closer reflection of such practice. Given the limitations imposed by the presence of feedback to the situation in fig. 2, one may conclude that current estimation models are in this sense *biased*. They reflect the aggregated behaviour of  $G'$  and  $H$ .  $H$  is not reflected, at least explicitly, in the currently considered cost factors [3,49]. Having said that, one then may ask: why do these approaches still work? In particular, why COCOMO [3] works? A number of answers, at the present mere hypotheses, may be considered:



**Figure 3 A closed-loop system in a white-box situation: variables internal to system such as  $u'$  are observable**

- *Feedback as a performance equalising factor.* Feedback control may act as a regulating mechanism forcing the performance (of otherwise different processes) to converge. The latter appears to have been initially suggested in [19]. Moreover, as one of the anonymous referees has appropriately commented, COCOMO estimates may work at least in part, due to the fact that often managers act based on comparison of estimates with in-progress actuals. By doing so, as a project progresses, managers act as feedback controls that turn the estimates into "self-fulfilling prophecies" [3].
- *Feedback control influences can be factored out.* One may consider that feedback control role is to a significant degree homogenous across projects and processes and hence can be safely factored out from the algorithmic cost estimation approaches.
- *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 and impact of the feedback control.

Others might be possible. The list suggests that the topic deserves wider investigation. Suggestions in this regard are:

- *Black box modelling.* This would attempt to model  $G'$  and  $H$  from records of attributes of a process, by, for example, system identification methods or adaptations of these. It is not evident how to apply such methods to industrial software processes. For example, it is not clear how one could provide the experimental conditions, such as *persistent excitation* in the process inputs [25], to guarantee achievement of appropriate dynamic models.

- *White box modelling.* A second alternative would be, essentially, a white-box approach. It would try to model the dynamics of G' and H by study of the process constituents and of the management control [10] over such process, by using, for example, SD modelling and tools. As already mentioned SD approaches have been applied to software cost estimation, as in [1,20]. However, SD modelling procedures do not force an immediate distinction between G' and H. The distinction between *process* feedback and feedback *control* must be imposed by the model builder, for example by using multi layers in the model. Such distinction may require to impose an extra discipline in the model building process. Such discipline does not appear to be enforced by any of the existing SD modelling tools.
- *Feedback control as a cost factor.* A third possibility would consist in investigating, by appropriate means, such as expert assessment of a process, where a given process and/or project stands in terms of feedback control. The result could be given a set of appropriate levels or categories related to different project cost performances. This, in turn, may be considered as cost drivers, in the COCOMO sense [3].

A detailed discussion of the first possibility is left for the future. With regards to the second, experience of the author as part of the FEAST projects [11] indicates that is worth pursuing though time and effort consuming. For wider impact one would like to achieve generic SD models. Whether these are achievable and how to achieve them is still a matter of investigation. Moreover in some processes, in particular in the less mature [22] feedback control may be undisciplined and/or undocumented; in short, difficult to find, formalise and model [18]. Here it is not the intent to discuss the relationship(s) between feedback control and process maturity, which in itself could be an interesting topic of investigation. If successful, the second possibility would lead to a valid decomposition or separation between the process G' and its control H and of appropriate abstractions and representations of both. This, in turn, would lead, if and when achieved and applied, to a higher degree of software process mastery. This seems still far from being achieved though progress has been made [11]. The third suggestion 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 software process performance, for example, may relate to the nature of H, the feedback control being applied to a given software process. For the sake of simplicity one would like to do this without a detailed model of the dynamics of G' and H.

One can argue that under appropriate feedback control the project would be likely to keep on track even under significant external disturbances and changes. 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 in operation in a given process? 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 an 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* [22] and later applied to investigate the relationship between maturity and software development effort [8].

An assessment procedure for feedback may have to involve assessment of attributes of the *global* software process<sup>5</sup> or organisation [17]. This is due to the fact the software process is generally embedded in a larger organisation and interacts with other agents and processes, from which dynamic influences stem and which sometimes are involved in across-organisational feedback control mechanisms.

Such assessment procedure may require the distinction between a set of major types of feedback control. One could use a classification of control systems in control engineering as a starting point [25]. Undoubtedly any classification will require to abstract appropriately the different types of feedback control in software and other industrial processes and their effects. 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,

- software project and organisation structure [10]
- number of hierarchy levels involved
- size, structure, location and physical/cultural distance between teams involved
- control procedures, policies and other mechanisms
- its degree of automation or computer support
- alignment between the agents involved, for example, degree of *model clash* [6] and
- structure and other attributes of the global process and the information network involving developers, users, supporters, marketeers, their managers and others.<sup>6</sup>

<sup>5</sup> The *global* software process is seen as the process that encompasses the activity not only of developers and maintainers but also of others involved, such as support personnel, marketeers, users and their managers [16].

<sup>6</sup> G. Kahen. Private communication.

Work in the study of the role and impact of control in other domains, not totally unrelated to software, as in [15]<sup>7</sup>, may be helpful, at least as 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 also to be able to distinguish between the cost of the feedback control H and the cost of running the productive process 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 *a priori* that an increasing number of feedback loops or mechanisms may lead to improved cost and schedule performance. For example, *lightweight* feedback may be more advantageous than other schemes that rely on high degrees of prescription, communication and/or strong dependencies between the agents involved. In this connection, it seems to apply to *feedback control* in the software process, what Eilon affirmed when referring to *organisational structures* in general. This author wrote "...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 [10]."

## 5 Final Remarks

This paper has argued that feedback control appear to have an important role in determining process performance and may have to be explicitly considered in algorithmic cost estimation if the accuracy of the latter is to be significantly improved. The arguments presented suggest that the feedback role must not be simply dismissed. Of course there has been progress in SD modelling of software processes with application to cost estimation [1,20]. These models reflect feedback interactions. However, how to adapt or develop *algorithmic* estimation approaches so that they reflect feedback-related factors appears to be a question worth pursuing, in particular, when seeking approaches to estimate cost for software evolution processes [23]. This paper has given some suggestions for further study. It is hoped that if further pursued it may lead to improved cost estimation approaches. Last, but not least, in pursuing this topic one would need to address the role of individual humans, groups and teams, human agencies and organisations as control agents (in the control theoretic sense) in software and business processes. Understanding this topic will necessarily require more than just mathematical models and modelling techniques and would be studied best from an interdisciplinary perspective [17].

---

<sup>7</sup> Reference kindly provided by G. Kahen.

## Acknowledgements

Grateful thanks are due to M Lehman, G. Kahen, L Barker and to the anonymous referees for their comments. Financial support from the UK EPSRC, through the FEAST/2 project, grant n. GR/M44101 is gratefully acknowledged.

## References

- [1] Abdel-Hamid T and Madnick SE, *Software Project Dynamics - An Integrated Approach*, Prentice Hall, Englewood Cliffs, NJ, 263 p.
- [2] 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 [15].
- [3] Boehm B, *Software Engineering Economics*, Englewood Cliffs, N.J, Prentice-Hall, 1981, 767 p.
- [4] 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.*, v. 1, 1995 pp. 57-94.
- [5] Boehm B, Mini-Tutorial: Model-Integrated Software System Engineering (MISSE), *Proc. ICSE 98*, v.2, April 19-25,1998, Kyoto, Japan, pp. 285 - 286
- [6] Box GEP, Hunter WG and Hunter JS, *Statistics for Experimenters - An Introduction to Design - Data Analysis and Model Building*, Wiley, New York, 1978
- [7] Chatters BW, Lehman MM, Ramil JF and Wernick P, Modelling a Software Evolution Process, ProSim'99, Silver Falls, Oregon, 28-30 June 99, to appear as Modelling a Long Term Software Evolution Process in *Softw. Proc. - Impr. and Practice* in 2000
- [8] Clark B, *The Effects of Software Process Maturity on Software Development Effort*, Unpublished PhD thesis, Univ. Southern California, August 1997
- [9] 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
- [10] Eilon S, *Management Control*, 2nd. Edition, Pergamon Press, Oxford, 1979,207 p.
- [11] FEAST web site <<http://www-dse.doc.ic.ac.uk/~mml/feast>>
- [12] Forrester JW, *Industrial Dynamics*, MIT Press, Cambridge, Mass., 1961
- [13] Gustafsson F and Graebe SF, Closed-Loop Performance Monitoring in the Presence of System Changes and Disturbances, *Automatica*, v. 24, n. 11, pp. 1311 - 1326, 1998
- [14] Kellner MI, Madachy RJ and Raffo DM, Software Process Simulation Modelling: Why? What? How?, *Journal of Systems and Software*, v. 46, n. 2/3, April 1999, pp 91 -106
- [15] Lee S and Han I, Fuzzy Cognitive Map for the Design of EDI Controls, *Information & Management*, 37 (2000), pp 37 - 50
- [16] Lehman M.M. and Belady L.A., *Software Evolution - Processes of Software Change*, Acad. Press, London, 1985

- [17] 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, v. 38, n. 11, 1996, 681 - 686
- [18] 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
- [19] Lehman MM, Why Cocomo Works, Invited Talk (in absentia), *COCOMO Symposium*, October 1996
- [20] Madachy R, System Dynamics Modeling of an Inspection-based Process, *Proc. ICSE 18*, Berlin, Germany, March 25 - 29, 1996, pp 376 - 386
- [21] Norton JP, *An Introduction to Identification*, Academic Press, London, 1986
- [22] Paulk MC *et al*, *Capability Maturity Model for Software, Version 1.1*, Softw. Eng. Institute Report CMU/SEI-93-TR-24
- [23] Ramil JF, *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 as: Ramil JF and Lehman MM, Metrics of Software Evolution as Effort Predictors - A Case Study, *Proc. ICSM 2000*, October 11-14, 2000, San Jose, CA
- [24] Soderstrom T, Gustavsson I and Ljung L, Identifiability conditions for linear systems operating in closed loop, *Int. J. Control*, v. 21, n. 2, 1975, pp 243-255
- [25] Wellstead PE and Zarrop MB, *Self-Tuning Systems - Control and Signal Processing*, Wiley, 1991, 579 pps

