

## Quantitative Studies in Software Evolution - From OS/360 to FEAST

Position Paper: Workshop on Process Modelling and Empirical Studies of Software Evolution,  
Boston, MA  
May 18th 1997  
M M Lehman  
Department of Computing  
Imperial College  
London SW7 2BZ  
tel: +44 (0)171 594 8214  
fax: +44 (0)171 594 8215  
mml@doc.ic.ac.uk

### 1 Early Work in Software Evolution

A 1968/9 study of the IBM programming process [leh69,85] led, *inter alia*, to the collection and analysis of metric data relating to the evolution of the IBM OS/360 programming system over a sequence of releases. The results of this work were summarised in a series of papers the most important of which were subsequently republished [leh85].

The study was based on the examination of system measures such as the sizes of successive releases, numbers of modules handled during each release cycle, release dates, release costs and so on. The data was examined by means of plots of the data sequences and of their derivatives against time. Time was measured by dates or, more frequently, in pseudo time expressed as a sequence number (RSN) associated with each release. Final results were summarised in [leh80b] which included also brief analyses of data from systems other than OS/360. Some simple statistical analysis was also attempted. There were, however, only 14 data points (as identified by their RSN) available at the commencement of the study and some 24 on its completion. It was, therefore evident from the start that while statistical techniques might give some indication of trends, they could not be expected to yield statistically significant results. The one exception to this was a study by Chong [cho81] that showed that one of the dependent variables was normally distributed.

The most remarkable aspect of the time behaviour of the evolution metrics was the regular patterns it displayed. Already in the first stages of the study the growth of the system over time was seen to display characteristic and repetitive features, for example a "... ripple ... typical of a self stabilising process with positive and negative feedback loops. From a long-range point of view the rate of system growth is self-regulatory, despite the fact that many different causes control the selection of work implemented in each release, with varying budgets, increasing numbers of users reporting faults or desiring new capability, varying management attitudes towards system enhancement, changing release intervals and improving methods..."[bel72]. At a latter stage it was suggested that one should "... regard the organisation developing and maintaining a large program as a *system in the system theoretic sense*. ... Observation has shown that the system behaves as a self stabilising feedback system. ... The process leads to an organisation and a *process dominated by feedback* ... with long range trends ... and invariances ..." [leh78]. The above quotations show that a characteristic only now encapsulated in an eighth law of software evolution dynamics was recognised over twenty five years ago.

Consideration of the various plots led to a series of consistent phenomenological interpretations that explained the evolutionary patterns observed. This reflected, *inter alia*, the realisation that human behaviour and organisational factors were, at least in part, responsible for the observed behaviour. That is, interpretation of the data suggested that the characteristics of system evolution were, in part at least, determined by factors other than the technology being used. Thus, when the observations and their implications were summarised and encapsulated in a series of statements reflecting the behaviour of people and organisations, these were seen as lying outside software technology. From the point of view of the software engineer they were to be viewed as *laws*. The first three of these laws were formulated in the mid seventies [leh74] and discussed in greater detail in 1978 [leh78]. Two further laws were introduced in 1980 [leh80a]. A sixth was introduced in a subsequent footnote [leh91]. The remaining two while frequently discussed in lectures have only recently been published [leh96b,c]. As restated<sup>1</sup> explicitly below the the full set of laws relate to *E*-type software [leh80b] that is, broadly speaking, to software systems that solve a problem or implement a computer application in the *real world*.

<sup>1</sup> Numbered in order of formulation and publication. Over the years the names and wording of the laws have been modified. The fundamental understanding they reflect remains, however, essentially the same although the focus of attention may have shifted.

The laws as developed over the years are, for convenience, summarised in table 1.

No.	Brief Name	Law
I 1974	Continuing Change	An <i>E</i> -type program that is used must be continually adapted else it becomes progressively less satisfactory
II 1974	Increasing Complexity	As an <i>E</i> -type program evolves its complexity increases unless work is done to maintain or reduce it
III 1974	Self Regulation	The <i>E</i> -type program evolution process is self regulating. Distribution of product and process attribute measures are close to normal
IV 1980	Conservation of Organisational Stability (invariant work rate)	The average effective global activity rate in an evolving <i>E</i> -type system is invariant over the product lifetime
V 1980	Conservation of Familiarity	During the active life of an evolving <i>E</i> -type program the average content of successive releases is invariant
VII 1980	Continuing Growth	The functional content of an <i>E</i> -type program must be continually increased to maintain user satisfaction over its lifetime
VII 1996	Declining Quality (Formulated in 1996 but implicit in earlier laws.)	<i>E</i> -type programs will be perceived as of declining quality unless rigorously maintained and adapted to a changing operational environment
VIII 1996	Feedback System (Formulated in 1996 but recognised already in 1972.)	<i>E</i> -type programming processes constitute multilevel, multi loop, feedback systems and must be treated as such to achieve significant improvement over any reasonable base

The purpose of this position paper is not, however to discuss the laws in detail. For this the reader is referred to earlier publications [leh74,78,80a,85,96b,c]. Nor is it to discuss the reception they and the work on which they were based was given by the Software Engineering community [law82]. The intent is to draw attention to the quantitative approach then employed and to indicate how it is now being used once again in a renewed, systematic and more detailed study of the software evolution phenomenon. First, let it be said that following the work of the 70s as summarised in [leh80a,85] quantitative investigations of software system evolution lay dormant for over a decade, for a number of reasons. Instead software engineering interest spread and developed with the software *process* becoming a major focus of interest. The software process had for some time been the subject of study by individual investigators [leh69, roy70, boe76]. Wider interest was engendered by the first [spw84] of a series of International Process workshops (that are still running and for which Proceedings are also available from IEEE), by Osterweil's keynote address to ICSE9 [ost87] and by a response to that address [leh87]. These events triggered a host of activity in process studies focussing primarily on process modelling in general and process programming in particular. These procedural studies paid little attention to and threw little light on the behavioural characteristics of the process or on its evolutionary properties.

## 2 The FEAST Hypothesis

Some years ago, the present author pondered the question as to why despite many significant innovations at the technical level of software development, innovations that include a more rigorous approach to process design, newly disciplined process steps, new paradigms such as structured programming and the object oriented paradigm, a host of new languages at ever higher levels and ever more sophisticated support tools the global industrial process still leaves much to be desired.

The question having been asked, an answer immediately became apparent. It appeared plausible that the solution might lie in the process attributes encapsulated in the Laws of Software Evolution in general and, in particular, in the observation (as quoted above) that the software process constituted a feedback system. Those laws are, in fact, increasingly understood as a direct aspect of the strong feedback system nature of the software process. Furthermore, an intrinsic property of feedback systems is that, to a degree dependent on the number and nature of the feedback loops, changes in the externally visible behaviour of the system will, in general, be significantly less than might be expected from the extent of the impact that changes have at their point of application. This, of course, describes precisely what has been observed, though not always appreciated, as the software process has evolved over the past decades.

This hypothesis was formalised under the name FEAST and an International workshop, the FEAST (*Feedback, Evolution And Software Technology*) [fea94] was organised to explore the hypothesis and its

implications. The interest aroused led to two further FEAST workshops [fea94,5] and, subsequently to a successful research proposal FEAST/1[leh96a] underway since October 1996.

### 3 FEAST/1

In its most recent formulation [leh96a] the hypothesis states:

*As for other complex feedback systems, the dynamics of real world software development and evolution processes will possess a degree of autonomy and exhibit a degree of global stability.*

It is not possible to discuss the hypothesis in detail in the present paper and the interested reader is referred to earlier publications [leh96]. Suffice it to say that the opening words imply the feedback system nature of the software process. The remainder suggest that consequent process (ie system) properties must have a significant impact on the techniques used for process modelling and to achieve process improvement. This paper draws attention to the metric based approach being taken in FEAST/1 to testing the validity of the hypothesis in industrial software evolution (ie development and maintenance) environments and to identifying significant feedback loops and mechanisms, their impact and how changes to them might be used to improve the processes. The problem being investigated is formidable involving, as it does, human driven and controlled feedback. It must, therefore, be expected that generalisation of the results, development of generic models of feedback and the formulation of an appropriate process theory will have to await more widespread, active interest, additional funding and follow-on projects.

A FEAST/1 project description will be found in [leh96a]. The quantitative analysis methods to be employed are based on the pioneering 1970s studies of OS/360 evolution. With strong support from its industrial collaborators, ICL, Logica, Matra-BAe and the Ministry of Defence the project will, in *black box* studies, examine the evolution of a variety of systems by studying the change over time of various metrics. Statistical and other techniques are being used to identify feedback like behaviour, its quantitative properties and its effect on the evolution of the respective systems. Once such feedback-like behaviour has been identified in, at least some of the systems, the new study will go beyond earlier work and seek to identify specific feedback control loops in those processes and their relationship to process characteristics and attributes. Development of *white box* models of the global process with its loop mechanisms using system dynamics [abd91, mad96] and distributed multi agent techniques [oha96] will permit identification of the impact of feedback and the design of changes to the feedback mechanisms to yield global process improvement. Proposed changes and their quantitative consequences will then be evaluated. Conclusive results are, however, unlikely to become available over the lifetime of this first (FEAST/1) investigation.

### 4 Early Results

At the time of writing, the FEAST/1 project has been running for only some three months but encouraging results are already available. In the main these derive from data on a Logica plc transaction system *FW* with additional evidence from a telephone switch subsystem. It is important to note that each of these is substantially different in nature to the OS/360 operating system. This refutes one of the main criticism of the earlier work that claimed that the phenomenology of OS/360 evolution could not be construed as applying more widely because of the unique nature of IBM, of operating systems in general, of OS/360 in particular and of the technology employed by IBM. The general conclusion from initial FEAST/1 results is that the observed behaviour is, to a remarkable degree, similar to that demonstrated by OS/360. Its interpretation supports, or at least does not contradict, the phenomenology encapsulated in the laws of software evolution [tur96, leh96].

As a preliminary illustration three corresponding plot pairs of the OS/360 and the FW data are reproduced below. These illustrate various time dependent behavioural trends, time being measured by release sequence numbers. The alert reader may notice that the FW plots reproduced here differ slightly from those in earlier publications [tur96, leh96]. This is due to errors in the original data which have now been corrected. No extensive discussion of these plots is provided here nor are the behavioural patterns presented in any way exhaustive. The intent is rather to exemplify the degree to which the two systems, so widely separated in nature and in time, share common behavioural features. It is features such as these that led to the formulation of the laws of software evolution and, more recently, to the FEAST hypothesis. The FEAST/1 activity of the next two years and any follow on investigations will explore the common evolutionary features, the differences between systems and the implications of both more completely.

Figures 1 and 2 indicate the evolutionary growth trend of the two systems, a trend that clearly reflects the first and seventh law of software evolution dynamics. It is, however, not possible from this data to determine the

degree to which the growth is due to each of the two phenomena. The ripple on each of the two curves is that referred to in the extracts from 1970s publications [bel72, leh78] in section 1. It suggests that the evolutionary growth is stabilised by feedback controls as per the FEAST hypothesis and the eighth law.

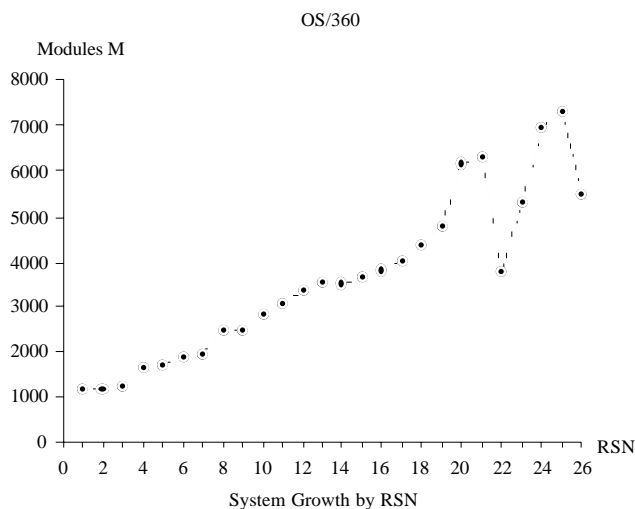


Fig. 1a 1970s System - OS/360

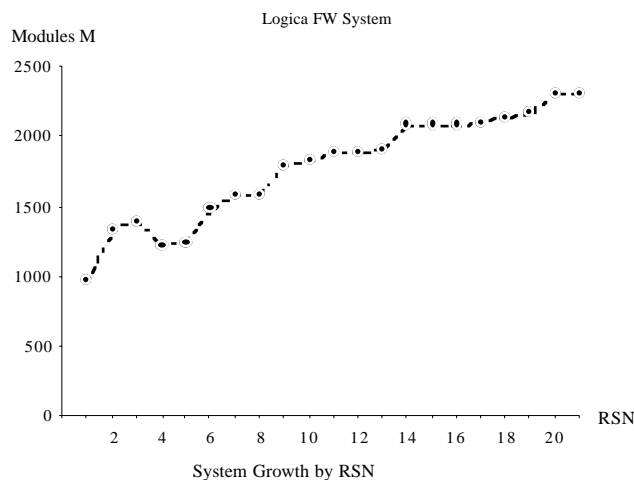


Fig. 1b 1990s System - Logica FW

Figure 2a is the plot from which the fifth law was inferred. The corresponding plot 2b for FW shows strikingly similar behaviour though many questions remain to be answered. Given the phenomenological interpretation associated with the observed behaviour [leh80] the plots represent an encouraging start to the FEAST/1 investigations.

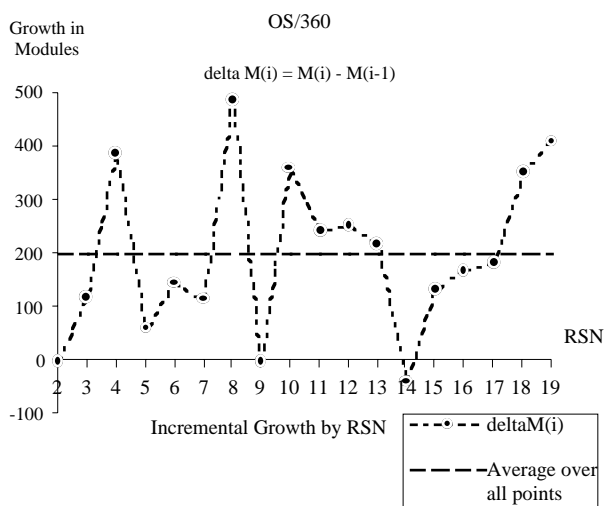


Fig. 2a 1970s System - OS/360

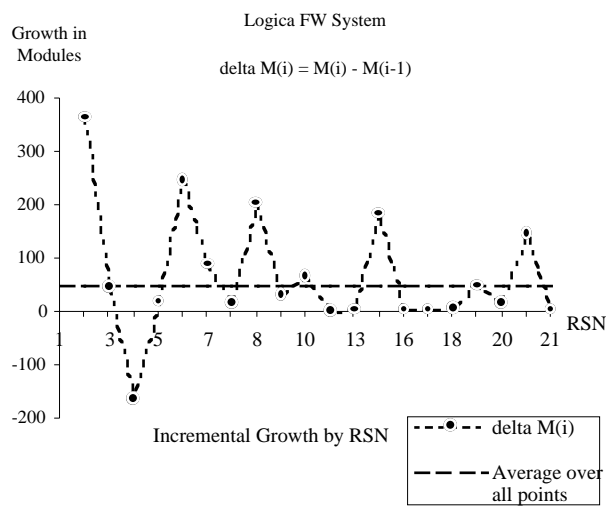


Fig. 2b 1990s System - Logica FW

On the basis of analysis to date the OS/360 gross system growth trend is best modelled by a linear fit. Gross FW evolution, on the other hand, is best fitted by a single parameter inverse square model. Plots 3a and 3b indicates the absolute error and standard deviation of the model *prediction* relative to the observed data as a function of the point in the release sequence at which the model parameter is estimated. That is, the abscissa P is the sequence number of the final release used in estimating this parameter. The plots clearly show the strength of the developing system dynamics [tur96]. For FW in particular it is observed that after some six releases the mean absolute error and standard deviation of the model value converge to constant values. It takes only some five or so releases to establish the evolutionary pattern. After that, the main growth trend is determined by the system dynamics rather than by managerial edict.

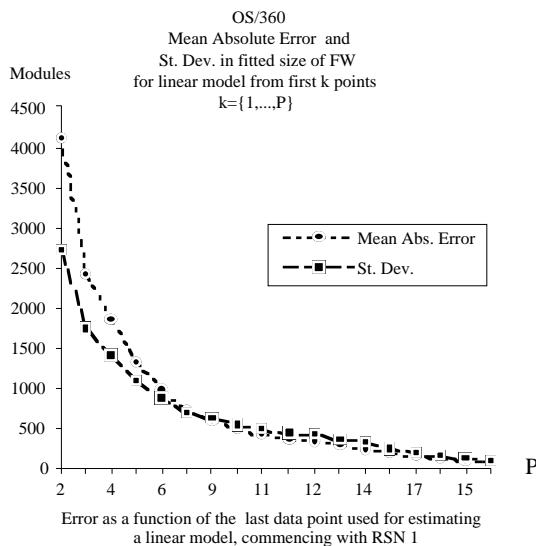


Fig. 3a 1970s System - OS/360

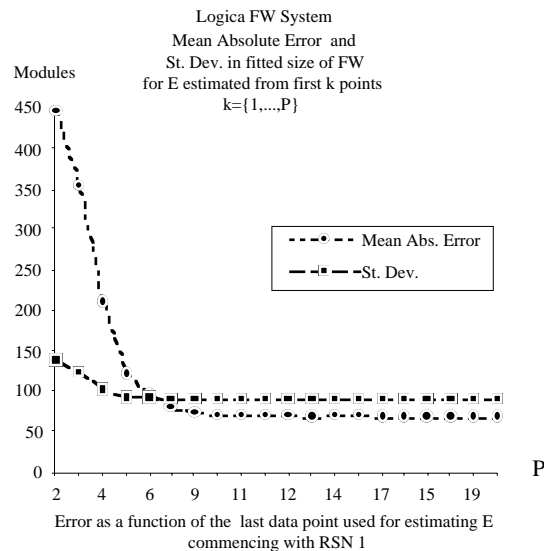


Fig. 3b 1990s System - Logica FW

## 5 Final Remarks

This position paper presents an introduction to an investigation that, if successful, should throw significant light on the software evolution phenomenon in general, the degree to which it depends on human direction and management and the role of organisational and system dynamics in driving and controlling the process. The project also represents the culmination of a twenty five year focus of interest which is expected to confirm conclusions reached so many years ago. Early results are most encouraging and confidence in the phenomenological interpretation is being strengthened. It is, however, far too early to reach firm conclusions. The intent of the present paper is to draw wider attention to the study. This will, it is hoped, encourage others to pursue related investigations so leading to development of a well founded metric and phenomenology based software process theory that can serve as a powerful tool in future software process improvement efforts.

## 6 Acknowledgements

Responsibility for claims made in this paper must rest entirely with the present author. The major roles played by Dr Dewayne Perry and Professors Vic Stenning and Wlad Turski (the FEAST core group) in developing and exploring the concepts and results presented here deserve full and grateful acknowledgement. More recently the two FEAST/1 Research Associates, Dr Paul Wernick and Juan Ramil-Fernandez have joined the team of investigators and have already made significant contributions. It is also appropriate to mention the many participants in the three International FEAST Workshops (1994/5) who in their critical appraisal and creative criticism of the work of the core group ensured that the current project could be funded, launched and move forward. Finally acknowledgement is due to the EPSRC for grants numbers GR/K86008 and GR/LO7437 that are supporting FEAST/1 and permitting the continuing involvement of Professor Turski And Dr Perry.

## 7 References - items marked with an "\*" are included in [leh85]

- [abd91] Abdel-Hamid T and Madnick S E, *Software Project Dynamics - An Integrated Approach*, Prentice Hall, Englewood Cliffs, NJ 07632, 263 p.
- [bel72] Belady LA and Lehman MM, *An Introduction to Program Growth Dynamics*, in *Statistical Computer Performance Evaluation*, W Freiburger (ed), Academic Press, New York, 1972, pp. 503 - 511
- [boe76] Boehm B W, *Software Engineering*, IEEE Trans. on Comp., v. C-5, n. 12, Dec. 1976, pp. 1226-1241
- [cho81] Chong Hok Yuen C K S, *Phenomenology of Program Maintenance and Evolution*, PhD Thesis, Dept. of Comp., Imp. Col., 1981
- [fea94,5] *Preprints of the FEAST Workshops*, MM Lehman (ed.), Dept. of Comp., ICSTM, 1994/5

- [law82] Lawrence M J, *An Examination of Evolution Dynamics*, Proc. Proc. 6th Int. Conf. on Softw. Eng., Tokyo, Japan, 13 - 16 Sep. 1982., IEEE Comp. Soc. ord. n. 422, IEEE cat. n. 81CH1795-4,
- [leh69]\* Lehman M M, *The Programming Process*, IBM Res. Rep. RC 2722, IBM Res. Centre, Yorktown Heights, NY 10594, Sept. 1969
- [leh74]\* *id.*, *Programs, Cities, Students, Limits to Growth?*, Inaugural Lecture, May 1974. Publ. in Imp. Col of Sc. Tech. Inaug.l Lect. Ser., vol 9, 1970, 1974, pp. 211 - 229. Also in *Programming Methodology*, (D Gries ed.), Springer, Verlag, 1978, pp. 42 - 62
- [leh78]\* *id.*, *Laws of Program Evolution - Rules and Tools for Programming Management*, Proc. Infotech State of the Art Conf., Why Software Projects Fail, - Apr. 1978, pp. 11/1 11/25
- [leh80a]\* *id.*, *On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle*, J. of Sys. and Software, v. 1, n. 3, 1980, pp. 213 - 221
- [leh80b]\* *id.*, *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Special Issue on Software Engineering, v. 68, n. 9, Sept. 1980, pp. 1060 - 1076
- [leh85] Lehman M M and Belady L A, *Program Evolution, - Processes of Software Change*, Academic Press, London, 1985, pp. 538
- [leh87] Lehman M M, *Process Models, Process Programs, Programming Support - Invited Response To A Keynote Address By Lee Osterweil*, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March 2 Apr. 1987, IEEE Comp. Soc. pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 14 - 16
- [leh91] *id.*, *Software Engineering, the Software Process and Their Support*, IEE Softw. Eng. J. Spec. Iss. on Software Environments and Factories, Sept. 1991, v. 6, n. 5, pp. 243 - 258
- [leh96a] Lehman M M and Stenning V, *FEAST/1: Case for Support*, ICSTM, March 1996
- [leh96b] Lehman M M, *Laws of Software Evolution Revisited*, Proc. EWSPT'96, Nancy, 9 - 11 Oct. 1996
- [leh96c] Lehman M M, *Process improvement - The Way Forward*, Invited Keynote Address, Proc. Brazilian Software Engineering Conference, 14 - 18 October 1996, pp. 23 - 35
- [mad96] Madachy R J, *System Dynamics Modelling of an Inspection Process*, 18th Int. Conf. on Softw. Eng., Berlin, 25 - 29 March. 1996, IEEE Comp. Soc. ord. n. PR07246, IEEE Cat. n. 96CB35918, pp. 376 - 386
- [oha96] O'Hare G and Jennings N, *Foundations of Distributed AI*, Wiley, 1996
- [ost87] Osterweil L, *Software Processes are Software Too*, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March - 2 Apr. 1987, IEEE Comp. Soc. Pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 2 - 13
- [roy70] Royce W W, *Managing the Development of Large Software Systems*, IEEE Wescon, August 1970, pp. 1 - 9
- [spw84] Potts C (ed), *Proceeding of the Software Process Workshop*, Egham, Surrey, UK, Feb. 1984. IEEE, cat. n. 84CH2044-6, Comp. Soc., Washington D.C., ord. no. 587, pp. 27 -35
- [tur96] W M Turski, *Reference Model for Smooth Growth of Software Systems*, IEEE Trans. on Soft. Eng. v.22, n. 8, August 1966

mml566[papers]  
3/2/97