

Cost Estimation and Evolvability Monitoring for Software Evolution Processes

WESS 2000 Workshop on Empirical Studies of Software Maintenance, Oct. 14, 2000, San Jose, CA

J F Ramil M M Lehman

Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
tel + 44 (0) 20 7594 8214; fax + 44 (0) 20 7594 8215
{ramil,mml}@doc.ic.ac.uk
<http://www-dse.doc.ic.ac.uk/~mml/feast>

1 INTRODUCTION

*Software*¹ *evolution* is a continuing process that encompasses not only *ab initio* development but *all* activities, enhancement, adaptation or fixing, that occurs after the first operational release. However, in the present paper we consider only the post first release portion of the evolutionary process. This includes projects for enhancement and adaptation of a current software system and subsumes its maintenance, however the latter is defined [e.g., ieee93]. Parenthetically we note that, in general and for several reasons, the term evolution is to be preferred to maintenance in the context of software for post first release activity. In hardware artefacts (e.g., cars), for example, the principal aim of *maintenance* is to compensate for wear, tear and material deterioration so as to restore the condition or performance to an earlier or even original state. Software, on the other hand, does not generally deteriorate through use but must be *evolved* to maintain it satisfactory with respect to the changing operational domain or purpose.

Cost estimation in the context of *ab initio* software development has received considerably attention over the last decades, but not so in the context of evolution. Thus, techniques such as algorithmic estimation [boe81], estimation by analogy [she96] and those based on expert opinion [boe81] have been applied mainly to the *ab initio* case. The application of algorithmic techniques is exemplified by the COCOMO model in its various versions and extensions [boe81,coc00]. In that instance, however, extensions for estimating the cost of software maintenance have been proposed [e.g., boe81,gra87]. Analysis of continuing software evolution reveals, however, a number of unique features that must be considered in the search for estimation approaches that remain useful as a system evolves. Such features must also be taken into account in considering *evolvability*. This is briefly discussed in the final part of this paper.

What are the characteristic features of the software evolution situation? A number are implied by the set of behavioural statements encapsulated in the laws of software evolution as described by their names, *continuing change*, *increasing complexity*, *self-regulation*, *conservation of organisational stability*, *conservation of familiarity*, *continuing growth*, *declining quality* and *feedback system*. These statements, their origin and the evidence in their support have been discussed elsewhere [leh74,85,feast]. Space limitations do not permit their discussion here. In the context of cost estimation, the laws suggest consideration of the following:

- evolution implies that the software system must continually be changed. Hence, in addition to its size, software change rate and change activity must be measured and reflected in cost models

¹ The reference here and throughout the paper is to *E*-type software, that is, software operating in and actively used to solve a problem or implement an application in a real-world domain. Need for continuing evolution is an inherent property of such software [leh85].

- if not properly controlled, increasing complexity and other aging effects [par94] may play an important role and become a significant cost factor. This is particularly so, where a system has evolved over a long period with change imposed upon change imposed upon change
- feedback forces [leh94], management policies [star98], organisational [stau98], project and team structures all play an important role in determining costs though their relative contributions may vary from system to system
- the evolution situation is dynamic, varying with time. This may limit the use, in the evolution situation, of static models such as those proposed for *ab initio* development [boe81], that is, models that, whilst possibly varying from system to system, do not reflect time-varying behaviour in their application to any one system.

Evolution of a particular system may span several years, even decades. This offers the possibility of using historical data from the system itself, as a basis for the calibration of models. In this respect, one may distinguish two types of cost models: *generic* and *specific*. General models, such as COCOMO, are intended to cover many different possibilities in terms of software project size and of other cost factors. In principle they are not restricted to a given project class, domain or development environment. However, one may wish to add the prefix 'quasi' to the word 'generic' to indicate that as software engineering practice improves these models may become outdated. Such models need periodic adaptation or replacement, at the very least re-calibration to the process they must reflect, to remain valid [boe00]. Current trend towards adoption of COTS-intensive and components-intensive processes [leh98] represents an example of a change in practice that requires new cost models [boe00,coc00].

The second category identified, *specific* models, are primarily intended to estimate the cost of a particular system (or family of systems) over its future evolution. Such models may be derived from and calibrated to historical data representing one or a small set of closely related systems and processes, for example those evolved by the same software organisation. Though the actual models may be limited in their validity to the domain within which they have been calibrated, the *approaches* to obtain such specific models have wider applicability. As briefly discussed in the next section, the present authors are currently interested in such approaches.

2 COST ESTIMATION IN THE EVOLUTION CONTEXT

In our investigation of approaches to cost estimation we have been seeking those that take into account the unique features of the software evolution situation. In searching for a suitable solution we have focused our attention on model-based cost estimation, in which estimation is pursued by means of mathematical models (e.g. regression-based, simulation-based and their combinations). In FEAST we have been distinguishing to classes of complementary [ram00c] models: *black-box* and *white-box*. The first reflect structure in the data. Thus, they model externally observed behaviour. The second category, white-box, reflects activities, influences and forces within the portion of the real world being modelled. Both classes can be used in cost estimation. Examples of such data would be a database of past evolution projects, data on a sequence of releases and time series data. Records may include size metrics reflecting system growth, system change rate, effort applied and variables that are potentially significant cost factors. All these are to be recorded for the project, the release or the time interval of interest. Black-box models are primarily derived from empirical data. White-box model structure are to be derived from expert experience, observations and insight. Their parameters are to be estimated, i.e., the model calibrated, by using empirical data at various levels of aggregation [gra80].

Issues such as the following are being examined:

- identification of the appropriate level of abstraction or aggregation of the model [leg80,zur67]
- selection of indicators or measures of growth, change rate, other evolution activity measures and effort applied
- identification of potentially relevant cost drivers in the evolution situation
- exploration of the impact of choice of *level of observation* and measurement (e.g., size measured by sub-system, module, procedure or line of code)
- exploration of the impact of choice of *level of granularity* of measurement unit (e.g., size measured in sub-system, module, procedure or line of code *counts*)
- exploration of the role and relative value of time (e.g., months, quarters, years) and pseudo-time (e.g., release sequence number) units
- identification of adequate model partitioning. For example, separate models for classes of activity, such as individual models for fixes, enhancements, new development; individual models reflecting parts of the systems that may display significantly different evolution rates (e.g., human-interface subsystem vs. other sub-systems) or models reflecting homogenous periods (defined in some sense [e.g., ram00e]) or over individual life cycle stages [ben00,raj00]
- identification of adequate mathematical model structures (open-loop vs closed-loop, linear vs. non-linear; static vs. dynamic; discrete time vs. continuous time; deterministic vs. stochastic; quantitative vs. qualitative, etc.)

In planning this investigation we have identified the need for the following steps:

1. Locate sources of empirical data:
Three industrial data sets that appear to be rich enough to conduct the cost estimation study have been obtained to date. These contain thousands of *change-log records*² that reflect evolution activity over several years, even decades, of evolution.
2. Perform empirical data extraction:
We have begun this process by using *Perl* scripts [wal96]. These have permitted the extraction of a set of metrics representative of size, change and effort applied.
3. Define models:
In principle, the number of combinations of potential cost drivers, model structures, levels of observation and granularity, sampling units, etc. is large, leading to a huge space of possible models. We have identified the need of a systematic exploration of the space of possible models. In this regard, top-down modelling may be relevant [e.g., zur67]. On the black-box modelling side of our work we have been looking at econometric modelling techniques [guj95], with particular interest in procedures to choose amongst competing models. On the white-box modelling side of the work we have been applying *system dynamics* [coy96,gra80,leg80] (SD) modelling techniques.
4. Calibrate model parameters and assess predictive power:
Results of our black-box work on cost estimation work are reported in [ram00a,b,d,e]. We plan to contrast our findings with those reported by others [e.g., bri92,jor95].

² The existence of unplanned change-log records provided a useful data source even despite that for one of the systems no planned historical data source was available. It was also found that by enforcing simple rules in the format of the change-log recording, data extraction for cost modelling would be greatly facilitated.

3 EVOLVABILITY MONITORING

In this section one of the WESS 2000 workshop topics, '*Can we measure maintainability? If so, how? If not, why not?*', is briefly addressed. Though the cost modelling work is still in its early stages, we are already able to visualise potential uses of the models beyond cost estimation, and indeed, to monitor *evolvability*³. Such a property may be considered either a scalar or a vector or a matrix, reflecting several dimensions. In any case it will be an indicator of the degree of simplicity in performing further evolution of the system. The focus here is on the economic, resource oriented, dimension. Other facets (e.g., code complexity, architectural integrity of the system, experience of developers, degree of orthogonality of foreseen changes with respect to current system architecture, state of documentation and its appropriateness) may have to be considered. From the economic (aggregated) point of view, however, an increase (or decrease) in evolution productivity will indicate that *evolvability* has also increased (or decreased, respectively), everything else constant. Therefore, cost models may be means to operationalise (define, measure, monitor) the *evolvability* concept. This may be achieved either from a black-box or a white-box perspective.

As an example of the black-box perspective, consider one of our initial effort models [ram00e]:

$$DEffort(t) = A \cdot ModulesHandled(t) + B \quad \text{Eq.1}$$

where $DEffort(t)$ is the effort in person-months applied during a one-month interval (from month t to $t+1$) and $ModulesHandled(t)$ is number of modules which were either added to the system or modified, or both (if both, the module is counted only once) during the interval. A and B , the model parameters, are to be derived from historical data, by, for example, least squares regression [guj95]. By detecting changes to A and B as a system evolves one may infer changes in *evolvability*. Note that A and B imply a high level of aggregation definition of *evolvability*. The lower the values of A and B are, the higher the productivity (and the *evolvability*) of the system would be, everything else constant. By detecting changes in A and B one may, in fact, detect changes in *evolvability*. The selection of a cost model different than the one in Eq. 1, as the other five models proposed in [ram00e] will yield a different operationalisation of the *evolvability* concept.

Black box models are appropriate to reason about the externally observed process behaviour, though they display the limitation, *inter alia*, of not reflecting the impact of changes in policies. One such policy that may impact *evolvability* is the fraction of the applied effort dedicated to complexity control activities, the so-called *anti-regressive* work [leh74]. Such type of work does not impact system functionality as perceived by the user. *Progressive* work increases the functionality or power of the system. In the SD model⁴ of Figure 1, *evolvability* is defined as a function of the difference between the cumulative progressive work and the cumulative anti-regressive work. The larger the latter, the lower *evolvability* would be. Again, the model operationalises the concept. In Figure 1 *evolvability* is defined within the concepts and the framework of the model. White box models such as the SD example presented in Figure 1 permit the exploration of the potential effect of various policies. In this regard they are superior to the black-box counterparts. When measuring and monitoring is the principal goal, white-box modelling may bring disadvantages. If *evolvability* is a direct model input, then one may directly measure it over a series of projects, releases or time intervals and assess whether it has changed or not. If *evolvability* is an internal model parameter or a model output, then the assessment of changes in *evolvability* may have to be performed in a more sophisticated way.

³ In the first section we have justified the preference of the term evolution over maintenance for software. For similar reasons we suggest the use of the term *evolvability* instead of *maintainability*.

⁴ See, for example, [coy96] for a general description of the SD modelling approach.

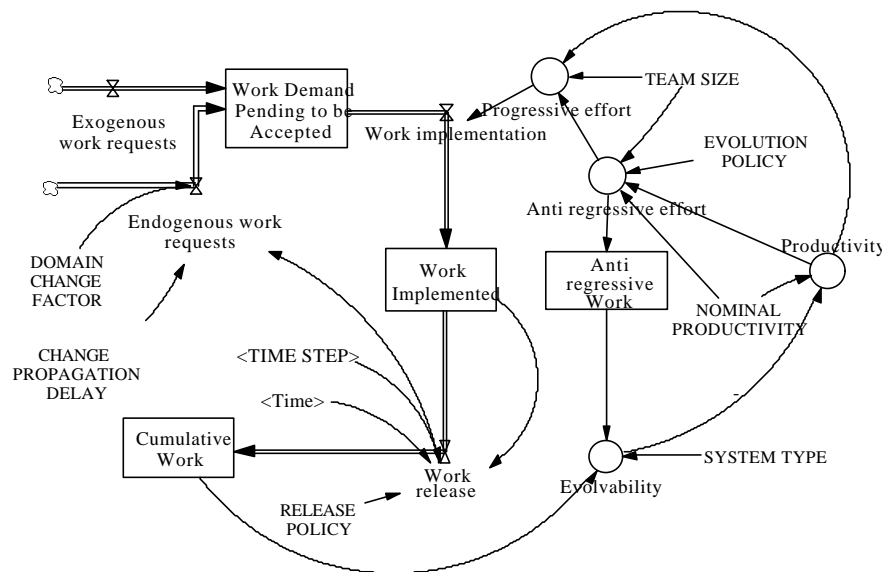


Fig. 1 - A High Level of Aggregation SD Model of an Ideal Software Evolution Process⁵.

A systematic approach to model-based monitoring of evolvability may be achieved by applying concepts from the *change detection* field [bas93,box97,pat94]. The approach is applicable to black-box and to white-box models. This encompasses a set of statistical techniques that have already found application as diagnostic aids and fault detection as part of navigation systems, biomedical systems and in chemical processes, for example. Some of the change detection techniques have found application in statistical quality control, for example [box97]. They cannot be discussed in detail here. Preliminary inspection suggests that such concepts may be applied to monitor productivity and evolvability changes and changes in other key attributes of the evolution process. The general idea of such approaches is illustrated in Fig. 2. The change detection scheme, which may involve a model calibrated and believed to represent the evolution process, is run 'in parallel' with the real process it models. The inputs and outputs of the process are statistically compared to model's prediction. A significant change in evolvability (or other key parameter) detected by such a scheme may prompt, for example, the need for process revision, major restructuring of the evolving software or even software replacement.

In summary, it is suggested here that models, and in particular, cost estimation models, may provide a useful tool to define and monitor evolvability at a high level of aggregation. Of course, the hypothesis that evolvability has changed must be also contrasted with other possible hypotheses. For instance, after observing a divergence (by any of the means suggested above or other) one may ask whether it is just a change of a parameter or whether the model structure is still reflecting the process. Structural changes in the evolution process and/or significant changes in the evolution rate may invalidate the model, in part or as a whole, and hence, invalidate any inferences drawn from it. What has actually changed in the real-world process? The change detection field provides numerous techniques to address a variety of models and situations. Though in a domain as the software process in which accurate models of process behaviour tend to be more the exception than the rule, one must proceed with caution in their application. White-box techniques may be increasingly relevant at the higher levels of process maturity. Less mature processes may be effectively monitored by means of simple black-box models.

⁵ See [kah00] for further details.

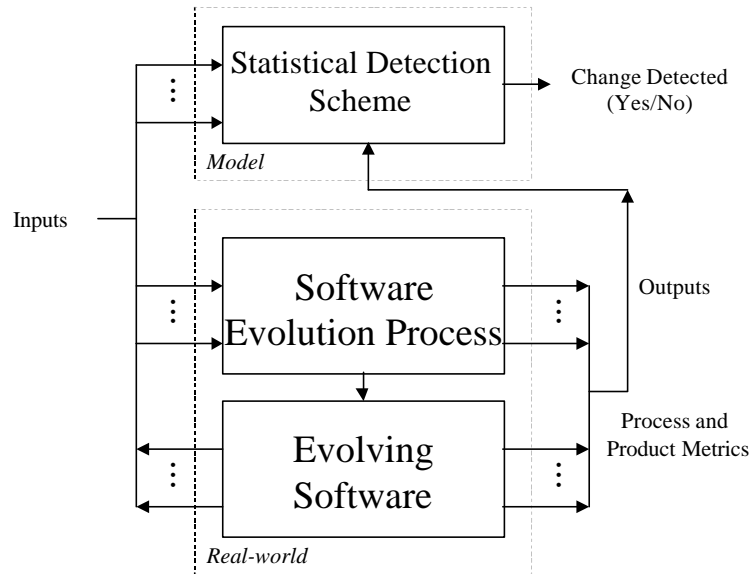


Fig. 2 - A Model-based Scheme for Change Detection

FINAL REMARKS

In view of the ever spreading use of computers in all aspects of human activity, collectively and individually, and the critical role that computers play in many of the applications, post first release evolution in timely fashion is becoming an ever more important challenge. In many situations today the time available for system adaptation, to maintain satisfactory operation is critical. At the same time, the major cost that system evolution represents in the lifetime of the system and the demands it makes on professional resources, makes it essential to be able to accurately predict, assess and control the cost of adaptation and evolution. It is perhaps surprising that so little emphasis appears to have been placed on estimation and planning of the evolution activity. We recognise that the present paper represents initial thoughts on this important topic which we put before the workshop as a challenge.

ACKNOWLEDGEMENTS

Many thanks are due to Dr Goel Kahen, a member of the FEAST/2 team, to Professors D Perry and Wlad Turski, Senior Visiting Fellows, and to the FEAST/2 industrial collaborators for helpful discussions. Financial support from UK EPSRC, grant n. GR/M44101 (FEAST/2) is gratefully acknowledged.

REFERENCES - Papers indicated with a '*' are available via links at <http://www-dse.doc.ic.ac.uk/~mml/feast>

- [bas93] Basseville M and Nikiforov IV, *Detection of Abrupt Changes: Theory and Application*, PTR Prentice Hall, Englewood Cliffs, NJ, 1993, 528 pp
- [ben00] Bennett KH and Rajlich VT, *Software Maintenance and Evolution: a Roadmap*, in A. Finkelstein (ed.), *The Future of Software Engineering*, ICSE 2000, June 4-11, Limerick, Ireland, pp 75 - 87
- [boe81] Boehm B, *Software Engineering Economics*, Englewood Cliffs, N.J, Prentice-Hall, 1981
- [boe00] Boehm BW and Sullivan KJ, *Software Economics: A Roadmap*, in Finkelstein A (ed.), *The Future of Software Engineering*, ICSE 22, pp 321 - 343
- [box97] Box G and Luceño A, *Statistical Control by Monitoring and Feedback Adjustment*, Wiley, NY, 1997, 327 pp
- [bri92] Briand L and Basili V, *A Classification Procedure for an Effective Management of Changes during the Software Maintenance Process*, ICSM 92, Orlando, Florida, USA
- [coc00] COCOMO *Constructive Cost Model*, web site, <http://sunset.usc.edu/>
- [coy96] Coyle RG, *System Dynamics Modelling - A Practical Approach*, Chapman & Hall, London, 1996, 413 p

- [feast] FEAST/2, *Feedback, Evolution and Software Technology*, web site, <http://www-dse.doc.ic.ac.uk/~mml/feast/>
- [gra80] Graham AK, *Parameter Estimation in System Dynamics Modelling*, in Legasto A.A. Jr, Forrester Jw and Lyneis JM (eds.) *System Dynamics*, Vol. 14, TIMS Studies in the Management Sciences, North Holland, Amsterdam, 1980, pp 125 - 142
- [gra97] Granja-Alvarez JC and Barranco-Garcia MJ, *A Method for Estimating Maintenance Cost in a Software Project: A Case Study*, *J. Software Maintenance: Res. And Practice*, vol. 9, 1997, pp 161 - 175
- [guj95] Gujarati DN, *Basic Econometrics*, 3rd. Edition, Mc Graw Hill Inc., New York, 1995, 838 pp
- [ieee93] *IEEE Std. 1219: Standard for Software Maintenance*, IEEE Computer Society Press, Los Alamitos CA, 1993
- [jor95] Jorgensen M, *Experience with the Accuracy of Software Maintenance Task Effort Prediction Models*, *IEEE Transactions on Software Engineering*, Vol. 21, No. 8, August 1995, pp 674 - 681
- [leg80] Legasto AA Jr., and Maciarello J, *System Dynamics - A Critical Review*, in Legasto A.A. Jr, Forrester J and Lyneis JM (eds.) *System Dynamics*, Vol. 14, TIMS Studies in the Management Sciences, North Holland, Amsterdam, 1980, pp 23 - 43
- [leh74] Lehman MM, *Programs, Cities, Students, Limits to Growth?*, Inaugural Lecture, in Imperial College of Science and Technology Inaugural Lecture Series, Vol. 9, 1970, 1974, pp 211 - 229. Also in *Programming Methodology*, (D. Gries. ed.), Springer Verlag, 1978, pp 42 - 62. Reprinted in [leh85].
- [leh85] Lehman MM and Belady LA, *Software Evolution - Processes of Software Change*, Academic Press, London 1985
- [leh94] Lehman MM, *Feedback in the Software Evolution Process*, Keynote Address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics. Dublin, 7-9 Sept. 1994, Workshop Proc., also in *Information and Software Technology*, sp. is. on Software Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp 681 - 686
- [leh98]* Lehman MM and Ramil JF, *Implications of Laws of Software Evolution on Continuing Successful Use of COTS Software*, DoC Tech. Rep. 98/8, Imperial College, London, Jun. 1998
- [leh00]* Lehman MM, *Rules and Tools for Software Evolution Planning and Management*, Pre-prints FEAST 2000 Workshop, Imp. Col., 10 - 12 Jul. 2000, pp 53 - 68
- [par94] Parnas DL, *Software Aging*, Proc. 16th ICSE, May 16-21, 1994, Sorrento, Italy, 279-287
- [pat94] Patton R, Frank P and Clark R (eds.), *Fault Diagnosis in Dynamic Systems - Theory and Application*, Prentice Hall Intl., NY, 1989
- [raj00] Rajlich VT and Bennet KH, *A Staged Model for the Software Life Cycle*, *Computer*, July 2000, pp 66 - 71
- [ram00a] Ramil JF, *Algorithmic Cost Estimation for Software Evolution*, Proc. ICSE 2000 Doctoral Workshop, Limerick, Ireland, 6 June 2000, pp 701 - 703
- [ram00b] Ramil JF and Lehman MM, *Effort Estimation from Change Records of Evolving Software*, Proc. ICSE 2000, Poster Summary, June 4 -11th, Limerick, Ireland, pp 777
- [ram00c] Ramil JF, Lehman MM and Kahen G, *The FEAST Approach to Quantitative Process Modelling of Software Evolution Processes*, Proc. PROFES 2000, 2nd International Conference on Product Focused Software Process Improvement, Oulu, Finland, June 20 - 22, 2000, in Frank Bomarius and Markku Oivo (eds.) LNCS 1840, Springer Verlag, Berlin, 2000, pp 311 - 325
- [ram00e]* Ramil JF, *Why COCOMO Works' Revisited or Feedback Control as a Cost Factor*, Pre-Prints FEAST 2000 Workshop, Imp. Col., London, 10 - 12 Jul. 2000, pp 89 - 94
- [ram00d] Ramil JF, Lehman MM, *Metrics of Software Evolution as Effort Predictors - A Case Study*, Proc. ICSM 2000, Int. Conference on Software Maintenance, 11-14 Oct. 2000, San Jose, CA, forthcoming.
- [rob99] Robertson J and Robertson S, *Mastering the Requirement Process*, Addison-Wesley, London, 1999
- [she96] Shepperd M, Schonfied C and B Kitchenham, *Effort Estimation Using Analogy*, Proc ICSE 18, 1996, pp 170 - 178
- [star98] Stark GE, *Software Maintenance Lessons Learned*, WESS 98, Bethesda MD, Nov. 16, 1998, pp 17 - 19
- [stau98] Staudenmayer N, Graves T, Marron JS, Mockus A, Siy H, Votta L and Perry D, *Adapting to a New Environment: How a Legacy Software Organization Copes with Volatility and Change*, 5th Int. Product Dev. Manag. Conference, Como, Italy, May 1998
- [ven95] *Vensim Reference Manual*, Ver. 1.62, Ventana Systems Inc., Belmont, MA, 1995
- [wal96] Wall L, et al, *Programming Perl*, O'Reilly & Associates, Sebastopol, CA, 1996, 645 pp
- [zur67] Zurcher FW and Randell B, *Iterative Multi-Level Modeling - A Methodology for Computer System Design*, IBM Res. Div. Rep. RC-1938, Nov. 19678. Also in Proc. IFIP Congress 1968, Edinburgh, Aug 5 - 10, pp D-138 - 142