Dynamic Modelling in the Investigation of Policies for *E*-type Software Evolution

Extended Abstract submitted to ProSim'2000, June 12 -14, 2000, Imp. Col., London UK

G Kahen MM Lehman JF Ramil
Department of Computing
Imperial College of Science,
Technology and Medicine
180 Queen's Gate, London SW7 2BZ
tel. +44 20 7594 8214 fax +44 20 7594 8215
{gk, mml, ramil}@doc.ic.ac.uk,

P Wernick
Department of Computer Science
University of Hertfords hire
College Lane
Hatfield, AL 10 9AB
tel. ++44 1707 286323 fax ++44 1707 284303
p.d.wernick@herts.ac.uk

Keywords

System dynamics, software process modelling, simulation, feedback, FEAST, *E*-type systems.

Introduction

Observations that led to the identification of the concept of software evolution were first collected during studies between 1968 [leh69] and 1985 [leh78,85] of the evolution of OS/360-70 and other systems between the mid-1960s and mid-1970s. This work led, for example, to the classification of software into types *S* and *E*. The latter are systems in use in a real-world domain and have the intrinsic property that they require to be continually evolved as long as they are in regular use [leh85]. Otherwise they deteriorate in effectiveness. Based on the observations and the interpretation at that time, three dynamic models [bel75,rio77,woo79] of the evolution process were developed. More recently, with the active collaboration of ICL, Logica and Matra BAe Dynamics the FEAST/1 project (1996-1998) [fea00] has been able to substantiate, refine and extend the earlier observations. This was made possible by analysis of data provided by these companies on the evolution of their respective systems, VME Kernel, the FW Banking Transaction system and a defence system. Data on two real time Lucent Technologies systems also became available for analysis during this time. In FEAST/1, system dynamics (SD) [for61,coy96] and the Vensim tool [ven95] were used to build models of two of the industrial software processes being investigated [wer98,cha99]. The model building work followed a top-down approach, and has produced models which are relatively simple when compared with other SD models, for example the one in [mcc99]. The focus of the investigation was to achieve a degree of *understanding* of the long term dynamics of the *global* software process, that is, involving the activities of developers, users, marketeers, support personnel and their managers in addition to technical activities. The on-going FEAST/2 project (1999 - 2001) [fea00], with BT Labs as an additional collaborator, is exploring, *inter alia*, means to support software evolution management decisions. It is approaching the problem by using SD models as decision support tools, and by seeing policies as feedback mechanisms whose behaviours and effects can, to certain extent, be engineered. This work aims at achieving effective management of long-term software evolution. The approach combines earlier results [bel75,rio77,woo77], the insights gained in FEAST/1 [wer98,cha99] and those being obtained in FEAST/2.

Elements in Software Evolution Decision Support

In SD modelling one identifies the segments or areas to be covered by the model, as for example in [mcc99]. The following list identifies a set of elements considered by the present authors in the investigation of long term software evolution policies:

- CONSTRAINTS - formalisation of *relevant* human, process, organisational, managerial bounds and limits. General constraints have been identified and encapsulated, for example, in Brook's law [bro75,95, page 274], the laws of software evolution [leh74,85,fea00]**.** Other general or local, such as financial and technical constraints, may also have to be considered.
- DYNAMICS OF APPLICATION DOMAIN - characterisation of the application domain in terms, for example, of volatility of requirements, rate of arrival of change requests, changes in the user population (e.g. number of users), user response to change (e.g. constant or function of release size and content).
- DYNAMICS OF TECHNOLOGY - same as above, but related to changes in the technology (software, hardware, process, etc) domains.
- EFFORT - characterisation of human resources applied to different aspects of system evolution, such as development, support, sales, etc. This reflects the size of teams, familiarity, experience, etc.
- EVOLUTION PROCESS - attributes of the process that are relevant in the investigation of policies, such as implementation, validation and rework rates.
- PAYOFF FUNCTION – the relationship to be optimised over the long term. In principle, one would assume the point of view of managers that seek to optimise, for example, the power of a system, stakeholder satisfaction, or the benefits to be derived from a system over its lifetime.

- POLICY DECISIONS - formalisation of the set of feedback-based rules to be enacted by management for the control of long term evolutionary attribute, such as growth rate, fault rate and evolution cost, including release timing & content, staffing, partial or full system replacement, etc.
- STAKEHOLDER SATISFACTION - set of attributes that reflect the degree of achievement of stakeholders' desires with regards to the system. It could be part of the payoff function or not, depending on whether its optimisation is actively sought. An alternative formulation is to seek that stakeholders' satisfaction meets or exceeds some minimum threshold.
- SYSTEM CATEGORY - characterisation of the application with regards to different dynamic behaviours. This is currently being investigated and is expected to account for some at least of the differences in dynamic behaviour arising from, for example, different application domains. In this way, 'System Category' would be analogous, in a long term evolution context to, for example, the differences in project productivity that led to the three COCOMO categories (embedded, semi-detached and organic) [ccm00].

The list identifies elements present, explicitly or implicitly, in previous models [bel75,rio77,woo79,wer98, cha99]. The ones believed to be new in the present work, are payoff function, stakeholder satisfaction and system category.

A Dynamic Model

In general the software evolution process encompasses activities that include fixing, enhancement, adaptation, addition and removal. Following Baumol's classification [bau67] of work effort into *progressive* and *anti-progressive* types, Lehman suggested a further category, *anti-regressive* [leh74,85]. The activities that enhance (by addition or modification) system functionality were termed progressive. The term anti-regressive was used to refer to work effort intended to compensate for the effects on the software product of the addition of change upon change upon change, growing complexity and other aging effects [leh74,85,par94], for example. Anti-regressive work consumes effort without any immediate visible stakeholder return, for example, in system value as reflected by system functional power or performance. Instead, they facilitate further evolution. Several other classifications have been proposed, that include, for example, defect fixing as a separate category. In the SD model below it is considered that defect fixing is progressive.

The approach and the SD models being developed in FEAST/2 and presented here are intended to demonstrate the impact of policy decisions on software evolution attributes. The SD model in Fig. 1 provides a vehicle to explore the long-term impact of policies with regards to anti-regressive activities.

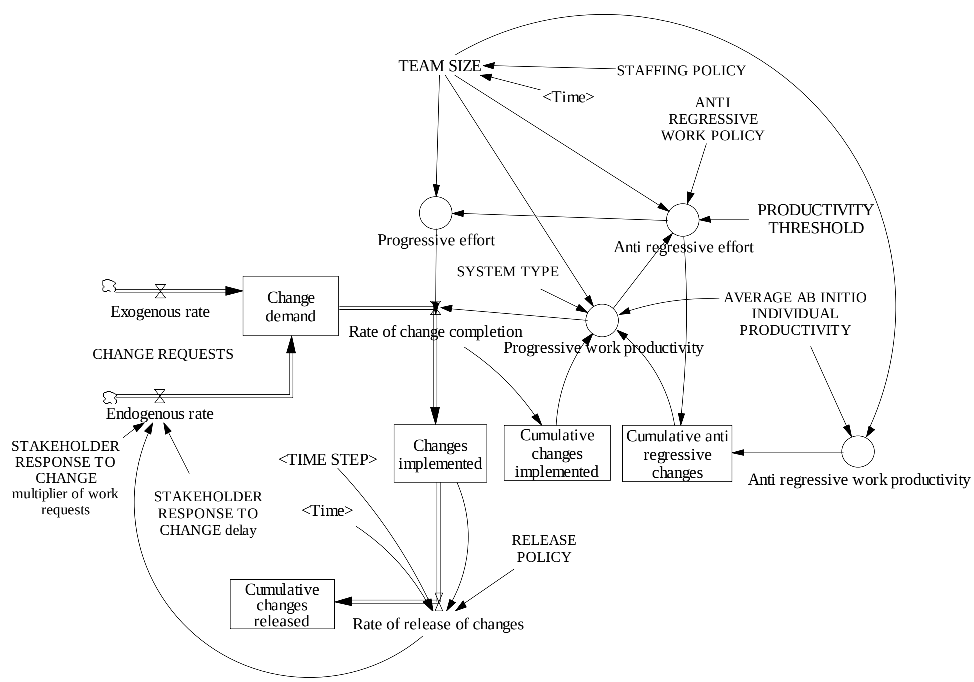

Figure 1 - System Dynamics Model to Investigate the Impact of Anti-Regressive Work Policies

The SD model is fully executable in Vensim [ven95]. It will be explained in detail in the full paper and can be demonstrated to interested participants. It considers the software evolution process as a process of implementation of *changes*. The term 'change' is used here in a wide sense to indicate not only changes to existing functionality and performance, but also the addition of new functions to the system. Note that in this model, not all of the elements mentioned in the previous section have been included. The excluded ones are currently assumed either to be constant and not impacting dynamic behaviour, or not being influential in the context of the particular policy being investigated. This is consistent with the FEAST SD modelling approach. This follows a top-down approach and successive refinement [zur67,wir71].

In Fig.1, ‘Anti Regressive Work Policy’ represents the proportion of team size assigned to such work. It may take any value between 0 (no resource assigned to anti-regressive work) and 1 (all resources applied to that activity). The model includes a management control mechanism that activates the anti-regressive work when a decrease in productivity occurs with respect to a 'Threshold Productivity'. The model considers a non-linear relationship between productivity and team size as suggested by the Brook's law [bro75,95] that will be explained in the full paper. Its output (Fig. 2), with ‘Anti Regressive Work Policy’ set to 0, reproduces the growth trend of one of the systems studied in FEAST/2 over its 180 months lifetime to late 1999, closely.

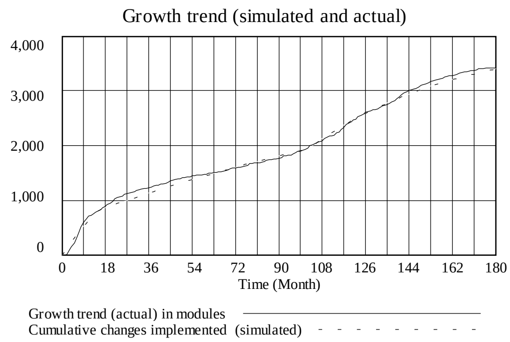

Figure 2 - Growth trend and model simulated output for one of the systems studied in FEAST/2

As an example of the use of the model for policy evaluation, Figs. 3, 4 show its significant impact on the predicted behaviour of productivity over the next 90 months of system evolution. In particular, Fig. 4 shows model's prediction that 30% of resources assigned to anti-regressive work will result in significant extension the potential system life span. Applying the successive refinement approach, this is an initial policy. Refined versions will be explored after further calibration of this model.

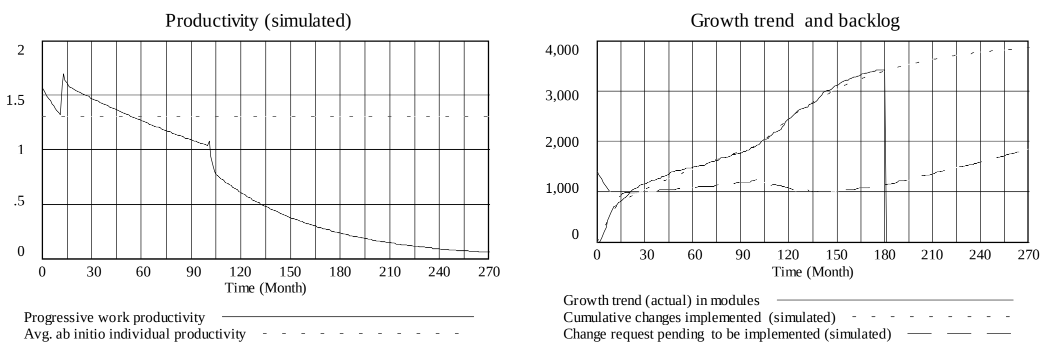

Figure 3 - Simulated Process Behaviour with No Resources Assigned to Anti Regressive Work

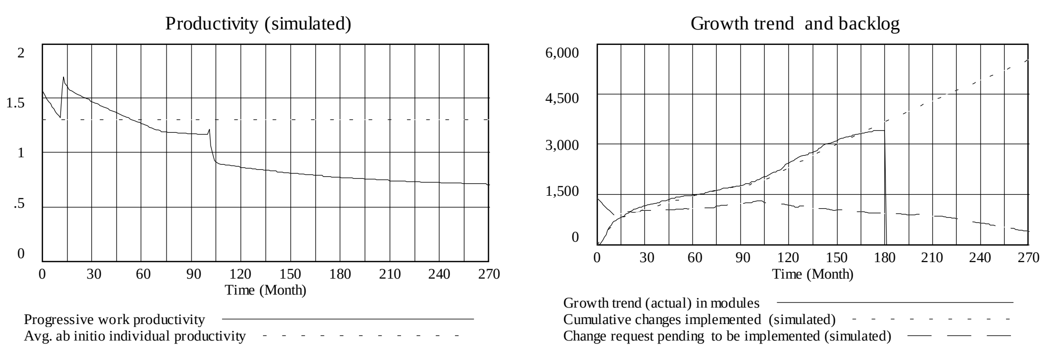

Figure 4 - Simulated Process Behaviour with 30% of the Resources applied to Anti Regressive Work

The full paper will summarise lessons learnt and will include comments on related work by others, as for example on the long term models proposed in [ara93,mcc99]. It is believed that by this summary of concepts relevant to the determination of policies for managing *E*-type system evolution, and by exemplifying such investigation by means of a SD model, this work provides foundations of a systematic approach to long-term software evolution planning and control.

Acknowledgements

Grateful thanks are due to our industrial collaborators and to Profs. Dewayne Perry and Wlad Turski, SVFs to FEAST/2, for their comments on early versions of the model presented here. Grateful thanks are also due to Ms. Siew F. Lim for her help in checking the references. Financial support from the UK EPSRC, grant number GR/M44101 (FEAST/2 Project), is gratefully acknowledged.

References

[ara93] Aranda RR, Fiddaman T and Oliva R, *Quality Microworlds: Modeling the Impact of Quality Initiatives over the Software Product Life cycle*, American Programmer Vol. 6, No. 5, May 1993, pp. 52 - 61

[bau67] Baumol WJ, *Macro-Economics of Unbalanced Growth - The Anatomy of Urban Cities*, Am.. Econ. Review, Jun 1967, pp. 415 - 426

[bel75] Belady L and Lehman MM, *The Evolution Dynamics of Large Programs*, IBM Res. Rep. RC5615, T J Watson Res. Centre, Yorktown Heights, NY 13598, Sept. 9 1975, 45 pps.

[bro75,95] Brooks FP, *The Mythical Man-Month*, 20^{th} Aniv. Edition, Addison-Wesley, Reading, MA, 1995, 322 pps.

[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 - Improvement and Practice in 2000

[ccm00] COCOMO Web site, http://sunset.usc.edu/COCOMOII/suite.html

[coy96] Coyle RG, *System Dynamics Modelling - A Practical Approach*, Chapman & Hall, London, 1996, 413 p

[leh69] Lehman MM., *The Programming Process*, IBM Research Report RC 2722, IBM Research Center, Yorktown Heights, NY, Sept. 1969

[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

[leh78] Lehman MM, *Laws of Program Evolution—Rules and Tools for Programming Management*, Proceedings of the Infotech State of the Art Conference, Why Software Projects Fail, Apr. 1978, pp. 11/1-11/25

[leh85] Lehman MM and Belady LA, *Program Evolution - Processes of Software Change*, Academic Press,1985, pp 538

[mcc99] McCray GE and Clark TD, *Using System Dynamics to Anticipate the Organisational Impacts of Outsourcing*, System Dynamics Review, Vol. 15, No. 4, (Winter 1999), pp. 345 - 373

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

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

[par94] Parnas DL, *Software Aging*, Proc. 16th Int. Conf. on Softw. Engineering, May 16-21, 1994, Sorrento, Italy, pp 279-287

[rio77] Riordan JS, *An Evolution Dynamics Model of Software Systems Development*, in Software Phenomenology - Working Papers of the (First) SLCM Workshop, Airlie, Virginia, Aug 1977. Pub. ISRAD/AIRMICS, Comp Sys Comm US Army, Fort Belvoir, VI, Dec 1977 , pp 339 - 360

[ven95] *Vensim - Ventana Simulation Environment*, Reference Manual, Version 1.62, 1995

[wer98] Wernick PD and Lehman MM, *Software Process White Box Modelling for FEAST/1*, ProSim '98 Workshop, Silver Falls, OR, 23 June 1998. Also in the Journal of Systems and Software, Vol. 46, Numbers 2/3, April 1999

[wir71] Wirth N, *Program Development by Step-wise Refinement*, Comm. ACM, Vol. 14, No. 4, April 1971, pp. 221 - 227

[woo79] Woodside CM, *A Mathematical Model for the Evolution of Software*, ICST CCD Res. Rep 79/55, Apr. 1979. Also in J Sys and Software, Vol 1, No 4, Oct 1980, pp 337 - 345, and as Chapter 16 in Lehman MM and Belady LA, Program Evolution - Processes of Software Change, Academic Press, 1985, pp 538

[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 Congr. 1968, Edinburgh, Aug 1968, pp D-138 - 142