Metrics and Laws of Software Evolution

M M Lehman
J F Ramil
P D Wernick
Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ
tel: +44 (0)171 594 8214
fax: +44 (0)171 594 8215
e-mail: mml@doc.ic.ac.uk

Abstract

The process by which real world, *E*-type, software is developed and evolved to remain satisfactory in a changing operational domain has, over the years, proven most difficult to improve. It is suggested that this may be due to the very nature of the global process, which is a complex, multi-variable, multi-loop, multi-level, feedback controlled system involving humans. This observation was first recorded in the early 70s during an extended study of the evolution of OS/360; a study that ultimately led to Lehman's laws of software evolution. Following the realisation that this observation is relevant to the current interest in software process improvement, it has been formalised in a FEAST hypothesis. If the hypothesis represents a widespread phenomenon, important research issues are raised. Their resolution could lead to significant technological advances.

An on-going project, FEAST/1, based on the observation is briefly described. Preliminary conclusions from an, as yet incomplete, study of the Logica plc Fastwire (FW) financial transaction system are outlined and compared with those reached during the earlier OS/360 study. The new analysis supports or, better does not contradict, the laws of software evolution as originally formulated or the more recent FEAST hypothesis, strengthening confidence in their validity. Analysis of the data provides, for example, indications that after some six releases the FW growth trend is largely determined by the dynamics of the process. More generally, the results so far obtained suggest that the 1970s approach to metric analysis of software evolution is still relevant in the 1990s. It is hoped that the two-year FEAST/1 study will provide a solid theoretical and methodological foundation for mastering the feedback aspects of the software evolution process. Continued multi-disciplinary studies and, for example, development of feedback based software process modelling, should open up new paths for major software process improvement.

Keywords: Software process, software evolution, process metrics, dynamics and improvement; Lehman's laws

1 Introduction

A 1968 study of the IBM software process [leh69,85] led, *inter alia*, to a metric based study of OS/360[1] evolution [bel72,leh74,80b,85]. Analysis of data relating ultimately to some 26 of its releases and sub-releases, identified and ordered by their release sequence number *rsn* [cox66], yielded a set of models of its evolutionary trends. An example of the output of the study is provided by figure 1.

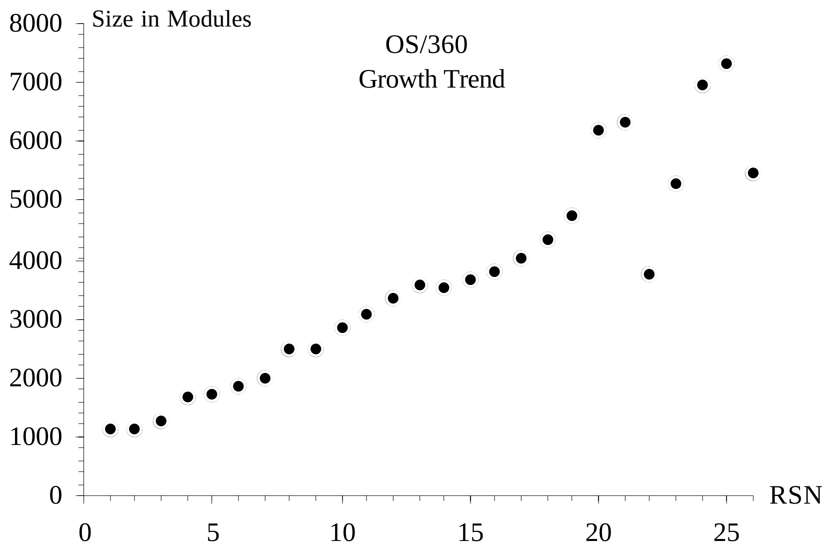

Figure 1 OS/360 growth trend by rsn

[1] All references in this paper to the IBM OS/360 system, refer to both that system and its successor OS/370

Up to release rsn_{21}, OS/360 growth as in figure 1 may be interpreted as steady growth with a superimposed *ripple*. This pattern is reminiscent of traces generated by self-regulating and self-stabilising systems with both positive and negative feedback [bel72, leh78]. The behaviour thereafter may be interpreted as a sign of instability induced by excessive positive feedback; excessive, rapid and ambitious functional evolution which led to a fission process; the transition from OS/360 to VS1 and VS2. Alternatively it may be interpreted as chaos-like behaviour. Either interpretation suggests that further prediction based on earlier behaviour is uncertain.

It was these observations that first suggested that the software process must be seen and treated as a feedback driven and constrained system [leh74,78]. Subsequently, examination of data on some non-IBM systems yielded many of the earlier observations [leh80b]. The feedback theme was also applied by Abdel-Hamid and Madnick [abd91] in their pioneering work on the use of *system dynamics*[2] in modelling aspects of the software process.

The 1970s study produced a series of models of OS/360 evolution as exemplified by figure 1. These revealed regularity unexpected for a process implemented, guided and controlled by humans. The behaviour could, however, be explained in terms of human and organisational behavioural patterns. It was therefore, captured in a series of statements abstracting the observed behaviour but lying outside the realm of software technology *as normally understood*. From the point of view of that technology, they must, therefore, be accepted as a regulating and constraining force. To overcome them requires expertise in organisational dynamics, management, sociology and other areas, not just software engineering. Thus they were subsequently referred to as *laws* of software evolution [leh74,78,80,85]. The laws, as proposed then [leh74] and subsequently amended [leh78,80], are summarised in Table 1. The date in column 1 indicates the year when each was first published. These conclusions could, however, be only be applied only to the IBM/operating system and process technology domains to which the data related. Generalisation depended on obtaining confirming evidence from other domains. Wider confirmation would provide a theoretical and practical base and framework for the evolution of *E*-type computer applications and software systems, that is, software solving a problem or addressing an application in the real world [leh80b].

No.	Brief Name	Law
I 1974	Continuing Change	*E*-type systems must be continually adapted else they become progressively less satisfactory.
II 1974	Increasing Complexity	As an *E*-type system evolves its complexity increases unless work is done to maintain or reduce it.
III 1974	Self Regulation	*E*-type system evolution process is self regulating with distribution of product and process measures close to normal.
IV 1980	Conservation of Organisational Stability (invariant work rate)	Average effective global activity rate in an evolving *E*-type system is invariant over product lifetime.
V 1980	Conservation of Familiarity	During active life of an evolving *E*-type system, average content of successive releases is invariant.
VI 1980	Continuing Growth	Functional content of *E*-type systems must be continually increased to maintain user satisfaction over their lifetime.
VII 1996	Declining Quality	*E*-type systems will be perceived as of declining quality unless rigorously maintained and adapted to changing operational environment.
VIII 1996	Feedback System (recognised 1974, formulated 1996)	*E*-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

Table 1 Laws of software evolution

2 Process Improvement

In recent years many businesses, and the software industry in particular, have developed a strong interest in and commitment to disciplined process improvement. More and more business processes are, however, dependent on software generated information. They are driven and controlled by computers and software, probably including *E*-type *legacy systems* that may have been operational for many years. Now specification and design of such systems requires assumptions about the intended application and its operational domain. These, in turn, will be reflected in the software. Subsequently, installation and operation of the system together with exogenous change will invalidate some of the embedded assumptions [leh89]. The system must, therefore, be continually updated to maintain their validity and adapt to changed circumstances. Business and software process improvement are strongly linked and interdependent [leh97].

[2] System dynamics modelling originated in the seminal work of Forrester and his colleagues at M.I.T [for61,70]. More recently it has been applied to the study of aspects of the software process, eg. [abd91,wae94,mad96]

The present paper focuses on the latter. The fact is that the software industry has been seeking improvement of the software development and maintenance process for many years [wil51,leh96]. Academic and industrial effort has yielded incremental improvement, through the introduction of new languages, formalisation, improved methods, mechanised support (CASE), new programming paradigms and so on. Nevertheless, the industrial track record raises the question why, despite so many advances, the global software development process from conception to use is still so often marred; why satisfactory functionality, performance and quality is only achieved over a lengthy evolutionary process, why software maintenance never ceases until a system is scrapped, why software is still generally regarded as the weakest link in the development of computer-based systems [leh94,96].

Explanations for individual failures can always be found. This paper summarises a more general approach arising from recognising that development and evolution processes for *E*-type systems are intrinsically feedback systems [leh94]. The remainder of the paper reports preliminary results from an investigation of this hypothesis.

3 FEAST (*F*eedback, *E*volution *A*nd *S*oftware *T*echnology) and FEAST/1

Some years ago, the realisation that feedback in the software process could explain the difficulties encountered in achieving its global improvement, led to the formulation of a FEAST hypothesis [fea94,leh94]:

As complex feedback systems, E-type software processes evolve strong system dynamics and with it the global stability characteristics of other such systems. Consequent stabilisation effects are likely to constrain efforts at process improvement.

More recently the hypothesis was restated in the following terms [leh96c]:

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

Both versions of the hypothesis include a number of assertions [leh96a] but these are not further discussed here.

The hypothesis and its implications were examined over a period of time by a FEAST core group consisting of Profs. M. M. Lehman, V. Stenning and W. M. Turski and Dr. D. E. Perry. Their deliberations [leh96b] led to three workshops held at Imperial College during 1994/5 [fea94,95]. The objectives were to expose the ideas to a wider group of people interested in the software process, to seek the objective criticism of experts and, in general, to explore the hypothesis and its implications.

The EPSRC[3] proposal that resulted from these discussions was entitled FEAST/1 [leh96c], the "/1" in the title indicating that this two year, 3 person project was to be seen as a first step in a longer and more widespread investigation. The proposal was approved in March 1996 and the resultant project commenced formal investigations in October 1996 in collaboration with ICL plc, Logica plc, Matra-BAe Dynamics plc and two groups within the UK Ministry of Defence. The stated objectives [leh96c] were as follows:

- provide objective evidence that feedback phenomena and the consequent system dynamics have substantial impact in the software process
- demonstrate that the phenomena can be exploited in both managing and improving industrial processes
- produce justification for a wider and more substantial study based on the feedback perspective

The two year investigation will seek to identify and characterise the feedback mechanisms active in the process, their impact on process characteristics and methods for applying the understanding gained to improve the respective processes. It expects to demonstrate (or otherwise) the feedback behaviour of some widely different systems. Three approaches are being employed:

- a *black box* approach will looks at quantitative data from a number of industrial software processes to identify patterns in the evolution of the respective systems. The data will be analysed in a search for footprints of dynamical behaviour and feedback control.
- A *white box* approach aims at the construction and enactment of system dynamics models of individual processes. These will reflect feedback mechanisms, their properties and their impact on the global process.
- A third approach, not included in the original proposal and not formally part of FEAST/1, is exploring the use of multi-agent systems [mca95] to model the selected processes and to evaluate proposed improvements.

Full investigation of the hypothesis and, if upheld, of means for its exploitation, is not straightforward. As previously discussed [leh96b], difficulties arises from several factors. The processes being investigated are likely to include tens, if not hundreds, of forward paths and feedback loops. A simulation approach such as the systems dynamics technique referred to in section 1 is, therefore more appropriate tool for the investigation than the analytical tools of control theory. Moreover, the processing and control mechanisms associated with these loops involve people, individually and in groups as managers or implementors. All observe, interpret, communicate, decide and act or refrain from acting on the basis of their overall perception, their instructions and, consciously or otherwise, their inclinations, experience and biases. Much of the feedback control is unplanned or even

[3] (UK) *E*ngineering and *P*hysical Sciences *R*esearch *C*ouncil

unconscious. Some, at least, of the feedback mechanisms are, therefore, stochastic and non-deterministic. Furthermore the system being modelled includes elements that contain implicit models of themselves. This is of course mathematically intractable posing a fundamental ultimate obstacle to the investigation [göd31, leh85]. Note also that convincing support for the hypothesis requires that the analysis and its associated predictive models, must necessarily be quantitative. The number of data points available for each of the classes of data is, however, likely to be relatively small. Statistical analysis and the determination of significance is, therefore, not straight forward. Work in the application of control theory to economic modelling [bec94] and, more recently, in the application of systems dynamics to aspects of the software process [abd91,wae94,mad96] suggest, however, that progress is possible. Note also that software engineers and others in the organisations in which and with which they work do not, in general, have the understanding, knowledge, skills or experience required for this analysis as part of their background. The long term study requires, therefore, a collaborative, extended, multidisciplinary approach to achieve exploitable results.

The remainder of this paper focuses on the initial results of the first FEAST/1 black box study. Much still remains to be done and the most significant contribution of the present project may well be to arouse wider international interest and so trigger the necessary collaborative investigations.

4 A First Case Study: the Logica FW system

4.1 FW data

After extensive discussion with the collaborators and others, selection criteria and candidate systems have been identified. The preference was for medium to large systems, how ever defined. It was also considered desirable to concentrate on systems that were being used in a number of locations so that the effect of users' feedback which, it is believed, is likely to have significant impact, could be be identified and assessed. Other criteria included the availability of historical data on system evolution to permit initial black box analysis to detect the presence or absence of feedback-like behaviour. Prior experience suggests that data on some ten releases is necessary for the identification of behaviour patterns, though preliminary data on a system in only its sixth release has also yielded interesting information. For the white box studies seeking to model internal process structure and to identify active feedback controls and their impact, ongoing projects were considered essential. Projects having, in addition, a sufficiently long history to provide meaningful black box data would be particularly useful since this would provide opportunities for linking characteristics inferred from the black and white box studies. At the time of writing collaborator products and processes satisfying these criteria have been identified, information on process structure and content is being gathered and metric data should be available shortly.

The first system evolution data to be made available to the FEAST/1 project was on the Logica plc Fastwire (FW) financial transaction system. This 8 years old system is now installed on some one hundred sites. The data set received covers the most recent 5 years of its evolution. Since then there have been several main releases and many more sub-releases. The data set as received from Logica related to some 100 releases (as defined by them) with each entry including three data items; release ID, size in modules and number of modules changed. Release dates were also available for most of the data points. Many of these releases were, however, of the same size as their predecessor. Clarification revealed that these were fix releases that were very frequently only transmitted to those (limited) number of customers adversely affected by a fault in an earlier release. A subset of the data was therefore selected. As more familiarity with the system history has been attained the criteria, and therefore the subset selected, have had to be changed to yield the set shown in shown in table 2. The analysis and plots presented below may, therefore, differ somewhat from those included in earlier publications [tur96], [leh96a,97]. The trends and patterns they display have, however, not changed significantly. Details of the selection and refinement criteria applied to the data have been documented in an internal report.

RSN	Size in Modules	Release ID	RSN	Size in Modules	Release ID
1	977	1.0	12	2087	5.0A
2	1344	2.0A	13	2091	5.0B
3	1390	2.0B	14	2095	5.0C
4	1492	2.0C	15	2101	5.0D
5	1581	2.0D	16	2151	5.0E
6	1595	2.0E	17	2167	5.0F
7	1800	3.0A	18	2312	6.0A
8	1832	3.0B	19	2315	6.0B
9	1897	4.0A	20	2696	7.0A
10	1897	4.0B	21	2699	7.0B
11	1902	4.0C			

Table 2 The FW data set

To protect their identity, the IDs of the releases listed in table 2 have been replaced by a sequence of identifiers that replace those assigned by Logica. In addition, and as was done in the OS/360 study [leh80b], consecutive integers have been assigned to the releases comprising the evolution sequence to be analysed. These provide a *pseudo-time* measure designated the rsn (release sequence number) in the sense of Cox and Lewis [cox66]. Basing the analysis on this measure is appropriate because only at the instant of release of an *E*-type software system are its properties, as determined by the then established software text, uniquely defined. By definition, an *E*-type system operates in a domain always liable to change at a rate that is accelerated by development, installation and operation of the system. Thus the software too, that is the code and/or its documentation, must be repeatedly updated and adapted[4] to remain a faithful model of the application in its operational domain. At the time of release the text is, by edict, fully defined. At all other times it is likely to be in a state of flux [leh85].

The releases included in the analysis whose results are presented below may be categorised into three classes:

- Major mainstream releases. These are intended to be adopted by the majority of user organisations. They are often required to achieve standardisation or for legal reasons.
- Minor mainstream releases. These provide minor improvements or enhancements. Such releases are included in the analysis presented below only where, in addition to other criteria, at least one module has been added or deleted with respect to its evolutionary predecessor.
- *Error correction* releases. These neither add nor enhance functionality. These also have been included in the analysis if they involve system growth by, at least, one module.

A fourth release type, termed *ad-hoc*, is aimed, in the first instance, at the satisfaction of the requirements of a specific client. Such releases are excluded from the results presented below. However, unless providing some temporary facility later to be removed, the enhancements included in such releases are sooner or later integrated into the main stream to maintain smooth, uniform evolution over all installed systems and simplify overall FW configuration management. Moreover, these releases absorb project resources and, therefore, impact other concurrent activities. Thus they need, ultimately, to be included in the analysis.

A consequence of ordering releases by rsn in the presence of the various types of releases is that a situation may arise in which the ordering adopted differs from the date ordering. For example, work on a new mainstream version may have been proceeding concurrently with minor enhancement or bug-fixing of an older release. The former may be shipped to mainstream clients before the latter is ready for delivery. For example, and as illustrated in Figure 2, release 2.0D precedes release 3.0A in real time and might, therefore, appear to be its parent. The former will, however, have inherited functionality and code first developed for and integrated into 2.0E. Therefore, in the evolutionary sense release 2.0E is the predecessor of 3.0A and is given the lower rsn.

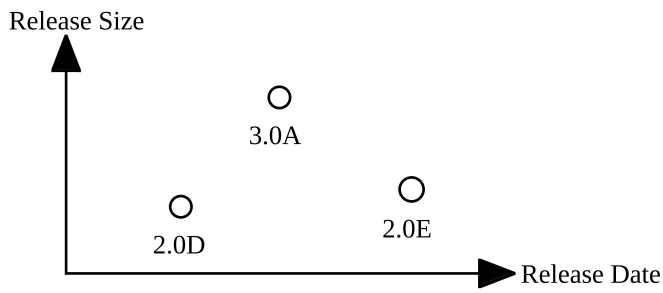

Figure 2 Example of release ordering by date (not to scale)

4.2 System Growth

With the cost of storage declining at all levels, system size is, in itself, not of major concern. It may, therefore, be seen as a independent and composite monitor of system evolution which, within limits, is neither planned nor managed. It is determined by other factors. Some of these will be managed, others "just happen". Size determinants include system design, programmer style and experience, development timetables and constraints, intensity of the desire to achieve compactness or clarity[5]. The great majority of reported software metrics work has tended to use *locs* (lines of code) as the measure of system size. As in the case of the original OS/360 study [leh80b,85] the FW analysis reported here has used *module count* for that purpose. In the absence of a better measure, module count also serves as an initial estimator of system functionality and power. The 1970s study did, in fact, compare the results of loc and module based studies. It was shown that these were essentially similar but with the locs measure providing a less consistent picture of the evolutionary behaviour of the system than did the module count. Locs are, therefore, considered inferior as a measure. The superiority of module count was explained by the observation that, however modules are defined, they have, within a given domain, some degree of functional integrity whereas locs have none [leh85]. The number of modules in a specific system is also not,

[4] As per the first law of software evolution as reproduced in table 1

[5] Or even, where productivity is measured in locs and the concern is with *productivity* improvement

in general, dependent on individual programmer practice. Module numbers may, therefore, be expected to provide a more consistent measure of system size and hence a better, though admittedly coarse, indicator of system functionality or power.

The *function point* (FP) [alb79] measure may be considered an alternative measure for system functionality or power. Their use does, however, raise some questions. For example, how arbitrary are the interpretation of the definitions or the factor ratings achieved and, as a consequence, how consistent are the results obtained by different raters? Moreover, the establishment of the measure requires judgment based on subjective measures and the overall determination is labour intensive and difficult to automate [kem93]. There is also little, if any, experience in application of the measure to larger systems. Finally, it must be observed that, unlike module count, FP data is not widely available from data archives of software systems across the industry. They are certainly not available from the systems being offered for study by the FEAST/1 collaborators. Module count is, therefore, being used as a size measure and, by implication, as a system power estimator, in the FEAST investigation. To date, and as illustrated by the results presented below, this decision appears to be justified.

The growth in modules of FW over releases rsn_1 to rsn_{21}, that is releases 1.0 to 7.0B is shown in figure 3. The overall growth pattern should be compared with that of OS/360 over its first 20 or so releases as in figure 1.

The abscissa of figure 3 represents the individual release sequence numbers as explained above. The figure clearly shows the upward trend of system growth. The trend is, therefore, consistent with the first and sixth laws of software evolution but does not distinguish between them. Moreover, it also shows a ripple effect strongly reminiscent of that of OS/360 as in figure 1. It was, of course, this ripple phenomenon that first suggested that the software process was stabilised by feedback control, as captured in the third and eight laws. Thus this initial result of the present study is certainly compatible with the conclusions and, in particular, the laws of software evolution, first reached in the study of OS/360 more than twenty years ago.

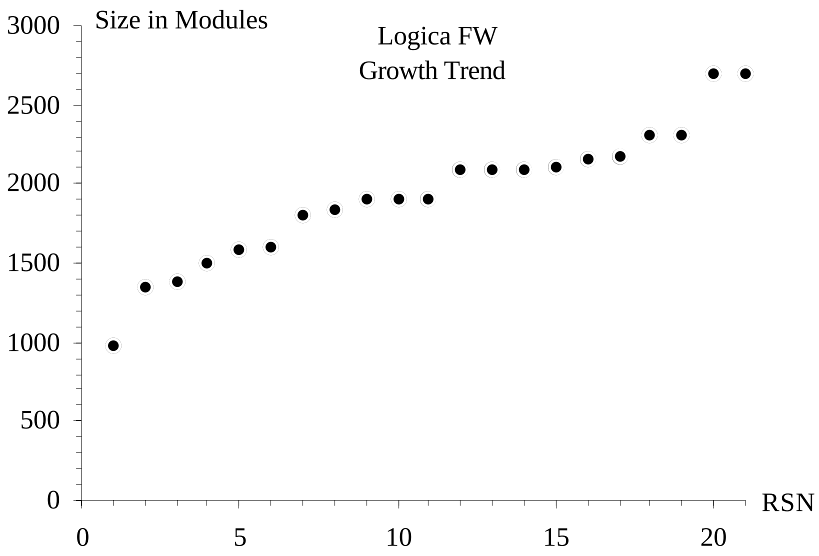

Figure 3 FW growth trend by rsn

4.3 Incremental Growth

Figures 4 and 5 show the incremental growth per release of OS/360 and FW respectively over the releases rsn_1 to rsn_{21} for each system. The horizontal line indicates the average growth per release over this range. For FW the plot includes all the data in table 1. For OS/360 the final five releases for which data is available are omitted from the plot since they reflect the transition (in growth trend terms) from OS/370 to VS1 and VS2.

The two plots display remarkably similar cyclic characteristics though, as one should expect, they differ in detail. They also resemble statistical process control charts [dav84]. It was, in part, the observation of this cyclic pattern and its symmetry around the average with respect to OS/360 that originally led to formulation of the third and fifth laws. The long term trend of the moving average of the incremental growth of *E*-type systems as they evolve will, in general, be difficult to determine because of the small number of data points generally available. It might be the case that this average declines because of increasing complexity. Alternatively, it might grow as a consequence of improving process technology. It may also be that these (and other?) contrary pressures compensate for each other over time so that the original assertion of invariance remains valid. It is, in fact, not certain that all systems or domains behave in the same way. The fifth law as stated in table 1 will have to be re-examined.The analysis outlined below will provide additional insight for FW with the preferred growth trend model indicating that, for that system at least, the first trend is the most likely.

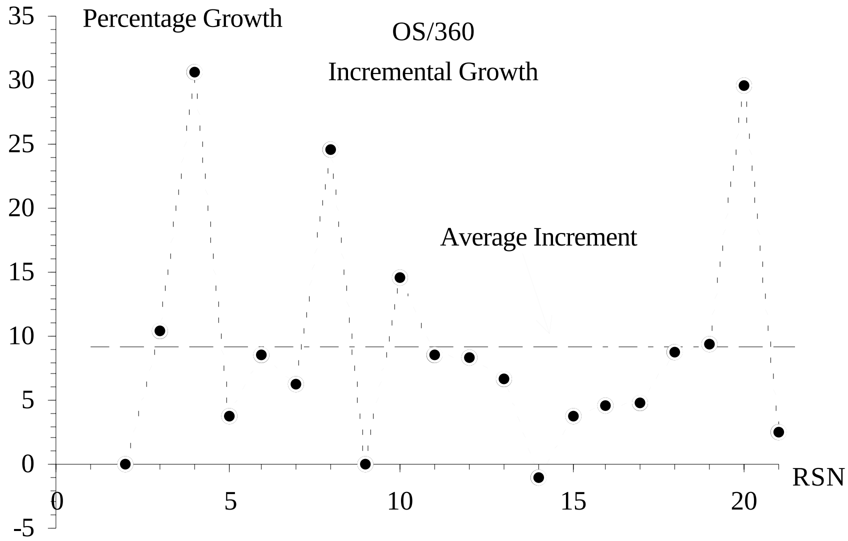

Figure 4 Incremental growth per release of OS/360 over releases rsn_2 to rsn_{21}

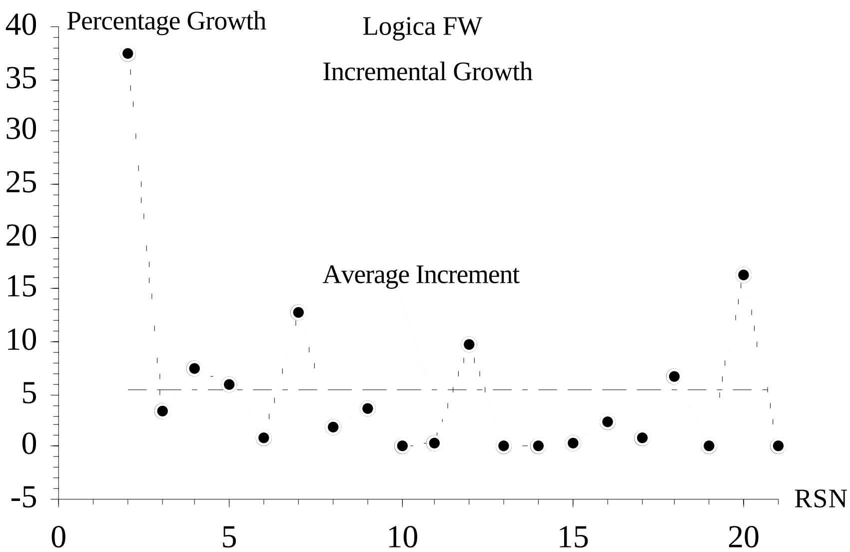

Figure 5 Incremental growth per release of FW over rsn_2 to rsn_{21}

As pointed out previously [leh74,78], the cyclic effect reflected by the *peaks* and *troughs* in the incremental growth plots may be indicating the presence of feedback driven and controlled growth. Thus, influences tending to increase system functionality, that is growth towards the peaks, may have their source in positive feedback. The declines may reflect size stabilisation and other negative feedback effects. An example of such feedback is the evolutionary pressure that arises when clients and users express a need for enhancements to existing capability or system extension. But as implementation of such changes proceeds the size and complexity of the system increases, leading to declining comprehendability, increasing error rates, increasing resistance to change or the impact of budgetary constraints. These lead, for example, to declining *growth* resources as the need for *fixing* resources and complexity control increases [leh85]. The process will be directed in its evolution and growth patterns by data reflecting these effects. That is, the data or its derivatives will be used to adjust process objectives (immediate and/or long term) and process parameters and used to drive, constrain, and in general, manage the process. Positive feedback drives growth while negative influences force a period of consolidation (correction and restructuring). An example of the former is provided by the final 7 releases, rsn_{20} to rsn_{26} of OS/360. It represents, in fact, an possible example of the consequences of excessive positive feedback. The hypothesis is that the instability over releases, rsn_{22} to rsn_{26} (figure 1), resulted from excessive growth, in response to market demand, in going from rsn_{19} to rsn_{20}.This brief analysis suggests that the FW data supports, in part at least, the third and fifth laws of software evolution as originally inferred from OS/360 evolutionary behaviour. Analysis of the long term growth trend of FW in the next subsection suggests, however, that the law, as stated in table 1, must be modified.

4.4 The Inverse Square Model (IS)

This section presents two models of FW growth. The first of these, illustrated in figure 6, is obtained from the data set of table 2 using a least square linear (LSL) fit. The models focus on the general trend and largely ignores the ripple. Detailed analysis of the latter is beyond the scope of this paper.

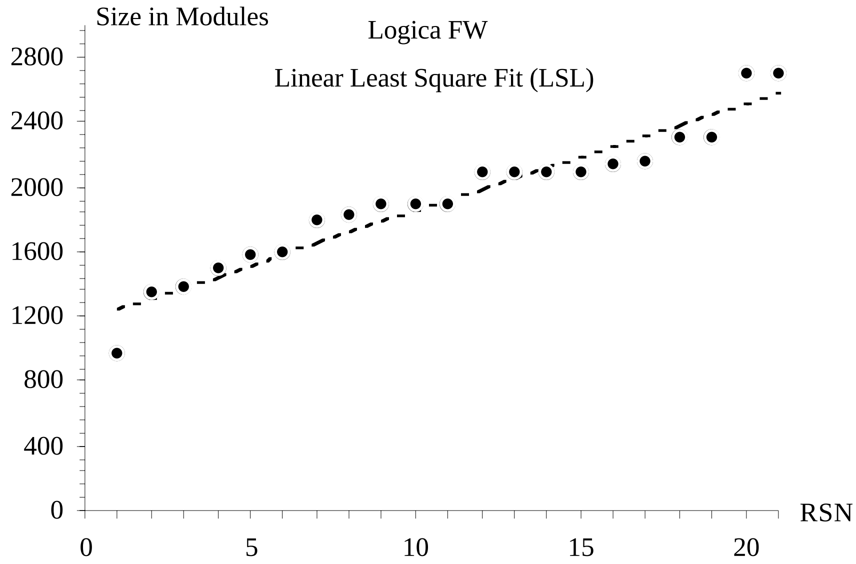

Figure 6 Least square linear fit to FW over rsn_1 to rsn_{21}

After investigating other alternatives Turski developed an alternative, inverse square, model (IS) represented by the nonlinear discrete-time dynamical recursion (1) [tur96]. In this model s_i is the actual value of rsn_i, $^{\wedge}s_i$ is its fitted or predicted size, "n" is the total number of releases in the data set and $\underline{E}$ is a model parameter.

$$^{\wedge}s_1 = s_1 \qquad (1a)$$

$$^{\wedge}s_i = {^{\wedge}s_{i-1}} + \underline{E}/(^{\wedge}s_{i-1})^2 \;\ldots\ldots\ldots\ldots\{i = 2,\ldots, n\} \qquad (1b)$$

The parameter $\underline{E}$ is the average of individual E_i, calculated from either (2) or (3).

$$E_i = (s_i - s_{i-1})s_{i-1}^2 \;\ldots\ldots\ldots\ldots\{i = 2,..., n\} \qquad (2)$$

$$E_i = (s_i - s_1)/\left(\Sigma_{k=1}^{i-1}(1/(s_k)^2)\right) \;\ldots\ldots\ldots\ldots\{i = 2,..., n\} \qquad (3)$$

Algorithm (2) uses only the two most recent data points in computing E_i. With (3) all data to rsn_i are considered. In either case the average of the resultant set of E_i gives an estimated value for $\underline{E}$. A third approach (LSIS) computes $\underline{E}$ from the entirety of data using a least square criterion and is illustrated in Figure 7.

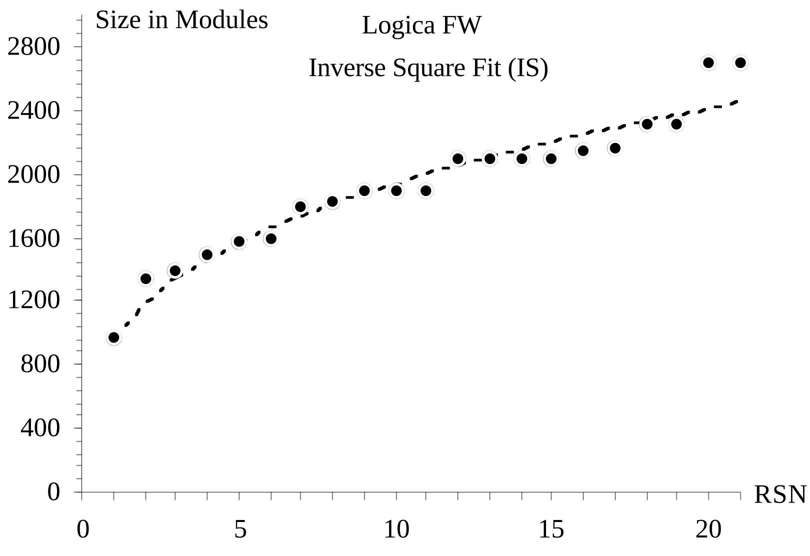

Figure 7 Least square inverse square fit to FW over rsn_1 to rsn_{21}

The conceptual implications guiding the selection of one of the three alternative algorithms for computing E are subtle and are not discussed further here. They yield slightly different values of E but in the context of this study they do not produce significantly different behavioural patterns. Nor do they change the conclusions to be drawn. Finally, the observant reader will notice apparent outliers rsn_{20} and rsn_{21}. No comment can be made at this time about the significance of these or their possible implication.

For the trend models based on the full set of data points statistical measures of the closeness of fit of the LSL and IS models do not differ significantly. This suggests that, at the present stage of the FW life cycle the inverse square damping on the long term trend is not strong. Moreover, neither model addresses the ripple. The FEAST hypothesis was triggered by the assumption that the pattern apparent in the ripple is not primarily due to the impact of a sequences of localised (often short term) management and implementation decisions superimposed on what, as indicated below, appears to be a "*system*" imposed long term trend. It was presumed to indicate a degree of feedback-controlled self-stabilisation as indicated by the fact that measures of the regularity of the ripple deviations appear to be at least as great, possibly larger than the apparently random disturbances. A current hypothesis is that the ripple as observed is, in fact, a composite of deviations from the trend caused by local management decisions and technical variations on the one hand and those arising from the system dynamics. This requires further investigation.

As pointed out by Turski [tur96] the phenomenology of the situation suggests several reasons for preferring the IS model:

- The IS inverse square property can be interpreted as reflecting the complexity growth of a software system over a sequence of releases. Such growth is due, in part, to increases in the complexity of the application, for example, as features not included in the original system definition, and often orthogonal to it, are added. Moreover, the process of evolution adds change upon change upon change with, in general, little attention paid to the resultant complexity growth [leh85]. It is this phenomenon that is captured by the second law (table 2).
- As a one parameter model IS is also compatible with the fourth laws of software evolution, with the parameter E reflecting the constant effort that the law identifies [leh78].
- IS also satisfies the Principle of Parsimony [box66].
- No system can grow forever. The linear growth model is thus incompatible with reason and common experience.

4.5 Further Consideration of the Inverse Square Model

The list of reasons for favouring IS over the LSL includes the observation that the single parameter E of the former may be interpreted as a *constant effort* parameter as predicted by the fourth law. Estimation of E from the available data strengthens that argument. Such estimation produces a value that, as shown below, remains relatively constant as FW evolves. That is, the single parameter of the model may be interpreted as the constant *effort* or *work output* identified in the fourth law as being required to take the system from one release to the next. The principal questions raised by this interpretation, questions not satisfactorily answered, relate to the interpretation of E and the units in which it is measured. Does E relate to the input effort required to achieve release by release system evolution or to the output achieved from the process measured by some measure of increase in system quality and power? To answer the first question requires further investigation and additional data. As to units, s_i is a dimensionless count. Hence E is dimensionless. But despite these unsolved questions it is concluded that, on the basis of currently available data, the above points, together provide some justification for preferring the inverse square model. It appears to reflect reality more closely.

The full implications of one further indicator of the superiority of IS over LSL must now be considered. When modelling large data sets, the first part is often used to estimate model parameters and the second to then evaluate its "predictive" capability [ger93]. With the small size of the data set available from FW, this might not appear to be a fruitful path to follow. Turski [tur96] did, however, investigate this question, asking: "How many points beginning with rsn_1 have to be considered in order to get an appropriately low error of fit over those points, an acceptable predictive capability?" In terms of the FEAST hypothesis this question is equivalent to asking: How fast is the FW dynamics established? An answer is suggested for FW by the plot of figure 8.

The figure shows the mean absolute error of fit (mae) that is relatively large over the first releases but declines rapidly, reaching a stable state by release rsn_6. Thereafter the *mea* varies has a mean of 74.6 with a standard deviation of 2.8 for estimates of E using from 6 to 21 points. At the stabilising point the model derived from data on only 6 releases yields an mae that is only 4.7% of the system size at rsn_6, 3.2% of its size at rsn_{19} and 2.8% of its size at rsn_{21}. This behaviour is counter-intuitive in several ways. Possible interpretations and implications are summarised below. Overall, it does, however, appear to indicate the strength of the system dynamics. Its constraining influence provides some justification for the observation made by one of the authors many years ago that "Rather than the managers managing the (evolving software) system, the system manages the managers." It must, of course, be understood that the reference here is to long term evolution, not to the specifics of individual decisions, often localised in time, system space and implementation space.

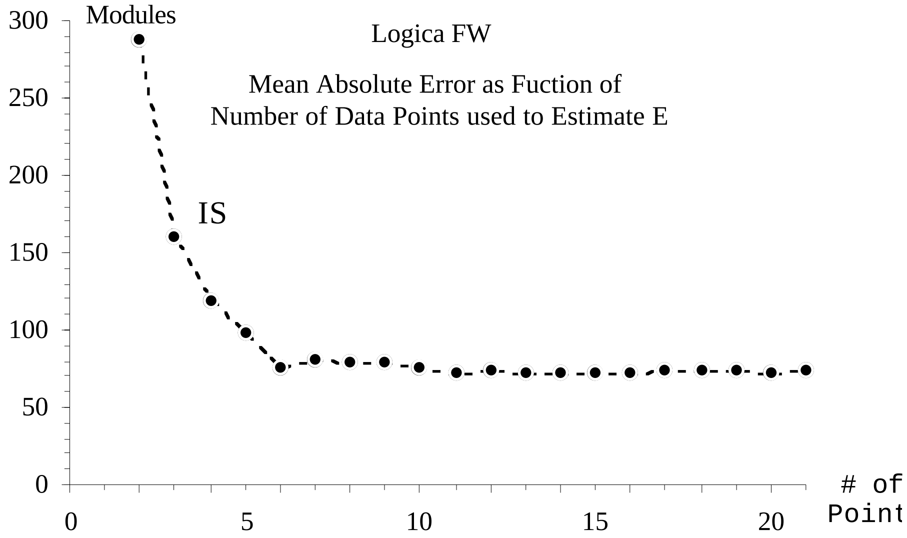

Figure 8 Mean absolute error of FW fit as a function of the number of data points used to estimate IS model

- Figure 8 based on the IS model suggests that the FW growth trend is established over some six of the releases included in the study. In accordance with the FEAST hypothesis, it is assumed that the dynamics arises from the characteristics of the software, the organisations developing, marketing and using the software, the communications between them and the controls that are exercised. In any event figure 8 supports the hypothesis that *E*-type software systems in evolution develop a strong momentum.
- The *mea* for IS of 74.6 modules with standard deviation of 2.8 over the stable range is very close to the calculated average incremental growth of about 86.1 modules over all data points (fig. 5). This raises the question whether there is some relationship between the variance of the ripple (which is a significant source of error for the trend fit) and the mean incremental growth. Establishing a correlation would lead to a concept of *safe growth* or *growth rate* limits. Establishing either would provide strong conceptual support for the *incremental* or *evolutionary release* strategy [gil88].The entire question remains to be investigated.
- Note that the *mea* of LSL over the stable range is, at 86 modules, even closer to the average incremental growth of 86.1 modules than is that of IS. The implications of this, for example on the evaluation of the relative value of the two models requires more investigation.
- The IS plot in Figs. 8 and 9 stabilises much more rapidly than does the LSL plot. Moreover, if IS and LSL are estimated by using only rsn_1 and rsn_2, the former outperforms the latter by an order of magnitude. Thus while there are still unanswered questions, figures 8 and 9 appear to support the earlier conclusion that IS is to be preferred over LSL. That they provide further support for the FEAST hypothesis and the laws of software evolution does not require further emphasis.

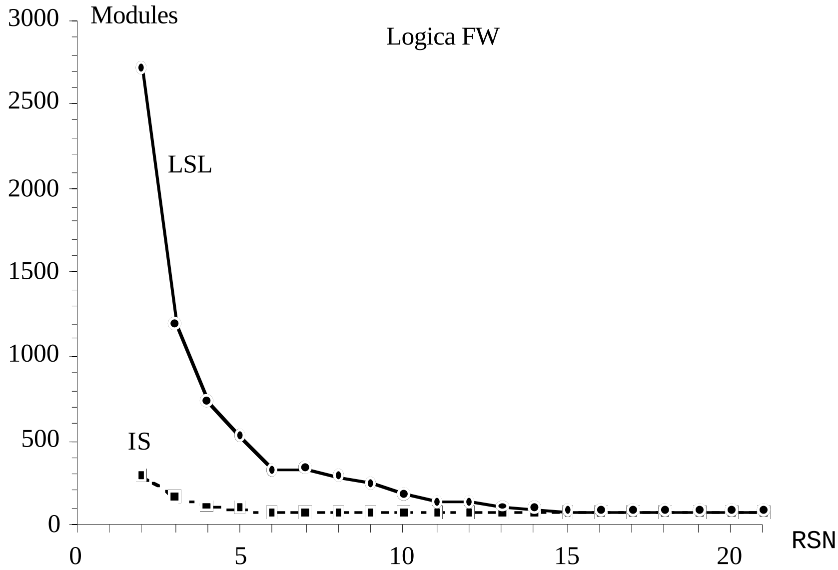

Figure 9 Mean absolute error of FW fit as a function of the number of data points used to estimate the LSL model (full) superimposed on that of the IS model (dashed)

The initial results presented above are based on the examination of only one current system. They provide significant evidence which should encourage further investigation. Note also that the investigation of OS/360 in the light of current findings has not been reopened.

4.6 Impact of the Study on the Laws of Software Evolution

More work is clearly required for firm conclusions to be reached in regards to the many issues raised above. It is nevertheless considered appropriate to indicate in table 3 the extent to which the investigators feel encouraged to see the present results as being compatible with, or even supporting, the laws of software evolution. The impression is gained that, despite the 20 year gap and the significant difference between IBM and Logica systems and usage environments, there are strong similarities in the phenomenology of their evolutionary growth. On the basis of the phenomenological interpretations that, because of space limitations, cannot be presented here it appears is likely that the results of this preliminary study will extend, possibly with modifications or generalisations to other *E*-type systems. Under the FEAST/1 project it is hoped to analyse data from such systems and their development processes and to seek confirmation or denial of the present initial findings.

No.	Support	Indicator
I	?[5]	More investigation is required to determine whether fig. 3 growth trend includes adaptation and change and so supports I or only functional growth as in VI
II	√	Inverse square law of growth, eq. (1), constant effort parameter <u>E</u>, figs. 3 and 7
III	?	Figure 3 suggests regularity but not normality. Further investigation needed
IV	√	Ability to obtain close fit with constant <u>E</u> in equation (1)
V	?[5]	Figure 5 which is similar to equivalent data in the OS/360 data. Conclusion is, however, tentative, awaiting observation of the consequences of a release whose content exceeds the average by a significant amount
VI	√	Demonstrated by the growth of as in figure 2 and analysis of its content
VII		No evidence for or against
VIII	√	Self stabilisation as in figs. 3, 5, 7, 8. Inverse square law, eq. (1). Proof that process is a feedback system requires identification of feedback mechanisms and cannot be inferred from quantitative data.

Table 3 The laws of software evolution in the light of the preliminary FW analysis[6]

5 Final Remarks

The present paper describes a black box approach intended to display the phenomenology of FW evolution, to provide a display of the influence of the dynamics, a behavioural model for interpretation and the basis for the formulation of explanatory hypotheses. A white box modelling approach is simultaneously seeking to model the structure of some industrial software processes including their feedback control loops and to simulate their behaviour. These investigations are being further backed up through the development of a multi-agent model. It is hoped that, together, they will confirm the laws of software evolution that now include the FEAST hypothesis and putting it on a firm foundation. Alternatively the results of the investigation could demonstrate that the laws and the hypothesis are not of general relevance though, possibly, satisfied for particular instances of *E*-type systems and their evolution processes. If successful over a range of systems the investigation will represent the basis for a plausible theory of software process and software evolution.

This preliminary study has already made visible progress in illustrating how metric concepts can be applied to the study of software evolution. It is succeeding in extending the 1970s techniques and applying more rigour [law82] to mastery of the observed phenomena. The specific results for the Logica system are of considerable interest, both in themselves and from a wider point of view. Nevertheless, the reader should appreciate that the primary significance of this paper, if any, is in the perspective and approach it presents.

Apart from the possible theoretical advance this study represents it should, if successful, lead to the development of methods and tools for process management, release planning and process improvement. This will shape the direction of software metrics, software process modelling and process improvement in the years to come. If the extent to which feedback phenomena in *E*-type evolution processes shapes and constrains the software process significantly, mastery and command of that phenomena will open up important new prospects. Moreover, the software process may also be seen as a special case for business processes, in general. The approach applied and the conclusions reached should find much wider application. It is believed that FEAST/1 is a study which, if successful, will eventually lead to a theory and to a technology which together can trigger major advances in the software and other business processes and their improvement.

[6] It is hoped to obtain more data that will provide evidence, one way or the other

6 Acknowledgements

We are grateful to Logica plc for providing access to the FW data and in particular to Joe Halberstadt for his collaboration. Sincere appreciation is also due to the FEAST core group (Profs. Vic Stenning, Wlad Turski and Dr. Dewayne Perry), for sharing their intuitions, insights, comments, experience and interest over the years with the first author, so clarifying the concepts and the results discussed. Also to Prof. Berc Rustem for his helpful insights into control theoretic aspects of this study and to Dr. Emma McCoy for her help with statistical aspects. Since October 1996 the work reported here has been supported under EPSRC grants numbers GR/K86008 and GR/L07437.

7 References[7]

[abd91] Abdel-Hamid T and Madnick SE, *Software Project Dynamics - An Integrated Approach*, Prentice-Hall, Englewood Cliffs, 1991 264 pps.

[alb79] *Measuring Application Development Productivity*, Guide/Share: Proc. IBM Application Development Symposium, Monterey, CA, 1979, pp. 83 - 92

[bec94] Becker RS, Hall B,and Rustem E, *Robust Optimal Control of Stochastic Nonlinear Economic Systems*, J. of Economic Dynamics and Control, n. 18, 1994, pp. 125 - 148

[bel72] Belady LA and Lehman MM, *An Introduction to Growth Dynamics*, Proc. Conf. on Statistical Computer Performance Evaluation, Brown Univ. 1971, Academic Press, 1972, W Freiberger (ed.), pp. 503 -511

[box70] Box GP and Jenkins GM, *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco, 1970, 553 pps.

[cox66] Cox DR and Lewis PAW, *The Statistical Analysis of Series of Events*, Methuen, London, 1966, pps.

[dav84] Davis OL and Goldsmith PL, *Statistical Methods in Research and Production*, 4th. ed. Longman, London, 1984, 478 pps

[fea94,5] *Preprints of the three FEAST Workshops*, Lehman MM (ed.), Dept. of Comp., ICSTM, 1994/5

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

[for70] Forrester JW, *Understanding the counter intuitive behaviour of social systems* in *Systems Behaviour*, edited by Open Systems Group, Third Edition. pp. 270-287, Paul Chapman Publishing Co. and The Open University. London, 1972

[ger93] Gershenfeld NA and Weigend AS, *The Future of Time Series: Learning and Understanding*, in Time Series Prediction: Forecasting the Future and Understanding the Past, Gershenfeld NA and Weigend AS (eds.), SFI Studies in the Sciences of Complexity, Proc. Vol. XV, Addison-Wesley, 1993, pp. 1-70

[gil88] Gilb T, *Principles of Software Engineering Management*, Addison Wesley, 1988

[göd31] Gödel K, *Über formal unentscheibare Sätze der Principia Mathematica und verwandter Systeme, I*, Monatshefte für Mathematik und Physik 38, 1931, pp. 173-1198. An English translation is provided as *On Formally Undecidable Propositions*, K. Gödel, Basic Books, New York, 1962

[kem93] Kemerer CF, *Reliability of Function Point Measurement: A Field Experiment*, CACM v. 3, n. 2, Feb. 1993, pp. 85 - 97

[law82] Lawrence MJ, *An Examination of Evolution Dynamics*, Proc. 6th. Int. Conf. On Softw. Eng., Tokyo, Japan, 13-16 Sept. 1982, IEEE Comp. Soc. Ord.N. 422, IEEE Cat N. 81CH1795-4, pp.188-196.

[leh69]* Lehman MM, *The Programming Process*, IBM Res. Rep. RC 2722, IBM Res. Centre, Yorktown Heights, NY 10594, Sept. 1969

[leh74]* Lehman MM, *Programs, Cities, Students, Limits to Growth?*, Inaugural Lecture, May 1974, Publ. in Imp. Col of Sc. Tech. Inaug. 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] Lehman MM, *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Spec. Iss. on Softw. Eng., v. 68, n. 9, Sept. 1980, pp. 1060 - 1076

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

[leh89] Lehman M M, *Uncertainty in Computer Application and its Control Through the Engineering of Software*, J. of Software Maintenance: Research and Practice, v. 1, n. 1, Sept. 1989, pp. 3 - 27

[7] Papers identified by a * in the reference listing are reprinted in [leh85].

[leh94] Lehman MM, *Feedback in the Software Evolution Process*, Keynote Addr., CSR Eleventh Annual Workshop on Softw. Evolution: Models and Metrics. Dublin, 7-9th Sept. 1994, in Information and Softw. Tech., sp. Iss. on Softw. Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp.681 - 686.

[leh96a] id., *Process Improvement - The Way Forward*, Invited Keynote Address, Proc. Brazilian Softw. Eng. Conf., SBES'96, Universidade Federal de Sao Carlos, Brazil, 1996, pp. 23 - 35

[leh96b] Lehman MM, Perry DE and Turski WM, *Why is it so Hard to Find Feedback Control in Software Processes?*, Inv. Pres., Proc. 19th Australasian Comp. Sc. Conf., Melbourne, Austr., Jan. 31 - Feb. 2, 1996

[leh96c] Lehman MM and Stenning V, *FEAST/1: Case for Support*, ICSTM - DoC EPSRC Proposal, March 1996

[leh97] Lehman MM, *Process Modelling - Where Next*, ICSE 9 Most Influential Paper Award, Proc. ICSE 19, pps. 549 - 552

[mad96] Madachy RJ, *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

[mca95] McCabe FG and Clark KL, *Programming in April: An Agent Process Interaction Language*, in Intelligent Agents, Springer Verlag, 1995.

[mea72] Meadows DH *et al*, *Limits to Growth*, Signet, 1972,

[tur96] Turski W, *Reference Model for Smooth Growth of Software Systems*, IEEE Trans. on Softw. Eng., vol. 22, n. 8, 1996.

[wae94] Waeselynck H and Pfahl D, *System Dynamics Applied to the Modelling of Software Projects*, in Software - Concepts and Tools, v. 15, Springer-Verlag, Berlin, 1994, pp. 162 - 174

[wil51] Wilkes M V, Wheeler D J and Gill S, *The Preparation of Programs for an Electronic Digital Computer*, Addison Wesley Press Inc., 1951, 167 pps.

mml568[papers]
20/6/97