Implications of Evolution Metrics on Software Maintenance

M M Lehman
Department of Computing
Imperial College
London SW7 2BZ
tel: +44 (0)171 594 8214
mml@doc.ic.ac.uk

D E Perry
Bell Laboratories,
Murray Hill, NJ 07974
tel:+1 908 582 2529
dep@research.bell-labs.com

J F Ramil
Department of Computing
Imperial College
London SW7 2BZ
tel: +44 (0)171 594 8216
jcf1@doc.ic.ac.uk

ABSTRACT
The FEAST/1 project is studying the impact of *feedback* on *E*-type software evolution, and a hypothesis which attributes the failure to achieve major software process improvement, in part, to overlooking its role. Amongst its activities the FEAST/1 project [leh97] is studying metrics of the evolution of several industrial software systems, ranging from a financial transaction system to a very large real time system. When comparing evolution metrics from so widely different systems, similarities emerge which support conclusions reached in a 1970s study of OS/360 evolution, enabling their further refinement and suggesting that, both metrics and conclusions derived from them are relevant and should be taken into consideration for successful software maintenance.

Keywords
Software:- maintenance, evolution, metrics, dynamics, feedback, improvement; Lehman's laws

1 Feedback in the Software Process

Some years ago one of the present authors wondered why major sustained improvement of the industrial software process is so difficult to attain. No sooner asked then an answer came to mind. He recalled 1970's work identifying the software process for *E*-type systems as a multi-level multi-loop feedback system with, in general, both positive and negative feedback mechanisms [leh85-cs5,12][1]. Now positive feedback tends to cause growth and process speed up. Negative feedback, on the other hand, tends to stabilise a system or process, constraining the external impact of changes to forward path elements. The software process itself can be visualised as encompassing a major feedback loop which starts off in *ab initio* development with a non trivial application concept and ends up at the first and subsequent iterations with a version of an operational system. A gap (partly due to invalidation of assumptions and partly due to learning and increasing ambition, market competition, etc) appears between the functionality of each version of the operational system (when eventually delivered) and the application concept [leh98b]. Filling this gap through change and enhancement represents a source of unending software maintenance, that is the sucessive iterations at a given rate on the software process major feedback loop [leh98b].

Software process improvement has always focussed on forward path mechanisms: the introduction of new process steps, better languages, improved methods and tools and so on. Such changes will be contained within feedback loops, such as those monitoring and controlling product quality, resource usage and the rate of progress. Other feedback mechanisms might direct projects and processes towards organisational goals or provide the checks and balances required to ensure a successful organisation.

As a software organisation evolves, feedback controls will be set up to ensure, for example, desire quality levels, timely delivery, organisational stability and progress to global goals. The external impact of changes to forward path mechanisms within negative feedback loops will then be constrained and yield less benefit than might be expected on the basis of local assessment. Thus, changes to feedback mechanisms, such as the use of inspections, rapid prototyping or evolutionary development for example, should be important ingredients of process improvement.

2 The FEAST Hypothesis

The FEAST hypothesis (*F*eedback, *E*volution *A*nd *S*oftware *T*echnology) first formulated in 1994 [leh94], encapsulates these issues. It may be stated as:

As complex feedback systems, E-type software processes evolve strong system dynamics and a tendency towards global stability as exhibited by feedback systems

Sub hypotheses include:

- *I* *Software evolution process for E-type systems constitutes a complex feedback system*
- *II* *Such feedback is likely to constrain benefits derived from forward path changes,*
- *III* *Major process improvement requires attention to feedback mechanisms and process dynamics*

3 The FEAST/1 Project

The hypothesis was initially explored by a *core group*[2] and discussed in a series of workshops [leh96a,fea84/5]. The

[1] To conserve space all papers included in [leh85] are referenced by their chapter number in that book

[2] Professors MM Lehman, V Stenning (Imperial College) and WM Turski (Warsaw U,) and Dr DE Perry (Bell Labs-Lucent)

FEAST/1 research project, initiated in 1996 [leh96b] has been seeking tangible supporting evidence from data on active industrial evolution processes. *Black box* (metric) analysis is being undertaken and *white box* (system dynamics) models are being developed and analysed. Findings have been reported [leh96,97,98a,b,wer98]

4 Laws of Software Evolution

Recognition of the software evolution phenomenon and its investigation arose initially from a study of the IBM programming process [leh85-c3]. Various attributes of successive releases of OS/360, as exemplified by fig. 1, were examined. The figure shows that system growth over 26 releases (identified by a *sequence number*, rsn,) follows a smooth long range trend, with a superimposed ripple and a region of instability or chaotic behaviour. Taken together, these phenomena suggested the feedback-system-like properties mentioned above.

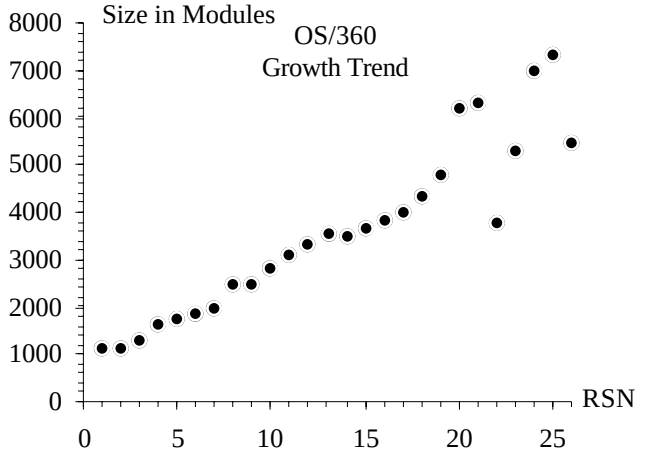

Fig. 1 IBM OS/360 growth trend

The essence of the OS/360 observations and interpretation were captured in eight Laws of Software Evolution [leh85-cs7,12,19, 96c] (see table A.1) - laws because they related to social, managerial and organisational, not technical, phenomena. Some, however, considered the data source tainted reflecting 1960/70s technology applied in a non typical (ie. IBM) development environment to the evolution of a, for its time, non typical system. The observed behaviour and laws derived from it were, therefore, unlikely to be widely applicable [law82]. But the laws were supported by subsequent work [kit82,leh85-c12,96d] and by the metrics gathered under the FEAST/1 project, as exemplified in fig. 2.

Figs. 1 and 2 show the growth trend of the earliest and most recently received data respectively. Despite a two decade interval between their creation and an order of magnitude difference in size there is sufficient similarity between these growth patterns to suggest an underlying common phenomenon. Similar pattern were observed in four other systems for which data was obtained in the 1970s [leh77,8, leh85, ch. 12] and for three now being studied. These systems address different application areas, cover a wide range of sizes, were/are being evolved in quite different development domains and are used in widely different contexts. The observed similarities are, therefore, unlikely to be due to chance. Their common patterns strengthens confidence in the validity of the original conclusions.

In observationals studies such as the present one, confidence may be increased in at least two ways. Observation of similarity in the behaviour of some evolution parameter across a number systems, increases confidence in laws that encapsulate the common behaviour. The more different the systems, the development or the application domains the greater the impact. Concentration on data sets of an individual system and comparison of its behaviour with that predicted by the laws is equally confidence building. Success with either approach strengthens confidence; failure leads to modification or rejection. Both are necessary to increase understanding and control of software evolution.

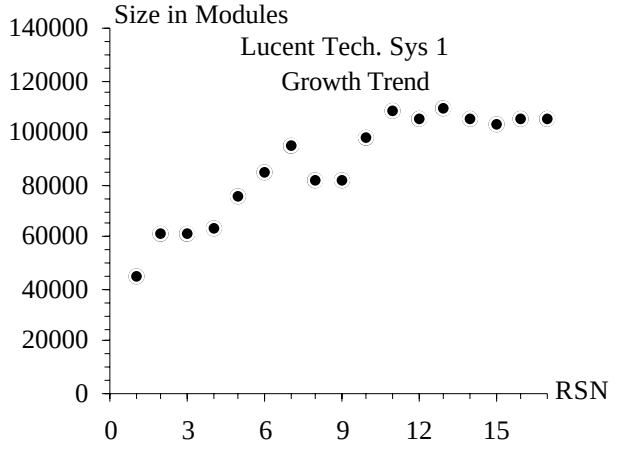

Fig. 2 Lucent Technologies system growth trend

The Logica FW financial transaction system study [tur96, leh96d,97a,b] exemplifies the observational case study approach. It suggests that FW evolution is largely consistent with at least five of the eight laws. The other three are not contradicted. There is simply no evidence[leh96c]. Here too a consistent picture is emerging.

Addressing the investigation of system evolution by pure experimentation has not been totally discarded, although not found feasible yet. Calibrated system dynamics models may offer an alternative to experimentation [leh98a,wer98].

4 Metrics of Software Evolution

A list of metrics and indicators currently being considered in FEAST/1 is shown in table 1. This list is in no way complete. For example, it does not include *derived* indicators obtained either as a function of one or more indicators. Alternative surrogate measures which convey the underlying attributes (size, activity, effort, etc) in the evolution of a particular *E*-type program may be defined. An investigator interested in the software evolution process tries to get a picture of several years of *E*-type system history from records and data stored in historic data bases, configuration management systems or similar support tools which were defined for other purposes [mcg79, roc75,

tus87]. One must live and work with the available measures and apply an analysis procedure that can be generalised across systems. This apparently lack of rigour, when compared with grass root software metrics programs is a fact of life in software evolution studies and it is one of the reasons to consider the relationship between the actual metrics and the attributes as fuzzy. It may also explain why an appropriate way of encapsulating findings is in linguistic statements such as those of the laws. Where more than one measure is available for the same attribute on a specific system one may wish to determine whether, at some level of detail both 'tell the same story'.

Release sequence number - rsn
Size of system - subsystems, modules, files, etc.
Elements handled - subsystems, modules, files, etc.
Elements added - subsystems, modules, files, etc.
Elements changed - subsystems, modules, files, etc.
Elements deleted - subsystems, modules, files, etc.
Element handlings - subsystems, modules, files, etc.
Release interval or general availability (G.A.) date
Effort applied - in appropriate units
Errors detected before and after G.A.- per release
Errors fixed before and after G.A.- per release

Table 1 FEAST/1 Metrics of Software Evolution[3]

Measures based on rsn as a pseudo-time indicator have proven consistently useful in both the early and current software evolution studies [leh85,97]. The age, absolute or relative, of a system at the time of release is, however, sometimes more appropriate, in particular when drastic changes in release intervals have ocurred. As previously discussed[leh85,97b], *module counts* (or related measures eg, files, etc.) offer a number of advantages (in the context of software evolution studies) over other measures, such as function points and, in particular that they are always available in historic data bases and so provide a potential for cross system comparison [leh97b].

The metric *elements* or *modules handled*, may be taken as *a* measure of the work done in developing release "i". It is widely defined as the sum of the number of elements or modules created and those changed in the course of that work. Changed modules are those differing in, at least, one non-comment line to the equivalent module in the previous version. Deletions may or may not be included. Issues that arise from the use of modules handled include the definition for *modules* used by each development environments, the stability of those definitions over time, the spread in the extent of the changes to individual modules and the variability of the portion of the system being handled at each release. The number or fraction of modules being deleted may also be significant at some point in the system's history.

An complementary measure is described by the term *handlings*. This measure differs from *handles* in that a module to which "n" (independent) changes are made is counted n times to determine the count. A module change is completed (an a handling counted) when an independent task such as a bug fix or the implementation of an enhancement has been performed. Handlings is not, however, a widely available count.

A top-down approach for metrics definition have been followed in the FEAST/1 studies. For instance, the size of a system may be split when needed or justified in size of certain sub-systems, or size of certain type of code (eg automatically generated modules from those which require intellectual effort). As a general guideline when starting a metrics program it is recommended to keep the number of different metrics to a minimum, so that it is possible to characterise the evolution of a given system. In general, metrics such as those which are dependent on a certain process definition should be avoided. They will only convey some meaning during the period under which a particular process model was being enacted.

Various problems with the metrics are: changes in technologies (language) or practices which may change the interpretation of the metrics, parallel work which may affect consistency and applicability of the *release work* concept, and drastic changes in release interval.

In addition to the source code (and the process of evolving the source code) other models of the application could also be measured and analysed. Metrics could be applied *inter alia* to the specification documents, requirements, high level designs and low level designs. Size could be measured in number of paragraphs, number of atomic sentences, number of components or statements, etc. Management units such as change documents or modification requests could also be counted and accumulated by release under the assumption that given a larger number, individual units can be considered homogeneous.

Because of the declining costs of storage, it appears to be more appropriate to establish an electronic archive with all the generated releases of the source code and related application models with metric purposes in mind, than storing summarised data only. Summarisation implies assumptions which may become invalid or forgotten. A case study of software evolution may begin with a set of metrics which later on may need to be redefined or refined. Access to raw data guarantees that new more appropriate metrics could be always defined and extrated.

5 Results to date

Paucity in the number of releases (in all of the cases less than 30) have hindered the application of standard statistical time series techniques to the FEAST/1 data. Nevertheless, analysis techniques which have been applied to FEAST/1 data are of the simplest and have yielded some results. They include visual inspection of the plots of data as a function of rsn or of the age of system (whatever more

[3] G.A.: general availability

appropriate) to assess general trends, to identify maxima and minima (these provide an idea of process capacity). General trends have been obtained by performing linear least squares fits and by assessing the sign of the resulting slopes.

Recent publications have discussed the black box results obtained to date [tur96, leh97b,leh98b]. Similarities can be perceived from the evolutionary behaviour and trends of the systems currently being studied. Results in connection with the laws are summarised in table A.1. Next section shows an example of the analysis focused on only some aspects of system evolution.

5.1 Trends of Total Size and Portion not Touched

Fig. 3-6 show both, the total size of a system in modules and the part of the system not touched at each release. Both measures are expressed as a percentage of the largest size achieved by the system. Linear least square models have been fitted to highlight the dominant trend in each series.

All of the plots support the sixth law (continuing growth). The plots also provide, in one way or another, support for the first, second, third, fourth and eighth law. The interpretation is analogous to the already published for Logica FW and the interested reader is referred to [leh97b].

When comparing the plots, OS/360 shows significantly different slopes for the linear models fitted to the total size (3.03), on one side, and to the part of the system not touched (1.13), on the other. That is, in OS/360 the portion of the system being touched (ie added or modified) at each release increased, ending up with a period of instability from rsn 21 onwards. On the contrary, in the other three systems (which display a stable evolution) the linear models (major component of each trend) display approximately *parallel* lines, indicating, on average, that a constant portion of the system being manipulated at each release cycle. This finding is consistent with early conclusions and with the hypothesis that an evolving system develops an inherent system dynamics and invariances. This happens in a way which is independent from the day to day managers' and developers' decisions. Again, the similarities between these parallel linear models in three different systems suggest an underlying common phenomenon.

5.2 Trends of the Incremental Growth

A linear model fitted to the incremental growth [leh85] over releases displays a positive slope only in case of OS/360 and negative in the three other cases (plots have been omitted). This suggest that on average, a positive but decreasing incremental growth can be the norm and not a constant incremental growth as originally suggested [leh85]. A positive and increasing incremental growth may, therefore, be discarded.

5.3 Trends of Changed and Added Modules

Initial findings seem to indicate that at some point in the lifecycle the number of changed modules per release is

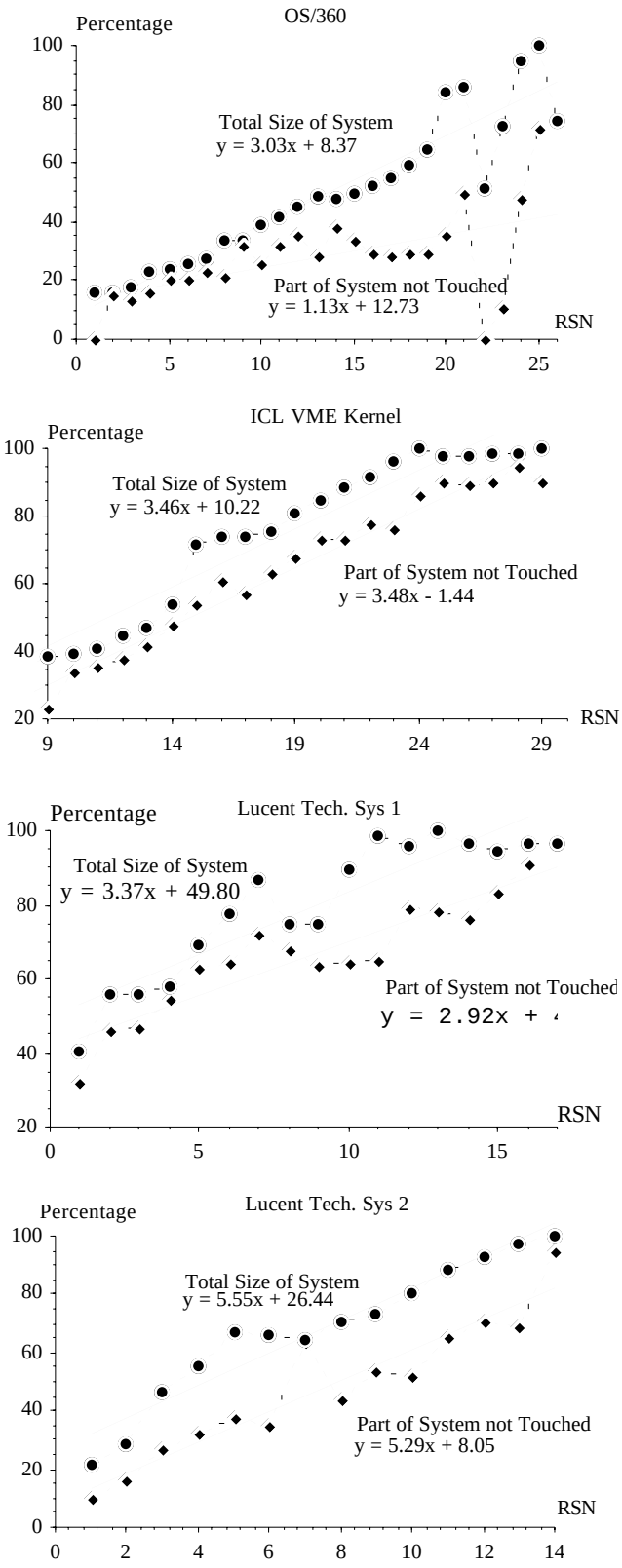

Fig. 3,4,5 and 6 Trends of the total system size and the portion of system not touched

greater than the number of added modules. Moreover the slope of a linear model fitted to the number of added modules over releases is negative or close to zero for the four systems mentioned above. On the other hand, a linear model fitted to changed modules over release show slopes which are positive in some cases and negative in others. In all the cases the slope of changed modules over releases is greater than the corresponding slope for added modules. This regularity across systems await appropriate interpretation.

6 Implications on Software Maintenance

It has been shown how simple analysis techniques (linear models) applied to a relative small number of metrics can yield insight and understanding intra and across different projects and characterise aspects of the software evolution and its stability. Collection and analysis of metrics of software evolution offer a number of other advantages. They provise a base for the building forecasting models. They facilitate the estimation of the process capacity (estimated from historic maxima and minima) which suggest a safety margin for smooth evolution. They provide a means to assess effects of process improvements policies on global long term behaviour. (FEAST hypothesis suggests that local process improvement will not alter global process characteristics). Moreover, the suggested metrics offer means for the calibration of system dynamic models for policy evaluation [leh98a, wer98].

The way forward requires more data on more systems in a variety of application and implementation domains.. As already observed, results to date have led to further support but also to restatement of the laws. As more data becomes available, further revision and more precise formulation may be necessary.

This study has complementary aspects. On the one hand, it provides a conceptual framework for and elements of a theory for the software process and its improvement. It demonstrates that software process modelling must more extensively and consistently reflect the feedback nature of both the software process in its technical aspects and the global business process, reflecting the impact of feedback control. But the metrics are also of practical importance. They provide a systematic basis for developing and controlling a systematic system evolution strategy and tactic and in identifying tools to support this process.

7 Final Remarks

Following the original software process study [leh85, ch.2] and its follow up [leh85, ch. 5] the study of software evolution was advanced during the 1970s by analysis of the behavioural patterns and trends of system and process parameters across a range of systems and organisations [leh85]. This led to formulation of the laws of software evolution (see appendix), to the development of a degree of confidence in their validity and, more recently, to the FEAST hypothesis and the FEAST/1 project. Initial findings derived from data on the Logica FW system, have reproduced and extended the earlier conclusions [leh97b]. The present paper provides orthogonal confirmation by demonstrating the common trend of one parameter across a several systems. Formulation in 1996 of the FEAST hypothesis and the eighth law that identifies the *E*-type process as a feedback system, as observed but not appreciated in 1972, the observations to date and the laws that result are seen a consequence of that reality.

Over the next months additional data is to be obtained and the modelling and analysis subsequently refined. It is expected that, by and large, this will confirm present conclusions and permit their extension. In addition the white box systems dynamics analysis of a number of systems [leh98, wer98] is enabling the identification of significant feedback loops and control mechanisms, essential for further successful progress of the FEAST/1 study. If these studies are more widely taken up in the software engineering community, and the process community in particular, and if aspects of the study that require multi disciplinary investigation receive the necessary attention the result will be an increasingly extensive and integrated basis and framework for software process engineering with practical methods and tools to support and improve it.

Acknowledgements

The authors are grateful to Brian Chatters (ICL) and Harvey Siy (Lucent Technologies) for providing the data on which this paper is based. Sincere appreciation is due to Profs. W M Turski (Visiting Fellow), to Dr. P D Wernick for their continuing contributions to this investigation. Since Oct. 1996 this work has been supported under EPSRC grants GR/K86008 and GR/L07437.

REFERENCES

[fea94/5] Preprints of the three FEAST Workshops, MM Lehman (ed.), Dept. of Comp., ICSTM, 1994/5, (http://www-dse.doc.ic.ac.uk/~mml/feast1/papers.html)

[kit82] Kitchenham B, *System Evolution Dynamics of VME/B*, ICL Tech. J., May 1982, pp.42 - 57

[law82] Lawrence MJ, *An Examination of Evolution Dynamics*, Proc. ICSE 6, Tokyo, 13 - 16 Sep 1982, IEEE Cat No. 82CH1795-4, pp. 188 - 196

[leh77] Lehman MM and Patterson J, *Preliminary CCSS System Analysis Using the Techniques of Evolution Dynamics*, Working Papers, Software Life Cycle Management Workshop, Airlie, VA, 1977, Publ. by ISRAD/AIRMICS, Comp. Sys. Com., Fort Belvoir, VA, Dec. 1977, pp. 324 - 332

[leh78] Lehman MM, *Laws of Program Evolution - Rules and Tools of Programming Management*, Proc. Infotech State of the Art Conf., "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, London, 1985, pp. 538

[leh96a] 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

[leh96b] Lehman MM and Stenning V, *FEAST/1: Case for Support*, ICSTM, March 1996

[leh96c] Lehman MM, *Laws of Software Evolution Revisited*, Proc. EWSPT'96, Nancy, 9 - 11 Oct. 1996

[leh96d]* *Metrics and Laws of Software Evolution - The Nineties View*, Metrics '97 Symp., Albuquerque, New Mexico, 5 - 7 Nov. 1997,. Also as *Process Improvement - The Way Forward*, in *Elements of Software Process Assessment and Improvement*, IEE CS Press, 1998

[leh97a] *id., Process Models - Where Next?*, "Most Influential Paper of ICSE 9" award, Proc ICSE 19, Boston MA, 17 - 23 May 1997, IEEE Cat No. 97CB36094, pp. 549-552

[leh97b] Lehman MM, Perry DE, Ramil JF, Turski WM and Wernick PD, *Metrics and Laws of Software Evolution-The Nineties View*, to be publ. in Proc. Metrics 97 Symp., Nov. 5-7th, 1997, Albuquerque, NM

[leh98a] 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

[leh98b] Lehman MM, *Feedback, Evolution and Software Technology - The Human Dimension*, Proc. Wrkshp. on Human Dimension in Successful Software Development, ICSE'98, Kyoto, Japan, April 20-21, 1998

[leh98c] Lehman MM and Ramil JF, *Implications of Laws of Software Evolution on Continuing Successful Use of COTS Software* , submitted to ICSM'98.

[mcg79] McGuffin RW, Elliston AE, Tranter BR and Westmacott PN, *CADES - Software Engineering in Practice*, Proc. ICSE 4, IEEE Cat No. 79CH1479-5C, Munich, Sep 17-19, pp. 136-144

[pau95] Paulk MC, *The Evolution of the SEI's Capability Maturity Model for Software*, Soft. Proc. Improv. and Practice, Pilot Iss., Aug. 1995, pp. 3 - 15

[roc75] Rochkind MJ, *The Source Code Control System*, IEEE Trans. on Soft. Eng. SE 1 - 4, Dec. 1975, pp. 364 - 370

[wer98] Wernick PD and Lehman MM, *Software Process Dynamic Modelling for FEAST/1*, ProSim'98, Proc. Int. Wrkshp. on Softw. Proc. Simulation Modelling, June 22-24, 1998, Silver Falls, OR [tur96] Turski WM, *Reference Model for Smooth Growth of Software Systems*, IEEE Trans. on Soft. Eng. v.22, n. 8, August 1996

[tus87] Tuscany PA, *Software Development Environment for Large Switching Projects*, Proc. of Softw. Eng. for Telecomms. Switching Sys Conf, 1987

Appendix

The initial FW analysis [leh97b], more recent work on data relating to both FW and other systems and the development of insight and understanding has shown that first statements of the laws [leh78,85, ch. 7] needed generalisation to cover behavioural variations amongst the systems and processes studied and modification to address the new insight gained.

The table below summarises the understanding of the salient aspects of the laws as of April 1998. Earlier versions of these statements will be found in the collection of papers brought together in [leh85]. A more recent discussion that includes both the original and revised statements of the laws and where source references are provided is also available [leh96c,97b]. Future work must be expected to lead to additional refinement, qualification or other modification of the laws. In particular, as the precise relationships between the first seven laws and the eighth become better understood the laws may be expected to become more specific, perhaps even formalised.

No. and Reference	Brief Name and Year of First Publication	Supported by FEAST/1 metrics	Current Formulation of Law
I [leh85, ch. 7]*	Continuing Change 1974*	√	*E*-type systems must be continually adapted else they become progressively less satisfactory.
II [leh85, ch. 7]*	Increasing Complexity 1974*	?	As an *E*-type system evolves its complexity increases unless work is done to maintain or reduce it.
III [leh85, ch. 7]*	Self Regulation 1974*	√	Except for the most primitive processes *E*-type system evolution processes are self regulating.
IV [leh85, ch. 12]	Conservation of Organisational Stability 1978	√	The average effective global activity rate in an evolving *E*-type system is invariant over product lifetime.
V [leh85, ch. 12]	Conservation of Familiarity 1978	√	The average incremental growth of a satisfactory evolving *E*-type system declines * steadily* as the system evolves.
VI [leh85, ch. 18]	Continuing Growth 1978	√	The functional content of *E*-type systems must be continually increased to maintain user satisfaction over their lifetime.
VII [leh96c]	Declining Quality 1994	?	The quality of *E*-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.
VIII [leh96c]	Feedback System first implied 1972 1996	√	*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.

Table A.1 Laws of software evolution restated