The Influence of Global Factors On Software System Evolution

MM Lehman, JF Ramil and PD Wernick

Department of Computing
Imperial College
London SW7 2BZ
tel: +44 (0)171 594 8214
fax: +44 (0)171 594 8215
e-mail: {mml,jcf1,pdw1}@doc.ic.ac.uk

ABSTRACT

The FEAST/1 project is investigating a hypothesis that
the software process is a multi level feedback system.
Using a number of complementary approaches to model,
analyse and interpret metric and other data records of
the evolution of systems from various application ar-
eas, the project has made significant advances in de-
tecting the effect of feedback and related dynamics in
the their evolution. Results obtained to date, supported
by theoretical reasoning, support the view that software
process improvement must focus on the global process
including its feedback mechanisms. In addition to tech-
nical aspects of the process, organisation, management,
marketing, user support and other factors must also be
considered. This paper discusses the conceptual frame-
work within which this question is raised, presents pre-
liminary evidence to support tentative conclusions and
indicates how the issue might be further investigated.

Keywords

Software Process, Evolution, Process Modelling,
Metrics, Feedback, FEAST, System Dynamics

1 The Global Software Process

In referring to the software process, numerous publica-
tions reporting the work of the FEAST/1 project [6]
apply the adjective *global*. The term is used to include
in that process more than just the people, technical
analysis, decision taking and implementation activity
required to develop a system *ab initio* or to make a
change to an already existing system. It also involves
those who influence the process, in one way or another,
by their decisions and actions at many levels of manage-
ment, marketing and application. They fuel the pro-
cess, determine, set and adjust objectives, define new
needs, change goals, fit the development into other or-
ganisational activities, control progress in the context
of changing organisational circumstances and evolving

application domains, explore and serve the marketplace
and so on. Formal and informal communication and
control paths between these numerous groups and in-
dividuals, with their technical, managerial, marketing
or user orientation, provide information flow and con-
trol paths. The flow may be in forward activity paths,
transmitting, for example, specifications to designers,
designs to implementors, implementations to testers,
validated elements to integrators, shippable subsystems
or systems to user organisations. Equally, information
may flow from corporate management defining chang-
ing targets or budgets to line managers for allocation to
individual activities. Marketeers may communicate de-
tails about future products or changes in policy to users.
As the project progresses, information and control from
many sources will be fed back to management who, in
turn, pass directives or other information to technical
management for implementation, so influencing process
activities. Changing organisational directions based on
management observation of project progress and out-
put, recognition of market opportunities and needs, in-
formation on competitive challenges, conveyance of user
requests for change, development of new needs, report-
ing user attitudes, delivery of performance and fault re-
ports, other quality issues, lead to changes in, for exam-
ple, specifications, process activities, code, timings and
so on. There is no end to the different types of informa-
tion and control flow that inevitably appear, intermit-
tently, or periodically. And each type has its own char-
acteristic spectrum of attributes in terms of frequency,
volume, authority, reliability and so on. These drivers
of the process show why study of the software process
and its improvement dare not concentrate on technical
activity alone. They indicate the need for a wider per-
spective on the software process that takes into account
the drivers, creative contributions and control activities
by managers, marketeers, users and others as exempli-
fied by the examples above. They lead to the adoption
of the *global process* view as contrasted with the more
prevalent *technical process* view.

2 Feedback

The eighth law of software evolution [14], previously
known as the FEAST hypothesis, states that the global

software process is a multi level, multi loop, multi agent feedback system. This assertion was first made [1] following a study of the programming process [7], though not then presented as either a hypothesis or law. As recorded in numerous publications [10, 6], the phenomenon that led to this assertion was a ripple appearing on the otherwise smooth growth trend of OS/360 (figure 1). Strikingly similar behaviour has now been observed on a total of seven other systems developed and evolved by industrial organisations, three studied at the end of the seventies, four more in the FEAST/1 project over the past two years. The new evidence emerging from that study greatly strengthens the case for feedback influence on software evolutionary growth [12, 13, 18]. In section 1 some of the information agents driving this behaviour were mentioned.

The dynamic behaviour of systems with feedback mechanisms is determined by the properties of both forward and feedback paths as well as by exogenous factors. Provided that the properties of the feedback mechanisms are adequate, the behaviour of the system will be less sensitive than open loop systems to, for example, changes in the characteristics of internal mechanisms or to noise. Feedback may be positive or negative. The former causes growth in some attribute of the process, its products or or other process objects and can, if excessive, cause instability. The latter maintains system stability and performance. It constrains the effect of changes on the outputs in sources other than system inputs. Feedback concepts, the system behaviour it induces and the metrics by which these may be observed are generally understood and well defined in the context of physical systems. They are formalised, for example, in control theory.

After an initial start up period, the system and behavioural attributes will reflect not only the state of the implementation domain during some interval surrounding that time but also the previous history and states. The precise relationships will be a complicated function of the properties of the properties of the various loops, particularly those containing the object being observed, the gains and delays in the forward and feedback paths, for example. The dynamic behaviour of the process and its constituents will be determined by all of the above. Analytic treatment as developed for example in control theory, is possible, subject only to limitations arising from system size and complexity. Should either of the latter prove an obstacle to such analysis, computer based numerical solutions provide an alternative.

To the best of our knowledge no equivalent work has been reported in the context of the software process. There is every reason to believe that the FEAST/1 project and the earlier work that underlies it is the first extensive study of the phenomenology of feedback in the software process. Such a study is an essential preliminary to the definition of feedback in the context of the software process and to the development of a corresponding theory. The present paper will focus on one aspect of the phenomenology, the evolutionary behaviour of the global process as reflected in the growth of several industry developed systems[1] spanning a wide spectrum of application areas.

Representation and analysis of software processes, technical or global, as feedback systems is likely to prove difficult if not impossible. While the feedback concept of delay applies directly, that of gain must be reinterpreted. Broadly speaking it has to do with of the manner in which information is conveyed, the extent to which it is stressed, suppressed and/or embellished, how it is understood and interpreted by the receiver and so on. More general sources of the difficulty include the size and complexity of a continually changing feedback structure, the transient, possibly informal, nature of many of the paths, their transfer mechanisms and the sources of feedback feeding them and the major, possibly dominant, role and influence of people, individually and collectively. People observe, interpret observations, react, communicate, receive communication, interpret messages, inject or suppress message components on the basis of their own viewpoints, experience and interests, act or refrain from acting and so on. Information and control paths, whether forward or feedback, are noisy and not immediate candidates for formal analysis except when uncertainty can be explicitly considered, perhaps in a statistical sense or with a fuzzy set perspective [5]. Yet plots of software process metrics first produced and studied in the seventies displayed a regularity of short term patterns and long term trends that suggested disciplining forces at work. It was, in fact, these regularities that led to the formulation of the first five laws of software evolution [8, 9, 10].

Some years ago, one of the current authors asked himself why, despite the major investment in research effort the world over and despite major improvement in the technical methods, techniques and tools available for software development and maintenance it was so difficult to obtain major improvements in the global software process. Once asked, the answer was immediately self evident. As for any other multi loop feedback system externally visible improvement to that process requires, in general, that changes or addition to the forward path mechanisms are accompanied by adjustment and control of the feedback loop structure and mechanisms. The

overwhelming majority of research and development in software engineering has concentrated on the technical process, its component steps, languages, methods, techniques and tools. It has not addressed feedback aspects of the process or its consequences. It was this observation that led to formulation of the FEAST hypothesis, to the FEAST/1 project and, following conclusion of that project, to the renaming by us of the hypothesis as the eighth law of software evolution [14].

3 FEAST/1

The objectives of the FEAST/1 project [11] included identification of feedback effects in the evolution processes of several systems being evolved by the industrial collaborators, determination of their impact and the presence of system dynamics effects. The approaches employed included black box analysis of system evolution over a series of releases based on metric data obtained from historical data bases of those systems together with data on new releases as these became available. The dynamics of evolution were to be investigated through the development of system dynamics (SD) models [4, 3] of the global processes by which these systems were being evolved using the VENSIM tool [17]. Data for model construction, calibration and validation was to be, and was, obtained by discussion with the staff of the respective industrial collaborators and from the black box studies. The present paper concentrates on presenting that subset of the results of the study that address the specific issue of the presence and impact of the global process dynamics. Other results will be found in the references below, publications after 1996 in [6].

4 System Growth

The parameters of software system evolution and corresponding metrics are many. From the point of user, for example, measures of functionality, system capability or system power are of major interest. Other, interrelated, attributes of interest include reliability, complexity, ease of use, quality and cost. And there are many more. All are difficult to define. This paper concentrates on only one, the size of the system counted in modules, a metric that has proven useful over many years as a measure of system evolution. Modules, and related terms such as *files* and *holons* (ICL), have the advantage that they posses a degree of functional integrity, while their count is generally large enough to offer an adequate level of granularity and to permit statistical treatment. The lower level metric, *lines of code* (locs or equivalently, klocs), much beloved by industry, does not have semantic integrity in the context of system functionality. Moreover counts base on it are very much a matter of individual programmer taste and style. Counts of objects as in OO programming, possess semantic integrity but they and functional complexity are believed to be a poorer measures of system size or func-

tionality than modules. At the other extreme counts of sub systems, are much smaller than counts of the elements of which they are composed. In general, they are too coarse a metric to establish system size, functionality or power as the highest level representation of system evolution. It must be admitted that the choice of modules to measure system evolution deserves further debate. But consistent results obtained from its use over a period of almost thirty years, in the study of some nine systems, stemming from almost the same number of industrial sources suggests that the onus to identify a better measure lies with others.

The report that follows concentrates on four systems OS/360, A BAe defence system, the ICL VME operating system kernel and the Logica FW financial transaction system. The selection of these to illustrate the present paper is purely pragmatic since the remaining systems studied display essentially the same behaviour, though in some cases less clearly, because of less data available. Figure 1 shows the growth of these systems, expressed in the units used by the respective development organisation over the period of time for which data could be obtained.
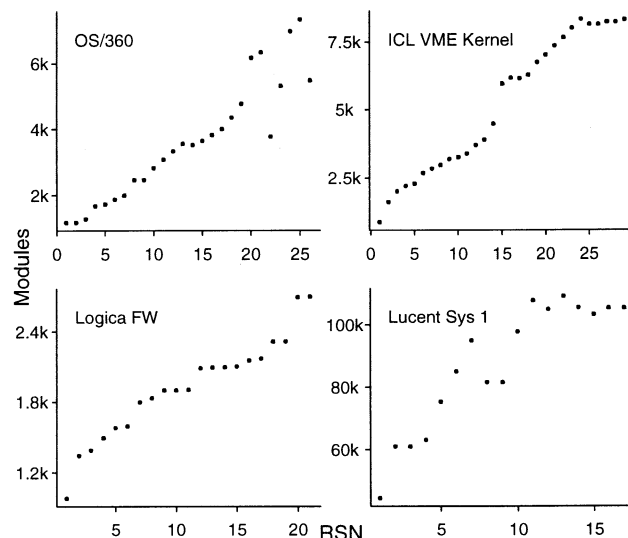

Figure 1: Software System growth.

Note that time metric use, *release sequence numbers* (RSN), measures pseudo-time rather than real time [2]. The underlying reason for its use is that the moment of release provides a well defined reference point to a system fully defined by its code and documentation, as released. At all other times the system is continually being changed. The use of actual release numbers is, however, not possible since each is, in effect, an arbitrary name. It may be noted that since it was first used in the 1970s

3

study of OS/360, the use of RSN has provided consistent and meaningful results that were smoother and more clearly interpretable than the equivalent based on real time. Its use leads, however, to one major problem that must be carefully and conscientiously solved. For a variety of reasons most full releases in industrial development are interspersed with smaller "decimal" releases. Some of these may be dedicated to fixing faults, others for making minor or additional enhancements or to release a change that was not ready to be included in the main release. Which decimal or minor releases are to be included in the evolutionary sequence whose growth rate or other characteristic is being investigated is a matter of judgment subject to many considerations. Exclusion of any implies that the integer sequence of RSN will represent a subset of actual releases. Their selection calls for great care with careful and precise definition of the criteria on which filtering is based. When such a subset has been plotted and analysed the existence of outliers or of gaps in the behavioural patter requires investigation of the circumstances surrounding the anomaly. Whether that anomaly must be regarded as just that or whether it is explained by circumstances lying outside the scope of the specific investigation, so indicating a need to drop a release from the selected set or to add one previously excluded, also demands conscientious investigation.

The figure reproduces the original OS/360 growth data and that of three of the five systems studied in the FEAST/1 project. These and the plots that follow, exemplify the FEAST/1 black box modelling outputs that are the sources of our interpretations. The plots suggest that the evolutionary behaviour of all nine systems display common *long term* features. They also all share a common *ripple* feature but differ in the detailed structure of the ripple. The present paper will concentrate on the long term trend and discussone of its implications. Note that that the VME plot suggests two distinct regions of evolution with a trend change in the interval between RSN 14 and RSN 15. Following discussion with and clarification by ICL personnel who were active in the process at that time, it is believed that the change is due to changes in release policy and other marketing factors. This will be further investigated if and when the FEAST/2 [13] project gets underway.

5 System Growth Models
In a paper published shortly before the start of FEAST/1 [16] Turski discussed the evolution of the Logica FW system. *Inter alia*, he suggested that its growth could be closely represented by an inverse square model. Such a model is conceptually satisfactory because it is consistent with the complexity constrained growth implied by the second law of software evolution [8, 9, 14]. The model together with details of its estimation are

provided an appendix to this paper. Figure 2 illustrates the closeness of fit of inverse square models to three of the four systems being discussed in the present paper. Reappearance of the same form when calibrating the first FEAST/1 white box systems dynamics model [18] provide further support for the view that complexity growth is a limiting constraint on system evolution. As indicated below OS/360 growth appears to be more appropriately modelled by linear growth, a deviation from the common pattern that must be further investigated.
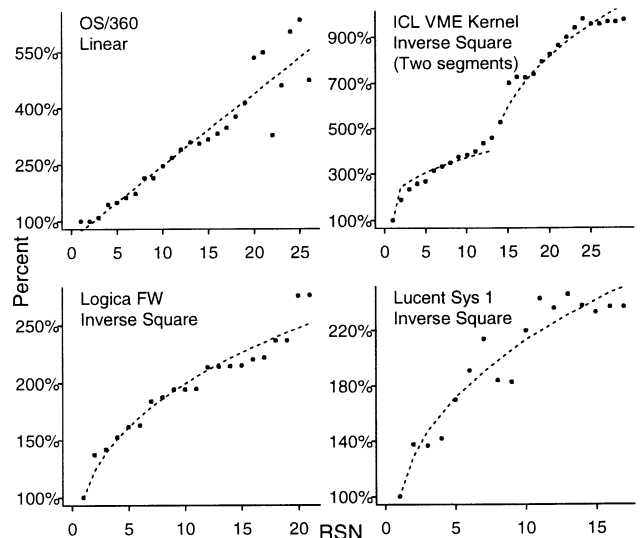

Figure 2: Models of Growth

Visual inspection provides a coarse indicator of goodness of fit of a numeric model. Mean absolute percentage error (MAPE) (appendix) and the standard deviation of percentage residuals are better, statistically derived, indicators. Notice that both indicators refer to percentages rather than absolute values, necessary because the concern here is with a growing system. Table 1 shows each of the indicators for all the systems discussed in the present paper. Note that for OS/360 the fit for both linear and inverse square models is provided and clearly show that the former is to be preferred. For reasons which are well understood [10] and as visible in figure 1, the transition in that system, from RSN 19 to RSN 20 and beyond was the start of a period of rapid change and oscillation. The table includes, therefore, separate fit indicators up to RSN 19 and over the full range up to RSN 26. For VME Kernel the indicators have been computed separately for each of the two segments briefly introduced above.

6 The Models as Predictors of Future Growth
The models as illustrated above and as described in the appendix are clearly good models of the growth of each

4

System	Model	MAPE	St.Dev. of % Residuals
OS/360 - RSN 1 - 19	Linear	4.7	5.9
OS/360 - RSN 1 - 26	Linear	9.6	14.0
OS/360 - RSN 1 - 19	Inv.Sq.	20.1	25.0
OS/360 - RSN 1 - 26	Inv.Sq.	30.9	36.7
VME Kernel (Seg.1) - RSN 1 - 13	Inv.Sq.	8.0	11.4
VME Kernel (Seg. 2) - RSN 14 - 26	Inv.Sq.	3.7	5.1
FW	Linear	3.6	4.8
Lucent Sys 1	Inv.Sq.	6.0	7.2

Table 1: Goodness of Fit of Growth Models over the Entire Release Set

Figure 3: Evidence for Dynamics Determined Software System Growth Trend.

of the systems, despite the fact that such evolution is planned by humans, generally on a release by release basis. One must then ask, how good is the model as a predictor. Turski already asked this question in respect of the Logica FW system [16] and his approached has been broadened and applied to all the systems under investigation in FEAST/1.

The models shown in figure 2 have been estimated as outlined in the appendix by using the full data set available. One may ask, "how many data points are required to obtain an acceptable fit and to yield a satisfactory predictor for future growth"? The answer to this question may be obtained by using less than the full set to estimate the model and computing the mean percentage prediction error over the remainder, varying the number of points used from a minimum of two to a maximum which leaves releases whose size may be "predicted". One may then plot the MAPE as a function of the number of points used to estimate the model. The results for for all the systems studied are illustrated by the plots of figure 3. The significance of these plots in illustrating the predictive power of the models is summarised in Table 2.

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

Table 2: Average and Standard Deviation of Mean Absolute Percentage Error MAPE after its Stabilisation

The plots and the derived statistical indicators give a clear answer. A handful of data points, 3 to 6 in this case
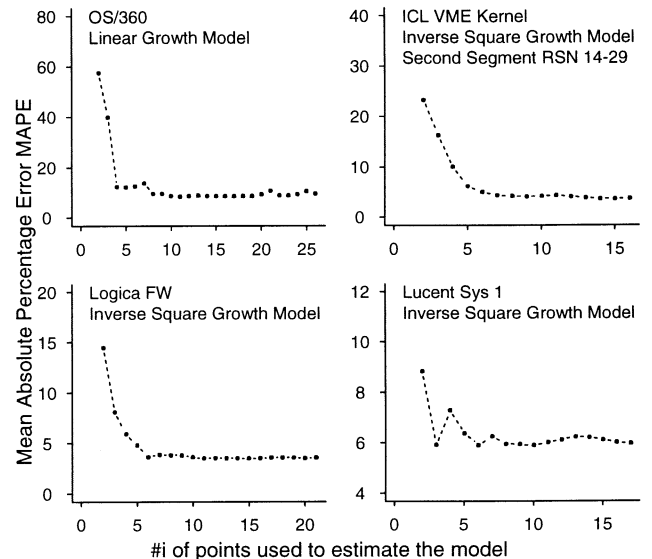
suffice to give a model that is an astonishingly good predictor of future average growth. A model derived from that small number of releases has more than adequate precision to predict system growth over many years[2], far greater than what one would expect from system evolution and a growth process planned and executed by people. Six releases appear to suffice to establish the growth trend. No significant additional precision in predictive power is gained by computing the model from more data points.

7 System Dynamics

When this result was first obtained [16] it was interpreted as a reflection of the dynamics of the feedback based evolution process. Thus it was seen as the first direct evidence in support of the FEAST hypothesis and of a remark made by one of the present authors to a Senior Vice President of his then employers, "The problem is that you managers think that you are in control of the system but actually the system is controlling you". The plots appeared to indicate that it took several releases for the process dynamics to establish itself; but once established it determined the basic long term (in this example) growth trend though management decisions and plans could could, and in practice did, cause local variations. Feedback determined self stabilisation ensured maintenance of some trend, determined by the dynamic properties of the process as a feedback system. The result was also seen as providing strong support for the laws of software evolution all of which had long

[2]The evolution periods spanned by the systems discussed here all exceed 15 years.

been seen as reflecting the feedback system nature of the software process [14].

After some time two problems were recognised in this interpretation. One related to the remarkable similarity between the behaviour of the systems. The second surfaced when it was realised that the history of FW did not begin with RSN 1. The release given that identity was, in fact, the fourth of a sequence of main releases. There was a prior history to the history being studied. Similar the so called Lucent System 1 was in fact the successor to several earlier systems each of which had evolved over many years. Thus the simple interpretation of a process dynamics establishing itself and reflected in the plots of figure 3 could not be maintained. Instead, it was realsied that the initial knee visible in all the plots was an artifact of the process estimation process, reflecting the convergence of the parameter(s) of the models, by whatever means estimated, to establish themselves. This interpretation was particularly attractive since it explained why, despite major differences between the various systems and the processes by which they were being evolved, the plots were so strikingly similar at the highest level. After all, the same procedures and algorithms were being used to estimate them.

This interpretation explained the initial knee. The stronglong term growth trend as reflected in the predictive power of the models must still be explained. The answer to this was immediately self evident and reflected a previously held conviction as implied by the discussion in section 1 above. From its conception, the FEAST hypothesis, the view of the software process as a feedback system, has asserted that the externally visible attributes of software processes are much influenced, even determined, by factors other than the methods, tools and techniques used in any specific technical process of software development. They even extend to influences outside the immediate organisation directly responsible for product evolution. Equally there will be influences that precede that process in time. Process, management, implementation procedures and controls for example will have been inherited from past projects and practice. For any other than a totally new organisation, general corporate practice will have been long established, informally or formally. Equally most organisations will have a traditional software process, may even dictate a generic or prototype process. Finally, all organisations operate in a, forever changing, domain, are subjec t to strong external forces, client needs, market forces, economic and political circumstances and so on. All represent global forces that will directly impact a new process initially, thereafter through feedback controls, checks and balances, pressure points of various forms for various purposes. That such forces are present cannot be denied. It appears likely that for any process

a major, possibly dominant, component of the project dynamics will have been inherited.

It is this analysis taken to its logical conclusion that underlies the phenomenology described in section 1. It supports the FEAST hypothesis that in modelling and seeking to improve the software process one must consider the **global process**, its attributes and its structures. The interpretation of the plots of figure 3 and of the long term evolutionary behaviour of all the systems is now clear. They reflect the dynamics of the global project, the dynamics of the developing, marketing and user organisations in their entirety, that is in extent and in history. There may even be influences of external economic and societal factors though the last two are likely to be occasional rather than continuous and will not be further considered here.

8 The Presence and Influence of Global Dynamics

To confirm this interpretation the model parameters were recomputed for a varying number of data points as for the plots of figure 3 but shifting the starting point to RSN m, that is ignoring $m-1$ earlier releases. The data points for releases preceding RSN m were discarded, thereby creating an additional, albeit more recent, inherited "history". Many alternative derivations can be made. Such an experiment has at least four parameters, the algorithm used for estimating the model parameter(s), the starting point, the interval between starting points, and the number of data points or size of window over which the forecasting accuracy of the model is evaluated (using MAPE, for example) This analysis and interpretation of the results is still underway (at the time of writing - August 1998) but what is important is that the many results to date all point to the same conclusion, that subject to perturbations introduced by local deviation from the model value at the starting point, a knee plot is obtained that has the same general characteristics as the plots of figure 3. For the purposes of this paper, eight plots typical of all the results so far obtained, are reproduced (figures 4 to 7). The MAPE of each model is calculated over the releases following those used for parameter estimation. For each system two plots are presented. One plots the MAPE for parameters computed for models estimated with RSN 1 as starting point, RSN 2, RSN 3, RSN 4 and RSN 5, respectively. Each of the five curves is identified by a starting point number. The starting point is shown on the plot as a star. The second plots the MAPE for models starting at other RSNs, as shown. The RSN 1 curve appears in both plots to facilitate comparison.

The fact that, apart from an increase in deviation over the last two or three point for the late starting sequences ($m = 13$) these plots confirm both the initial knee believed to reflect the model start and the essentially con-
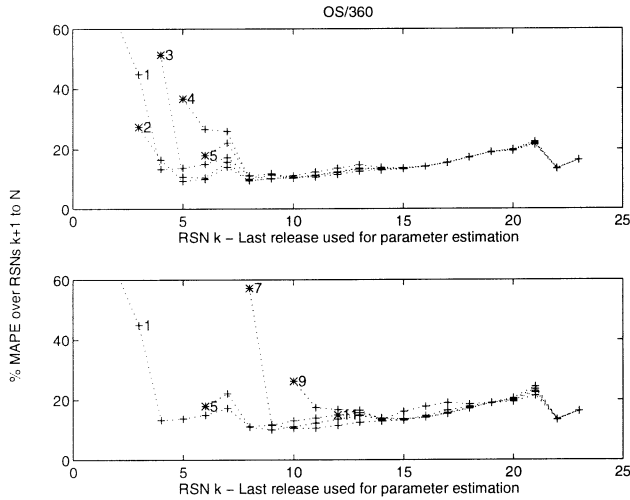
Figure 4: Model Build Up and the Initial Case for Inherited Dynamics. System: OS/360. Growth Model: Linear.

Figure 5: Model Build Up and the Initial Case for Inherited Dynamics(cont). System: ICL VME Kernel. Growth Model: Inverse Square.

stant error thereafter, is interpreted as an indicator of the influence of the inherited dynamics. Further work is clearly necessary but the accumulated results to date are believed to constitute a *prima facie* case for the strong influence of what has been termed *global factors* on system evolution.

9   System Dynamic Models

To this point the paper has focused on the support derived for the FEAST hypothesis in general, presenting some results of a metric based black box analysis indicating, *inter alia*, the presence, influence and power of global dynamics. The FEAST/1 investigation also included a white box modelling effort to study process dynamics. The paradigm and tool used [17] stress the importance of feedback structures and flows in simulating the dynamic behaviours of systems. Five related principles were applied in developing the models; that the modelling effort should proceed top down, that model detail should be added only as required to reflect observed behaviour, that such model refinement should reflect likely sources of that behaviour, that the amount of detail added should be kept to the minimum required to reproduce the behaviour being modelled and that model validation and calibration should involve the development organisation and team.

Two models based on these principles have recently been described [12, 18]. The results of calibrating and running them demonstrate that they are able to replicate actual evolutionary trends such as system growth and effort expended. In the context of this paper, they provide further support for the thesis that process be-
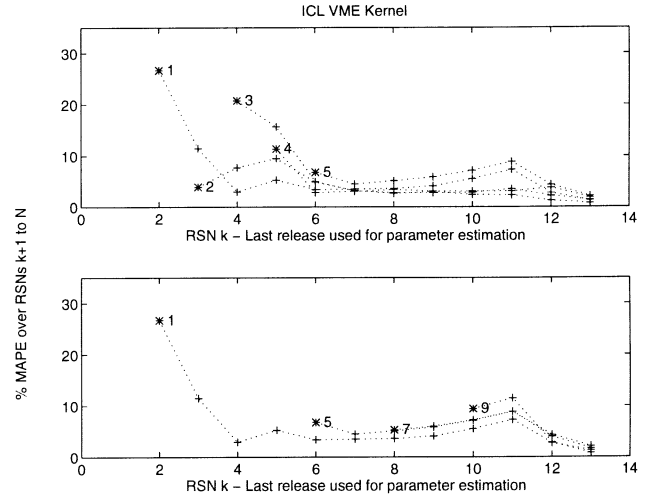
haviour is, at least, strongly influenced by factors external to the technical development process. Figures 8 and 9 reproduce the models and figures 10 and 11 show, respectively, an output from each.

The simplicity of the models is self evident, indicating that models reflecting high level concepts and global factors can capture long term system behaviour. Their outputs as in figure 7 provide further support for the hypothesis that feedback-driven dynamic forces inherited from outside the technical process can have a dominating influence on long-term product evolution. The faithful reproduction of long term process behaviour that the models show supports the conclusion reached from the phenomenological analysis and the black box analysis outlined in the previous section. It remains to be demonstrated that reproduction of the behaviour is due to the model reflecting the properties of the real world process behaviour and is not just a reflection of the calibration process. However, the organisations evolving the systems and we are confident that the models and their calibration parameters reflect actual mechanisms, influences and behavioural values. The models have already proved their value to the industrial collaborators, helping them understand and, potentially, improve their system development activities.

10   Conclusions

Much remains to be done, continuing the work started in the FEAST/1 project, to derive practical implications that can be put to good use by our industrial collaborators and by industry at large and to create , at least, a base and framework for a theory of software evolution
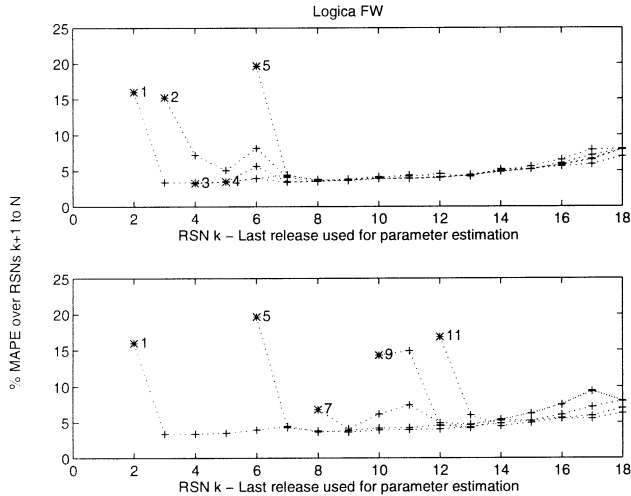
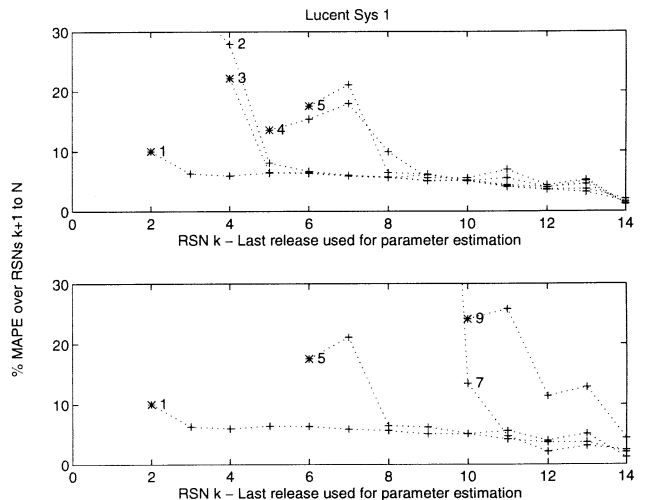Figure 6: Model Build Up and the Initial Case for Inherited Dynamics(cont). System: Logica FW. Growth Model: Inverse Square.

Figure 7: Model Build Up and the Initial Case for Inherited Dynamics (cont). System: Lucent Sys 1. Growth Model: Inverse Square.

and the software process. It is hoped to continue that work in FEAST/2 [13].

In the context of this paper several challenges may be mentioned. A fuller listing is provided in the FEAST/2 proposal [13]. Much of the work briefly described here needs to be extended and its practical consequences derived. To explore the degree of influence of global factors on system evolution different sources and their significance should be identified, explored and assessed. This may be achieved via the black box approach by obtaining further data from different systems from the organisations who have already contributed data. In this way it may be possible to eliminate organisational influences. If the new data comes from a system addressing a different application and/or user domain, their influence may be identifiable or eliminated. The hope is to identify the nature and extent of any actual project influence. The refinement of the system dynamics models, their extension and the development of new models will also help in this respect.

Acknowledgements

Appendix
Inverse Square Model [16]: S_i, the size predicted by the inverse square system growth model at release i, is $S_i = S_{i-1} + \hat{E}/S_{i-1}^2$ $(2 \le i \le N)$ where N is the total number of releases. S_1 is assigned the actual size of release 1. $\hat{E}$ may be defined as the average E_i, where $E_i = y_{i-1}^2(y_i - y_{i-1})$ $(2 \le i \le j)$, y_i is the actual size of release i and j is the number of releases used to estimate the model.

More generally $\hat{E}$ can be obtained numerically based on the minimisation of some function, such as the sum of squares of the residuals, that is, of the errors of fit.

Mean Absolute Percentage Error MAPE as plotted in figure 3: For a growth model based on the first j releases only, MAPE(j), is calculated as 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.

Mean Absolute Percentage Error MAPE as plotted in figures 4, 5, 6 and 7: A growth model estimated based on releases m to k (defined as (m,k) model) ignores $m-1$ earlier releases. MAPE for a model (m,k), MAPE(m,k) is obtained from MAPE(m,k) = $\frac{100}{(N-k+1)} \sum_{i=k+1}^{N} \frac{\|S_i(m,k)-y_i\|}{y_i}$, where $S_i(m,k)$ is the predicted size at release i using the (m,k) model, y_i is the actual size at release i and N is the total number of releases.

$\hat{E}$ and MAPE estimations shown in all the plots were obtained using MATLAB [15] scripts. A MATLAB script for the calculation of $\hat{E}$ is available via the FEAST/1 web site [6].

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
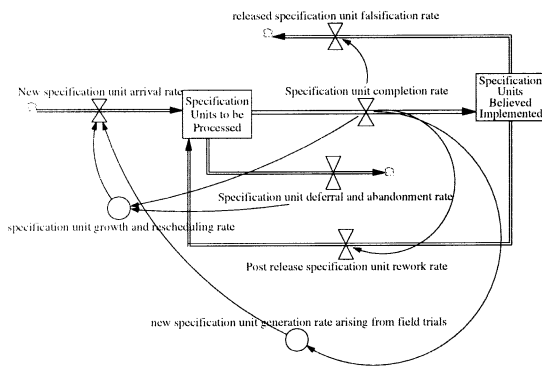
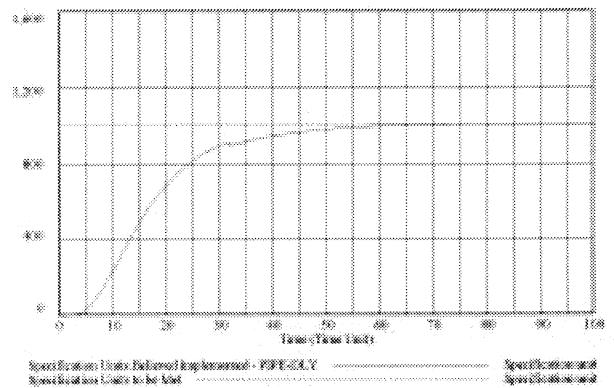Figure 8: System Dynamics Model of the BAe System.

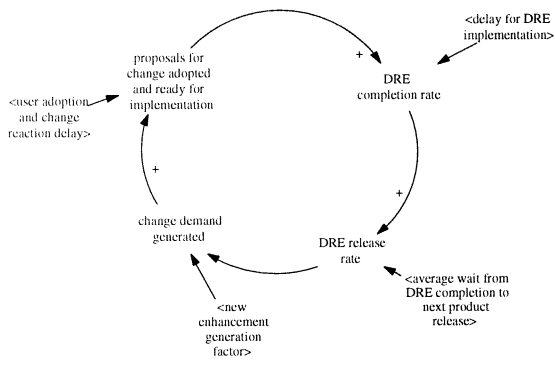Figure 10: An Output from the 'BAe System' Model

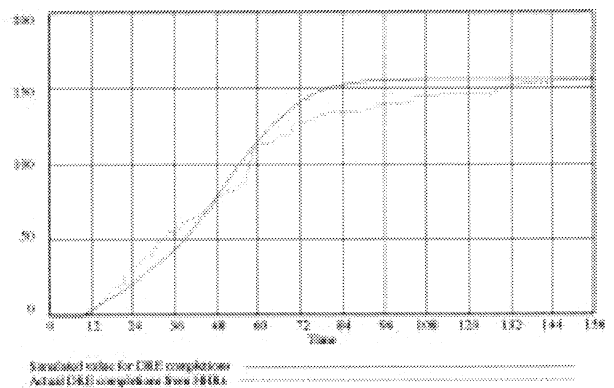Figure 9: System Dynamics Model of the ICL VME Kernel System.

Figure 11: An Output from the 'ICL VME Kernel' Model

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

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

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

[5] Friedman Y and Sandler U, *Evolution of Systems Under Fuzzy Dynamic Laws,* Fuzzy Sets and Systems, Vol. 84, 1996, pp. 61 – 74

[6] http://www-dse.doc.ic.ac.uk/~mml/FEAST1

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

[8] id., *Laws of Program Evolution - Rules and Tools for Programming Management,* Proc. Infotech State of the Art Conference, Why Software Projects Fail, April 9-11 1978, pp. 11/1 – 11/25

[9] id., *Programs, Life Cycles and Laws of Software Evolution,* Proc. IEEE Special Issue on Software Engineering, v. 68, n. 9, Sept. 1980, pp. 1060 – 1076

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

[11] Lehman MM and Stenning V, *FEAST/1 - Feedback, Evolution and Software Technology: Case for Support,* EPSRC Research Proposal, Nov.1995/March 1996, 11 p.

[12] Lehman MM and Wernick PD, *System Dynamics*

Models of Software Evolution Processes, Proc. Int. Workshop on the Principles of Software Evolution, ICSE 98, Kyoto, Japan, April 20 -21, 1998, pp. 6 – 10

[13] Lehman MM, *FEAST/2 - Feedback, Evolution and Software Technology: Case for Support,* EPSRC Research Proposal, July 1998, 11 p.

[14] Lehman MM, Perry DE and Ramil JF, *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution,*to appear in Proc. Fifth Int. Symp. on Softw. Metrics, Metrics 98, Nov. 20-21, Bethesda, Maryland

[15] *MATLAB High-performance Numeric Computation and Visualisation Software - Reference Guide,* The MathWorks, Inc., Natick, MA, 1992

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

[17] *Vensim Reference Manual,* version 1.62, Ventana Sys. Inc., Belmont, MA, 1995

[18] 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