On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution

M. M. Lehman, D. E. Perry and J. F. Ramil

Abstract— **As part of its study of the impact of feedback in the global software process on software product evolution, the FEAST/1 project has examined metric data relating to various systems in different application areas. High level similarities in the growth trends of the systems studied support the FEAST hypothesis.** *Inter alia*, **the results provide evidence compatible with the laws of software evolution, subject only to minor adjustments of the latter.**

Keywords— **Laws of software evolution, system dynamics, feedback, FEAST hypothesis,** *E*-**type systems**

1 Introduction

The FEAST/1 project [5] is examining metric and other records of *E*-type software systems addressing a variety of applications. Such systems model the behaviour of people and/or machines active in a real world domain. The system studied include ICL's VME operating system kernel, Logica's FW banking transaction system and a Lucent Technologies real time system. All these have been (and continue to be) evolved by their respective organisations. Their growth over more than 15 years, in sequences of between 17 and 29 releases, display short and long term trends that are similar to each other and also to those observed in the study of OS/360 and three other 1970s systems. It was study of the latter that led to the first five laws of software evolution [3]. On the basis of a recent formulation [6], [8] with some refinements, this paper restates the laws and summarises the support provided by analysis of the above data. Other refinements are likely to follow as future studies yield further insight [11].

2 Support for the Laws of Software Evolution

2-1 Continuing Change (I) and Growth (VI)

[I] E-type systems must be continually adapted else they become progressively less satisfactory.
[VI] The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.

The first and sixth laws reflect complementary phenomenological interpretations. They are treated together since in practice, it is difficult to isolate their individual contribution to system change and growth as a system evolves from release to release. Their separation requires detailed examination of change records (not always available) and interrogation of maintenance personnel. Fig. 1 shows the *long term* growth trends (system size in modules per *release sequence number* (RSN)) of the above mentioned systems. These growth plots illustrate the striking similarity referred to above, that is, close to linear long term growth with a superimposed ripple. The deviation from smooth growth is addressed in section 2-5.

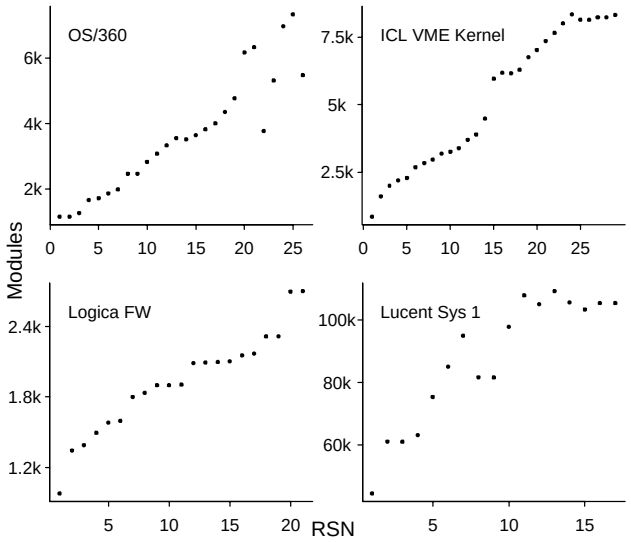

Fig. 1. Software system growth plots

We note that the first law discusses system behaviour and does not refer to changes in the operational domain. Such change is inevitable for *E*-type systems. It is driven in part by installation and operation of the system, in part by exogenous forces. This leads to a need for continuing system adaptation to *maintain* it satisfactory as a model of the application in its operational (user) domain and so to maintain system behaviour satisfactory to the user [3].

E-type applications in their operational domains are countably infinite in their attributes while the implementation is finite, that is bounded, valid only for a bounded domain. Properties excluded by the bounds eventually become a source of performance limitation, irritation and error. These must be eliminated by adding system capability and functionality to achieve a satisfactory system [4]. The sixth law follows.

M. M. Lehman and J. F. Ramil are with Imperial College of Science, Technology and Medicine, Dept. of Computing, London SW7 2BZ, E-mail: {mml,jcf1}@doc.ic.ac.uk, tel: +44 (0)171 594 8214, fax: +44 (0)171 594 8215, http://www-dse.doc.ic.ac.uk/~mml/

D. E. Perry is with Lucent Technologies, Bell Laboratories, Murray Hill, NJ 07974, E-mail: dep@research.bell-labs.com, tel: +1 908 582 2529, fax: +1 908 582 5809, http://www.bell-labs.com/user/dep/

Grateful thanks are due to our academic and industrial collaborators and to the UK EPSRC for their support.

2-2 *Increasing Complexity or Entropy (II)*

[II] As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.

The second law arises from both local and global phenomena. On the one hand, most local enhancements require, *inter alia*, the insertion of mechanisms such as calls to new procedures or the insertion of objects. This adds branch points to the system, increasing its complexity however defined. At the other extreme, functional extension such as the addition of a new device or a functional subsystem will increase functional complexity of the system as well as its global structural complexity. It will also increase complexity at lower levels because of calls, references, object and communication links required to achieve functional and operational integration with the rest of the system. As systems evolve complexity growth at many levels is inevitable. Awareness of this, including occasional maintenance activity specifically directed to complexity reduction, is needed if this is to be controlled. Locally it may be appropriate to recode elements destructured through the addition of change upon change upon change. Globally complexity reduction may be achievable by architectural restructuring, the creation of new subsystems and/or the reallocation of function amongst, modules or subsystems, for example.

In the absence of appropriate metrics consistently applicable across the systems studied, complexity change could not be directly measured. The law is, however, supported by the fact that the growth of the Logica [14], Lucent Technologies and ICL systems can be closely modelled by an inverse square relationship [14], [8]. Fig. 2 shows inverse square models (Appendix) of the three systems, along with a linear least squares growth model for OS/360. The factors associated with the discontinuity of the VME Kernel trend are still being investigated. The reasons for the different behaviour of OS/360 has not yet been determined.

The inverse square relationship yields precisely the growth trend to be expected if growth is constrained by growing complexity. So while not proving the law, this model is consistent with it. Having been observed in several instances adds to the confidence in its validity. It may be noted in passing that the inverse square relationship has also appeared in a *system dynamics* [2] model of the Matra-BAe process, another system studied [15] as part of the FEAST/1 project.

Confidence in the validity of the second law is thus strengthed by both rationalisation and empirical evidence. The former comes through recognition of intrinsic complexity growth that accompanies all software evolution as described above. The latter is provided by complexity related behavioural patterns that are modelled by inverse square relationships.

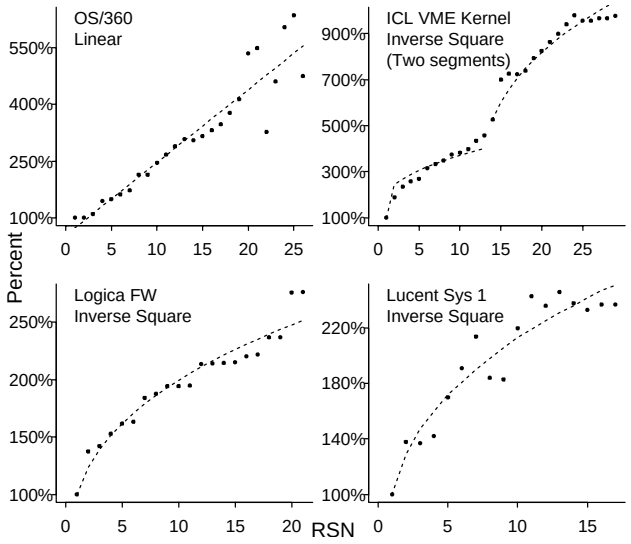

Fig. 2. Growth models estimated using all the data points (releases) available for each system.

2-3 *Self Regulation (III)*

[III] Global E-type system evolution processes are self regulating.

The phenomenon described by this law was first identified by the ripple in the growth trend plot of OS/360 as reproduced in fig. 1. Similar ripples have now been observed in the growth plots of systems studied since then (eg. [3], [8]) as illustrated in fig. 3 for three of the most recently analysed systems.

From the start [1] the ripple was interpreted as an indicator of the feedback like nature of the *E*- type software process. Its identification suggested that *global* *E*-type software processes display a strong, feedback generated, dynamics. In part, at least, this would be driven by feedback based checks and balances intended to drive the process to its goals. Feedback controls, positive and negative, have been recognised and referred to in [7], [9], [15]. Note that the term global process as used in this paper encompasses the activities, at many levels, of technical personnel, management, marketing and support personnel, users and others involved.

2-4 *Constant Work Rate or Conservation of Organisational Stability (IV)*

[IV] The average effective global activity rate in an evolving E-type system tends to remain constant over the product lifetime.

Project activity can be measured by work input or work output. The most obvious measure of input effort is *person time*. In the context of a study relying on historic databases, this is however, likely to prove unreliable, requiring for example manual scanning and detailed analysis of archived time records and results of questionable accuracy [13].

An alternative activity measure relates to work output. Because the latter reflects the impact and constraining influence of many more feedback loops than does work input, it appears to represent a better choice. An example generally derivable from basic data in a system's evolution data base is elements *handled*. This records the number of different modules, files or objects, etc. (henceforth elements) removed, modified or added in the preparation of each release. Lines of code are frequently used to measure activity because it is more readily available. The limitations of this metric can, however, not be discussed here [3].

Handle counts leave much to be desired since a handle may represent anything from a simple spelling correction to the re-engineering of an entire element. However, when one is observing and analysing evolution trends in *large* systems [3], the number of elements handled is sufficiently high to justify statistical treatment. One may expect that such variations will then have, at worst, a minor impact on long term trends.

The fourth law was originally deduced from the shape of the cumulative handles plot over releases of OS/360 evolution [3]. The data was so analysed to reduce the effect of release overlap, concurrent work on elements released at different times. Since questions relating to the validity of inferences from cumulative data have not yet been resolved, this result is not included here.

Finally, counts of *handlings* (that is, the sum over all elements of the number of different changes being implemented within a release) provide still another alternative indicator but has not yet been applied by us.

The extent and manner in which these various metrics permit behavioural inferences in general, and in the investigation of the fourth law in particular, is expected to be part of future investigations [11]. Given these uncertainties in the interpretation of these various activity measures the fourth law must be considered *sub judice* in the context of this paper.

2-5 Conservation of Familiarity (V)

[V] On average, the incremental growth tends to remain constant or to decline.

As an E-type system evolves all associated with it must maintain mastery of its content and behaviour to achieve satisfactory usage and evolution. Excessive growth diminishes that mastery and leads to a transient reduction in growth rate or even shrinkage. On the other hand, positive feedback from, for example, marketeers and users, leads to pressure for increasing growth rate. FEAST/1 observations (fig. 3) suggest that in the long term the net result is constant or declining average incremental growth. That is, the incremental growth above some threshold tends to be followed by a smaller one, reflecting a need for, for example, an increased bug fixing or restructuring content. If two or more increments exceeding some threshold are forced through the plots suggest that the next increment(s) are close to zero or even negative, possibly reflecting a system clean up. In extreme cases this can even lead to system fission, as exemplified by OS/360 releases following release 20. That is, incremental growth greater than some higher threshold (for example, mean growth plus twice the standard deviation ($m + 2s$)), leads to a serious decline in incremental growth.

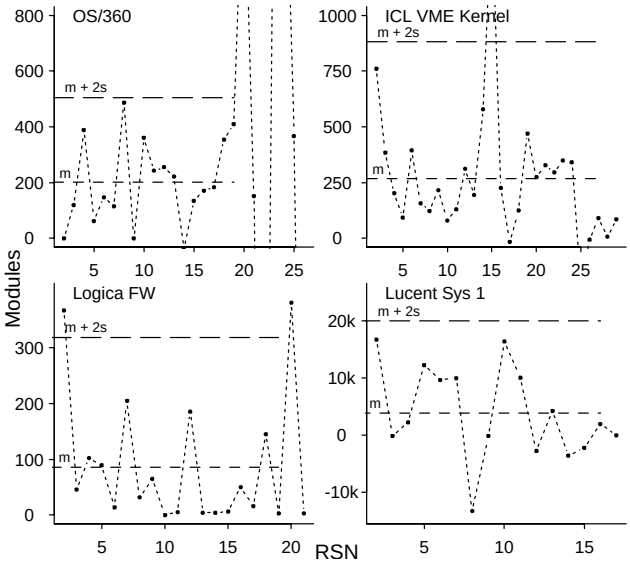

Fig. 3. Regulation in incremental growth (third law) and limit infringement (fifth law). m represents the mean incremental growth and s its standard deviation.

This phenomenon is interpreted as relating to peoples' information absorptive capacity. Individual ability to absorb and apply a number of items of loosely or unrelated information is widely believed to be limited [12]. Once the number is past some critical point the ability to digest and apply the information decreases very rapidly. The net result is likely to be, for example, poorer design, higher error rates, performance deterioration, slow down in work, increasing numbers of queries from the field, resistance to release adoption even declining sales. The consequences affect developers, their managers, marketeers and users. Precise determination requires further research [11].

The fifth law was originally inferred from the OS/360 plot of incremental growth per release (fig. 3). Essentially the same underlying phenomenon is now observed in systems being studied in FEAST/1 (fig. 3). Differences between the plots, however, suggest differences in detailed behaviour. These may be attributed, for example, to the properties of the individual pro-

cesses being executed, the organisations implementing the evolution and the application and usage domains being addressed. It is merely noted that the relevant behaviour that, when observed in the OS/360 plot gave rise to the fifth law, is observed in the evolutionary growth trends of the systems studied.

2-6 Declining Quality (VII)

[VII] The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.

This law follows from the *principle of uncertainty* and the characteristics of *E*-type systems [4]. As already mentioned in section 2-1, an *E*-type application and its domain are at least countably infinite in their attributes. The software that implements the model is certainly finite. Thus the system as a model is incomplete and imprecise. The gap between the system and the application in its operational domain is bridged by assumptions. But the application and its domain are dynamic, always changing, and such change is accelerated by the installation and use of the system. Unless detected and fixed, such changes are likely to progressively invalidate assumptions embedded in the system; the system becomes polluted with a growing number of invalid assumptions. The effect of encountering the embodiment of an invalid assumption during execution is not known. The principle of uncertainty and the seventh law follow.

2-7 Feedback System (VIII)

[VIII] E-type evolution processes constitute multi level, multi loop, multi agent feedback systems and must be treated as such to achieve significant improvement for other than the most primitive processes.

The observation that the software process is a feedback system that develops a dynamics of its own and that attempts to improve the process must take its feedback nature into account dates from the 70s [3]. The eighth law is essentially equivalent to the FEAST hypothesis [5] and there is now enough confidence to justify its expression as a law. Note that the process referred to here is the global process (Sect. 2-3). All the levels of activity referred to there are contained within, driven and constrained by the feedback loops, mechanisms and controls that evolve as an organisation (and its activities) ages [3]. The process is a multi level, multi loop, multi agent feedback system. Note that the eighth law suffices to explain the behaviour encapsulated in the other seven. Thus, the statements of the latter apply only as long as there is no change in relevant global-process feedback mechanisms.

Because of space limitations further discussion of the eight law is limited to one piece of supporting evidence and the implication that feedback induced dynamics influences software system evolution trends. Additional discussion based on system dynamics models including also examples of feedback mechanisms in the global software process, is provided elsewhere [9], [15].

In Sect. 2-2 an inverse square model of growth was considered for three of the four systems referred to in this paper. A linear growth model was used in the case of the fourth system (OS/360). In estimating these models (Fig. 2), all available data points were used for each system. One may, however, ask (as Turski did [14]) how many points are necessary to obtain an acceptable predictor for future growth? That is, the model may be based only on the first i points of each data set. The *mean absolute percentage error* MAPE (Appendix) may then be used to assess the forecasting ability of the resultant models (Fig. 4).

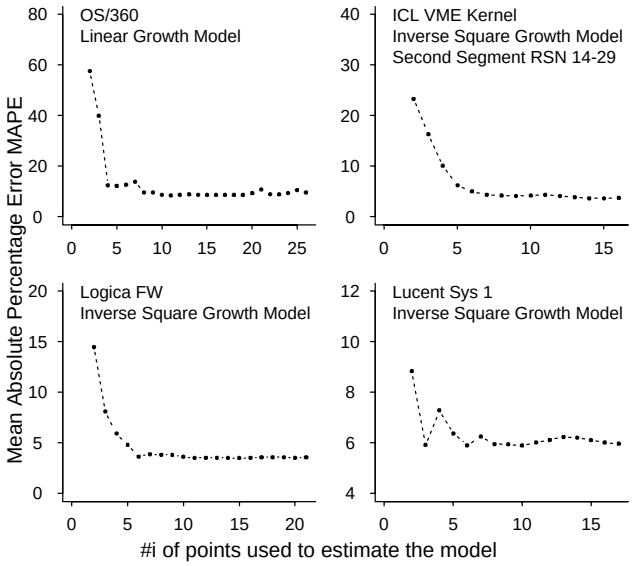

Fig. 4. Mean Absolute Percentage Error (MAPE) of fit as a function of the number of points used to estimate the growth model. MAPE was calculated over all releases. Note the quick settling of MAPE to a value less than 10% after only a few points are used to estimate the model.

The significance of the trends shown in these plots is summarised in Table 1. The result is, indeed, astounding. The plots and Table 1 indicate that no more than a handful of releases are required to obtain a good predictor. This suggests to us that the dynamics rather than management determines the medium and long term growth trend. The dynamics is related, one way or another, to the characteristics of the evolving software system, its development and application domain and to the processes by which it is evolved.

Two *caveats* must be noted in regard to the above interpretation. In the first instance, the initial decline in MAPE over the first 3-5 points for each of the systems plotted in fig. 4 must reflect, at least in part, the process of model parameter convergence to a steady value [10]. Secondly, few projects begin from scratch.

TABLE 1

AVERAGE AND STANDARD DEVIATION OF MAPE AFTER THE STABILISATION POINT

System	Stabilisation Point	Avg. MAPE	St.Dev. of MAPE
OS/360	4	9.7	1.6
VME Kernel	6	4.1	0.4
FW	6	3.6	0.1
Lucent Sys 1	3	6.2	0.4

For any other than a new organisation, organisational dynamics and the corporate practice in the software process will have been long established, informally or formally. Many of the management and implementation processes and controls, and therefore their dynamics, will have been inherited from past projects and practice. Such inheritance is likely to constitute a major, possibly dominant, component of the project dynamics. This is because the outermost loops of a multi level multi loop feedback system are likely to have the greatest impact on externally visible system (process) attributes. The separation of the influences of the various sources of dynamics on the observed behaviour, whether indeed they can be distinguished is, however, an open question [11]. It is, however, precisely this issue that underlies our continued emphasis on the global software process.

3 Final Remark

The metrics based evidence provided in this paper increases our confidence in six of the eight laws. Confidence in the seventh (ie. law VII) is based on theoretical analysis. Law IV is neither supported nor negated. This conclusion is summarised in Table 2.

TABLE 2

SUPPORT FOR THE LAWS OF SOFTWARE EVOLUTION

Law	I	II	III	IV	V	VI	VII	VIII
	√	√	√	?	√	√	√	√

Moreover, the available evidence indicates that dynamics arising from the feedback nature of the software process is a powerful force in determining its evolutionary characteristics and behaviour. Of particular interest to the present symposium is the fact that, in the main, the conclusions have been derived from measured growth trends of systems currently being evolved in industry. More general application of the approach described should facilitate major process improvement based on measurement and modelling of the global software process and interpretation of the results. These may then be applied to process improvement through *adjustment* and *tuning* of organisational and process feedback mechanisms.

Appendix

Inverse Square Model [14], [8]: S_i, the size predicted by the model at release i, has been calculated from $S_i = S_{i-1} + \hat{E}/S_{i-1}^2$ $(2 \leq i \leq n)$ where n is the total number of releases. S_1 is assigned the actual size of release 1. More generally $\hat{E}$ may be defined as the average E_i, where $E_i = y_{i-1}^2(y_i - y_{i-1})$ $(2 \leq i \leq j)$, y_i is the actual size of release i and j is the number of releases used to estimate the model.

Mean Absolute Percentage Error MAPE: For a growth model based on the first j releases only, MAPE_j, is calculated as $\mathrm{MAPE}_j = \frac{100}{n}\sum_{i=1}^{n}\frac{\|S_{i,j}-y_i\|}{y_i}$, where $S_{i,j}$ is the predicted size at release i using a model based on the first j releases only, y_i is the actual size at release i and n is the total number of releases.

References

[1] Belady LA and Lehman MM, *An Introduction to Growth Dynamics,* Proc. Conf. on Stat. Comp. Perf. Eval., Brown University 1971, Academic Press, 1972, W Freiberger (ed.), pp. 503 – 511

[2] Forrester JW, *Industrial Dynamics,* Productivity Press, Cambridge MA, 1961

[3] Lehman MM and Belady LA, *Program Evolution - Processes of Software Change,* Academic Press, London, 1985

[4] Lehman MM, *Uncertainty in Computer Application and its Control Through the Engineering of Software,* J. of Softw. Maint.: Res. and Pract., v.1, n.1, Sept. 1989, pp. 3 – 27

[5] Lehman MM and Stenning V, *FEAST/1: Case for Support,* Imperial College, DoC, March 1996

[6] *i.d., Laws of Software Evolution Revisited,* Proc. EWSPT 96, Nancy, 9 - 11 Oct. 1996, LNCS 1149, Springer Verlag, 1997, pp. 108 – 124

[7] 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

[8] Lehman M M *et al, Metrics and Laws of Software Evolution - The Nineties View,* Proc. Fourth Int. Soft. Metrics Symp., Metrics 97, Nov. 5-7, Albuquerque, NM, 1997, pp. 20 – 32

[9] Lehman MM and Wernick PD, *System Dynamics Models of Software Evolution Processes,* Proc. Int. Wrkshp. on the Principles of Software Evolution, ICSE '98, Kyoto, Japan, April 20 -21, 1998, pp. 6 – 10

[10] Lehman MM and Ramil JF, *The Impact of Feedback in the Global Software Process,* Keynote Lec., Workshop on Software Process Simulation and Modelling, ProSim'98, June 22-24, Silver Falls, OR 1998

[11] Lehman MM, *FEAST/2: Case for Support,* Imperial College, DoC, July 1998

[12] Miller GA, *The Magic Number 7, Plus or Minus 2: Some Limits on Our Capacity for Processing Information,* Psychological Review, 63, 1956

[13] Perry DE, Staudenmayer NA and Votta LG, *People, Organisations and Process Improvement,* IEEE Software, July 1994, pp. 36 - 45

[14] Turski WM, *Reference Model for Smooth Growth of Software Systems,* IEEE Trans. on Softw. Eng., v.22, n.8, Aug. 1996, pp. 599–600

[15] Wernick PD and Lehman MM, *Software Process White Box Modelling for FEAST/1,* Workshop on Softw. Process Simulation and Modelling, ProSim'98, June 22-24, Silver Falls, OR, 1998, to appear in J. Syst. and Softw. 1999

Some of the material listed above is available via:
`http://www-dse.doc.ic.ac.uk/~mml/feast1`