

Feedback, Evolution and Software Technology - The Human Dimension

ICSE 20 Workshop
Human Dimensions in Successful Software Development
Kyoto, Japan

20 April 1998

M M Lehman
Department of Computing
Imperial College
London SW7 2BZ
tel: +44 (0)171 594 8214
fax: +44 (0)171 594 8215
mml@doc.ic.ac.uk
<http://www-dse.doc.ic.ac.uk/~mml/>

Abstract

A 1968 study of the software process led, *inter alia*, to the observation that the software process constitutes a feedback system. Attempts at its management and improvement must take this basic fact into consideration if optimum results are to be obtained. A project, FEAST/1, was initiated in 1996 to examine the assertion. Its objective is to identify feedback mechanisms and controls in the software evolution processes of several industrial collaborators, assess their impact and examine their role and potential in process improvement. The approach being used in the study is to construct black box, white box (System Dynamics) and other classes of process models of selected collaborator systems on the basis of the available process evolution metrics and other global data about process components and structures.

The feedback that exists in these processes involves many humans in many roles. These observe, interpret, communicate, listen and act (or not). Global models of their activities cannot, however, in general, reflect the actions or behaviours of individuals. The level of detail which global process models can reflect is, therefore, limited.

As a complex multi level multi loop feedback system, the long term behavioural patterns and trends of software processes are largely determined by its internal dynamics which, in turn is feedback generated and controlled. Process models must, therefore, reflect the feedback phenomenon. Individual human assumption and decision plays a central role in the feedback processes and this must be taken into account when considering the human dimension, the role and impact of people, in and on the process. Individually this cannot be done, but the aggregate effect of many individual decisions by many people may, in general, be represented by statistical means. One constructs, therefore, meaningful models relating to global process behaviour over time without addressing individual behaviour. In practice, this is not a significant limitation since individual actions have, in general, only local, impact on the long term performance of the global process.

The paper includes brief mention of FEAST/1 results to date and provides literature references for further detail.

Keywords: Software Evolution, Software Process, Feedback, Feedback Control, Human Role, Process Improvement

1 Overview

More than forty years of universal experience has shown that *E*-type (real world) [leh80b] software must undergo continuing change to remain effective. It must continually be evolved to satisfy the conditions, needs and operational requirements of a changing environment at each moment in time. Software that is used and not evolved becomes progressively less useful. The evolution process is executed to develop software in the first place and, subsequently, to maintain it satisfactory and of adequate quality.

The degree to which user satisfaction can be maintained depends critically on the detailed nature and quality of the evolution process. The make up of that process is crucial if satisfactory properties are to be achieved in the first place and maintained under continuing change. It has, unquestionably, been considerably improved over a period of almost forty years but still does not meet the need. A recently formulated hypothesis [leh94] suggested that *feedback* may be the common factor to software processes that explains why process innovation has had only limited impact *at the global level*. The feedback nature of software processes and its consequences have direct implications for all developers of *E*-type software. It must be noted that people play a critical role in all the feedback paths and mechanisms. Their behaviour (observation, interpretation, transmission, action, non-action, etc), individually, in groups and collectively, necessarily affects process behaviour and will, eventually need to be considered. It has not, in general, been addressed in software process studies. It is now being implicitly addressed in the FEAST/1 study [leh96c] now underway.

Recognition of the feedback phenomenon suggests a new dimension for *process improvement*. Substantiating the hypothesis and achieving its successful exploitation represents a major challenge to software engineering and many other areas. FEAST/1 is addressing these issues in collaboration with major industrial organisations¹ whose processes are being studied, modelled, analysed and are, eventually, to be improved. Despite the fact that the role of individual humans in the feedback mechanisms is not being specifically considered at this stage significant results are being obtained. The results achieved to date will be summarised. It is hoped that a feedback based process improvement technology, the impact of humans and a general theory of software process, software evolution will eventually emerge.

2 Feedback, FEAST and FEAST/1

In a 1972 paper [bel72] discussing some results of a study of the IBM programming process [leh69] it was observed that the ripple in a plot of the growth of IBM's OS/360 operating system, "... is typical of a self stabilising process with positive and negative feedback loops. ... the rate of system growth is self-regulatory, despite the fact that many different causes control the selection of work implemented in each release, with varying budgets, increasing numbers of users reporting faults or desiring new capability, varying management attitudes towards system enhancement, changing release intervals and improving methods ...". It was the observations made during this initial study and the analysis and interpretation that followed that led to the formulation of eight laws of software evolution [leh74,80,96a], a Principle of Uncertainty [leh89] and, in general, to a deeper understanding of the nature and phenomenology of software evolution. Earlier it had been suggested [leh78] that "one must ... regard the organisation developing and maintaining a large program as a system in the system theoretic sense. Observation has shown that the system behaves as a self stabilising feedback system. The process leads to an organisation and a process dominated by feedback ... with long range trends ... and invariances ...". And that was how the matter was left.

Recently [leh94] it was recognised that the first seven laws followed from the eighth law which states that *E*-type processes are feedback systems [leh96a]. That law could also explain the universally experienced difficulty in achieving major improvement in the global software process; that is the total process whereby software systems are developed, used and evolved [leh96d]. This observation was expressed as the FEAST hypothesis and led eventually to the setting up of the FEAST/1 project. The project objectives included identification of significant feedback controls in the processes of industrial collaborators, modelling them and determining their impact and exploitation of the resultant insight and any tools developed to improve these processes [leh96c].

The study has proven even more difficult than was initially foreseen. Some obstacles were specifically anticipated, for example, as discussed in [leh96b] and by the initial decision to develop

¹Matra - British Aerospace, ICL, IST, Logica, Lucent Technologies, UK MoD - DERA

system dynamics (SD see sect. 3) [abd91] models top down rather than bottom up; the approach taken by other SD modellers [mad96]. Other potential problems were recognised but underestimated. The latter included that of retrieving data for modelling from the collaborators' archives, cleansing and interpretation of retrieved data, the difficulty of statistical inference from relatively small (10 to 30 point) time series data sets and that of applying SD technology and tools to produce models reflecting total organisational processes (including user feedback if appropriate) that can be interpreted and calibrated to reproduce the global behaviour and trends revealed by real world data.

3. Modelling Approaches

The original proposal [leh96c] envisaged just two approaches. Black box models of the evolution of one system per collaborator were to be visually interpreted and statistically evaluated. They were intended to identify patterns or regularity in short and long term trends of individual systems, similarity or commonality of behaviour across several or all systems, evidence of feedback-like behaviour (eg self stabilisation) and the degree to which the data and interpretation of the observed behaviour supports, modifies or contradicts the laws previously formulated. In parallel, white box models were to be developed and calibrated on the basis of discussions with development staff and other collaborator personnel using the SD approach and the Vensim modelling tool [ven95]. The objective was to reveal the internal structure of the processes being studied in the black box work, the activities involved and replication in the SD models of the behaviour displayed by the black box models. This should permit progressive identification of the real world feedback mechanisms and controls contributing to produce the characteristic behaviour of each process and provide an indication of their impact.

As the work proceeded, and with the fortuitous availability of a PhD candidate interested in multi-agent modelling, an effort was initiated to produce such models and to calibrate them to reproduce the same behavioural patterns. This work was seen as providing an independent validation tool for the white box models. More recently the phenomenological imprecision of evolution data and some of the process attributes which appear to underlie it, suggested that Fuzzy Dynamics [lev90, fri98] techniques, and models derived therefrom may prove a powerful technique for advancing the study. Some small effort is being devoted to following through this approach but early results are not expected and must await a further project and access to the appropriate expertise.

4 Results to date

Recent publications have discussed black box results obtained to date, [tur96, leh96a, d, 97] for example. These will not be discussed again in detail in the present paper. Suffice it to say that strong similarities may be observed in the evolutionary behaviour, patterns and trends of the systems currently being studied² as exemplified by the plot of the growth of a very large Lucent Technologies' real time system in fig. 1 and of modules added per release of the ICL VME Kernel in fig. 2. The reader may wish to compare these plots with those in earlier publications [leh80, leh96a, d, 97].

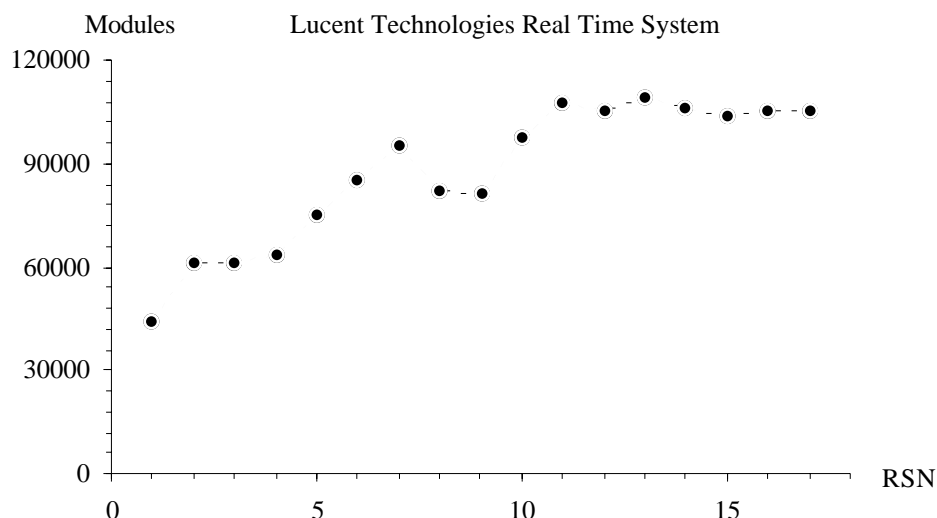


Figure 1. Modules Added per Release - where RSN is release sequence number

² BAe ASRAAM, ICL VME Kernel, IBM OS/360, IST X-Designer, Logica Fast Wire, Lucent Technologies rt system
31/3/98, 2:06 pm

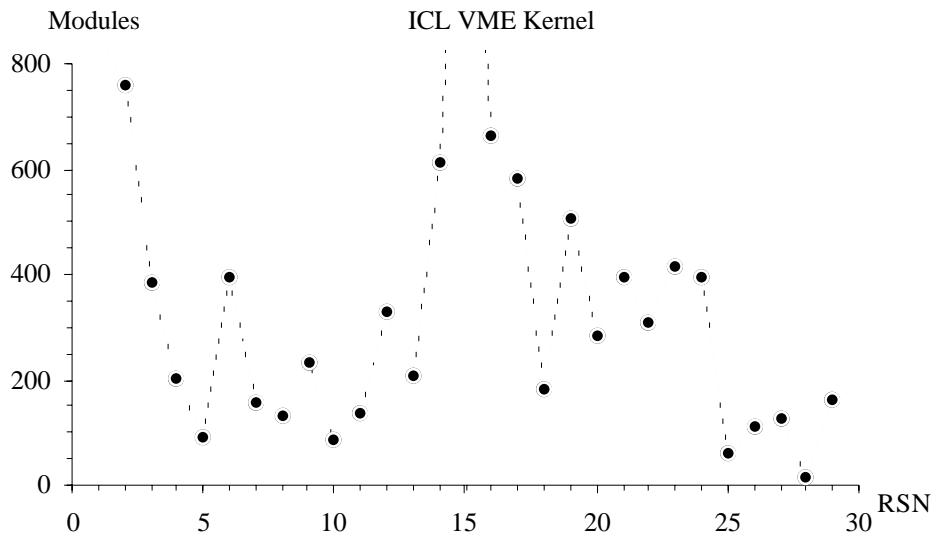


Figure 2. Modules Added per Release

Many of the behavioural patterns observed in all the parameters studied are essentially similar to those identified in the 1970s and encapsulated in the laws of software evolution [leh80b,96a,d,97a]. The present studies do, however, suggest minor refinements of the latter.

In considering these plots, a key issue is to determine whether the observed patterns and behaviours are, or are not, random like; whether or not they indicate the presence of underlying structures. Indications of the latter at an acceptable confidence level would provide support for and confidence in the FEAST hypothesis that: *as complex feedback systems, E-type software processes evolve strong system dynamics and with it the global stability characteristics of other feedback systems*. This evaluation is currently underway. Early results indicate non-random behaviour, correlations and auto correlations. This suggests the presence of underlying stabilising and control mechanisms, possibly the feedback phenomenon addressed by the FEAST hypothesis. Positive results also provide support for the Laws on Software Evolution [leh74,78,80] and so significantly advance development of a theoretical base and framework for software engineering process design, planning and management.

Recent results have also provided indicators of the presence and strength of an internal dynamics that influences long term behaviour. This is exemplified by the inverse square models fitted to growth data of the Logica FW [tur96,leh96d] and Lucent systems. The closeness of fit of the models and the strength of their growth dynamics is illustrated in fig. 3a and 3b respectively. Data from six releases suffices to ensure an error in FW growth predictions that is essentially constant (and small). The Lucent system displays even stronger dynamics with two to three releases being sufficient to determine the model parameter. In either case, increasing the number of data points used to estimate the latter produces no significant reduction in mean error, the precision of the model as a predictor.

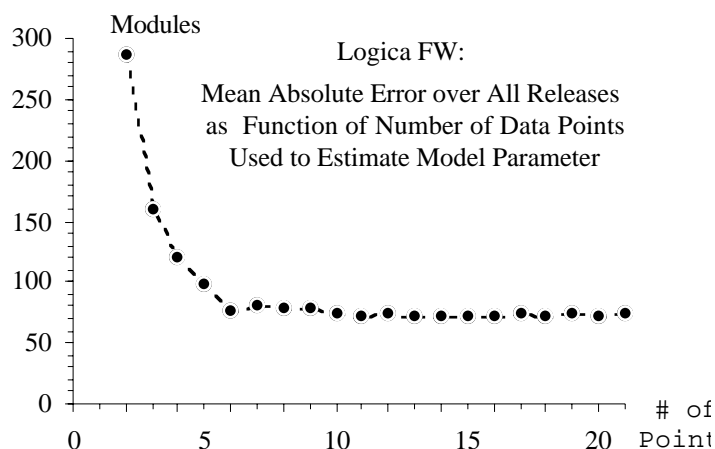


Figure 3a. Mean error over all releases of the size predicted by an inverse square model as a function of the number of data points used to estimate the single model parameter

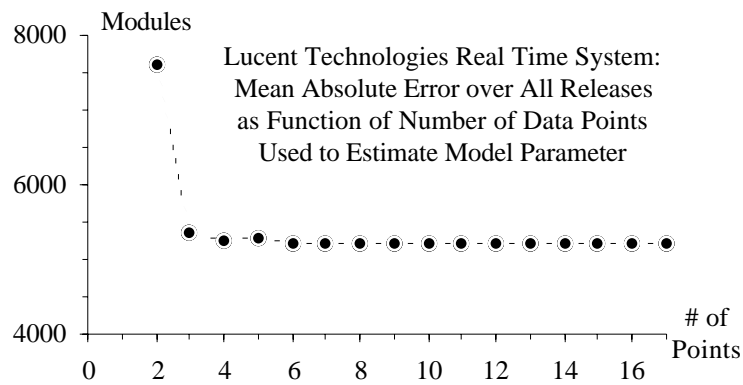


Figure 3b. Mean error over all releases of the size predicted by inverse square model as a function of the number of data points used to estimate the single model parameter

The inverse square model has also been successfully fitted to several other of the trend plots and has provided a calibration parameter in the first of the system dynamics models. While the point has not yet been conclusively proven, this repeated appearance of the inverse square pattern is significant, suggesting as it does, increasing system complexity and/or feedback control as significant factors in the processes studied and in their behavioural patterns.

Progress in the development of the white box models was, initially slower than had been planned. In the first place none of the three full time FEAST/1 team members had prior experience of System Dynamics techniques or tools. Moreover it had not been fully appreciated that in addition to absorbing the concepts and approaches of SD analysis it was desirable to transform many of the SD concepts and modelling approaches to reflect a semantics that is more appropriate to the software process. Finally, despite the adoption of a top down approach to model construction with the intent to achieve step by step refinement (as expressed in next paragraph), the early models quickly became very complex, involving many tens of *levels* and *flows*. These models were difficult to comprehend as a whole, reflecting specific understanding of process minutiae. Their construction provided a learning experience, initial clarification and a theoretical basis for discussing the nature of the processes modelled. But their intrinsic contribution to advancing the project towards its goals was relatively small. Functional and structural validation through exposure to and discussion with the respective industrial groups, of their structural and functional content was difficult and, hence, slow and related to detail rather than global issues. Their *meaningful* calibration appeared a daunting, possibly impossible, task. Thus despite their providing an experience base and in clarifying top level issues these early models were abandoned.

The models that followed are of two types, both high level. One we term a *production line* model, the other a *gap* model. Both approaches, as described elsewhere [leh98,wer98], are meaningful and relevant and it is expected that eventually they may be integrated. The guiding principle in their development was that each should be developed top down, should initially involve only the absolute minimum of elements, must be phenomenologically interpretable and its behaviour validated with respect to project behaviour. In adopting this top down approach we were conscious of following the approach first advocated by Zurcher and Randell in 1968 [zur68]; a precursor to Wirth's Stepwise Refinement [wir71] and Dijkstra's Structured Programming [dij72]. Thus the respective models are being developed and elaborated, step by step, in discussion with the development team, progressively calibrated to achieve broad brush agreement with the observed behaviour as measured by appropriate metrics and systematically refined, in coverage and detail. Significant progress has been made as we have undertaken cycles of discussion and calibration of the resultant models and assessed their relevance and contribution to our understanding and the FEAST/1 study.

At the time of writing the first SD simulation model has, with the aid of BAe staff, undergone top level calibration. In execution, the model closely follows the long term trajectory of actual project behaviour. The next development step is to add further model elements that reflect the next level down of process attributes, the object being to achieve a model that displays the short term variances (higher frequency components) that appear on the actual evolutionary trajectory. The ICL VME models, too, are close to being ready for calibration.

5 Software and Systems Processes

The previous sections have provided an overview of the background to the FEAST/1 project. This is examining software evolution data, metric and otherwise, seeking and evaluating supporting behavioural evidence from process and product metrics for the thesis that the software process behaves as a feedback system and constructing executable models that reproduce the observed behavioural patterns.

Feedback control mechanisms are to be identified, their impact determined and means devised for managing and improving the processes. For detailed discussion of the results obtained to date the reader is referred to the referenced literature. The present section will go on to identify essential steps in the realisation of the *ab initio* development of a software system in the context of some application or in the application of changes to such a system to achieve a new, improved, release. In either case the work to be undertaken will have arisen from a stated or implied an application concept. The resultant activities may be conveniently displayed in diagrammatic form to illustrate the evolution process. There is nothing sacrosanct in the descriptions that follow but they are believed to include all the basic steps required to transform an application concept into a satisfactory operational system or a system change definition into a new release of an existing system.

The real world domain in which an *E*-type system operates is unbounded. The software that implements an application within that domain is, of necessity, bounded. Thus the first step in any evolution step of an *E*-type system, whether *ab initio* or by changing it in some way, must be to bound the application and its operational domain or the changes and adjustments to the domain. Such bounding implies the essential incompleteness of the software system as a model of the application in its real world domain; a gap between the finite operational system and the actual (rather than adopted) unbounded real world domain. That gap is bridged by making assumptions which are ultimately reflected in design, implementation and operational decisions. They are deeply embedded within the system. The assumption and bounding processes involve such varied stake holders as clients, users, management, marketeers, technical staff. The views of all on both technical and pragmatic issues, must be elicited, reconciled, merged and limited to provide a system definition that meets acceptable functional, quality, performance and cost objectives that can be implemented, delivered and introduced into service within the required time frame. A process of successive elucidation blends the many viewpoints to provide a starting point for the development and subsequent evolution of the software system. The process is essentially iterative and, as pictured in fig. 4, as it converges the bounds of both the application and its operational domain change.

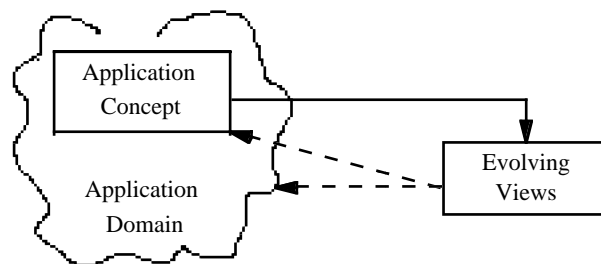


Figure 4. Defining the Application and its Operational Domain

This iterative first step of the process involves feedback. Control is applied at various technical and management levels to ensure that, as far as can be foreseen, the final system as delivered will have the desired and required properties and attributes as now foreseen; that it will be satisfactory and acceptable to the stake holders and the user community. Unfortunately, however, the real world is always changing, often in unexpected ways. It is, therefore, unlikely that the system will provide a perfect fit to the operational domain as at the time of installation and during subsequent operation.

Once the bounding process has reached a stage where, for the time being, the application and domain views appear to have been satisfactorily established, further evolution steps are initiated. Fig. 5 illustrates a possible sequence of activities to the point where a validated system has been completed and is ready for installation. As already stated, the steps shown exemplify the information that must be generated and agreed to satisfy the product evolution goals. They are not intended to establish a definitive process model.

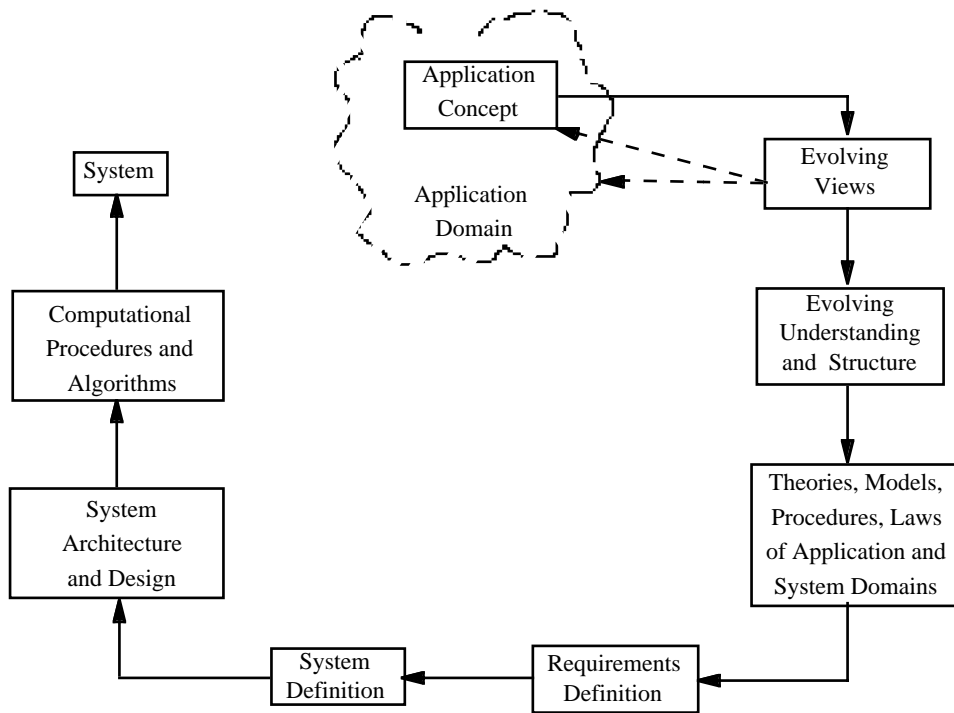


Figure 5. System Development

The end product of this process is (or should be) a validated system ready for installation and operation in the application domain as in fig. 6. Such installation changes the operational domain. The domain with a system installed and operational is different to that before installation. Installation also changes the application. Apart from the additional activity arising from operation of the system, there will be changes in the activities associated with the application. If that were not so, there would be no point in installing the system. Thus the application and its intended operational domain that provided the initial input to the development process is changed by the output of that process. At its primary technical level the software evolution process, whether executed for an *ab initio* development or to implement a system change or enhancement, is a closed loop feedback system.

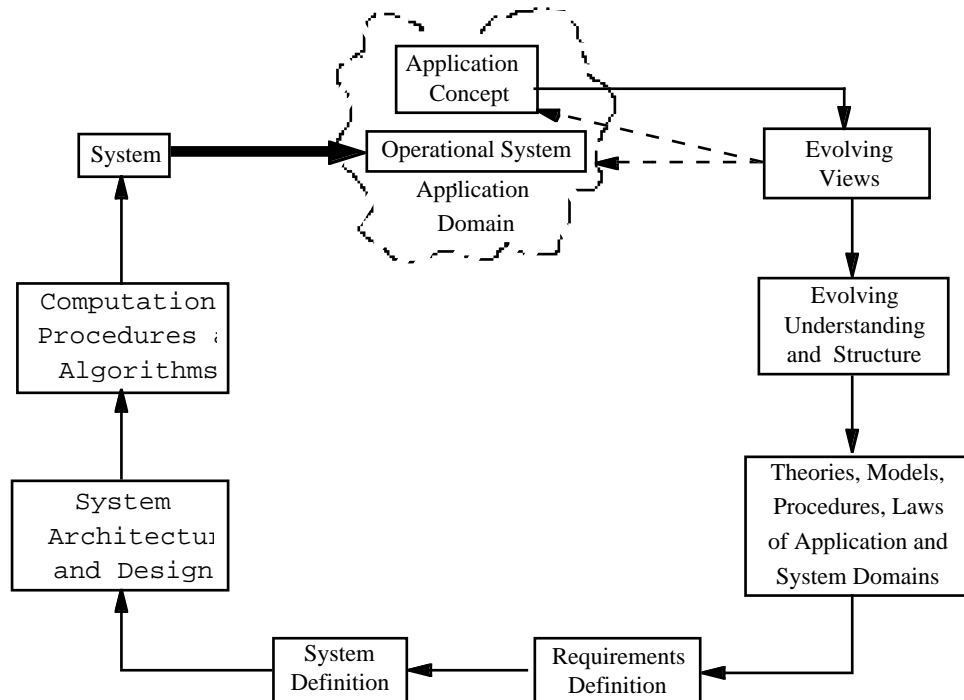


Figure 6. The System Installed and Operational

Because of its theoretical implications, one further observation must be included here. The model of figure 6 expresses the fact that the operational system is installed in and becomes part of the operational domain. This implies that, conceptually, the system includes a model of itself. This cannot, of course, be explicitly achieved in a finite system, nor, for that matter, implicitly. Thus, as already inferred from the fact that the domain is unbounded and the system is finite, the operational system is, in its very essence, an imprecise model of the operation with which it is associated, the application and its operational domain. That imprecision is embedded in assumptions that are, in turn, embedded in one form or another in the system. But the real world operational domain is dynamic, always changing at a rate that is accelerated by the installation and operation of the system. The loop identified is a source of positive feedback. Some of the assumptions embedded in the system, consciously or unconsciously, explicitly or implicitly, justifiably or not justifiably will eventually become invalid. What the practical consequences are of an invalid assumption being involved in a system execution is, in general, not known. Hence, however often a system has been run and produced satisfactory results, its next run may fail. The outcome, output, behaviour and consequences, of any *E*-type system execution, its effect in the operational domain is intrinsically uncertain, not unpredictable. It is these observations that led, some years ago to formulation of a Principle of Uncertainty in relation to software systems [leh89,90].

6. The Role of People

The feedback loop observed in fig. 6 identifies one of the sources of the ubiquitous and unending evolution required by all serious computing applications. But that is not quite the end of the story. Firstly apart from the initial viewpoint development, the process has been pictured as a sequential process. The process model as illustrated does not, for example, reflect the learning experience, the human communication and, hence, the iterative activity that is an integral part of each of the steps and between steps. Equally prominent by its absence from the figure are the influences, driving forces and controls that are exercised, consciously or unconsciously, by all levels of management, by marketers and support personnel, by users and by the people, activities and processes that make up the operational domain. Figure 7 provides a pictorial model that is a closer illustration of the reality within which *E*-type computer applications are developed and operate.

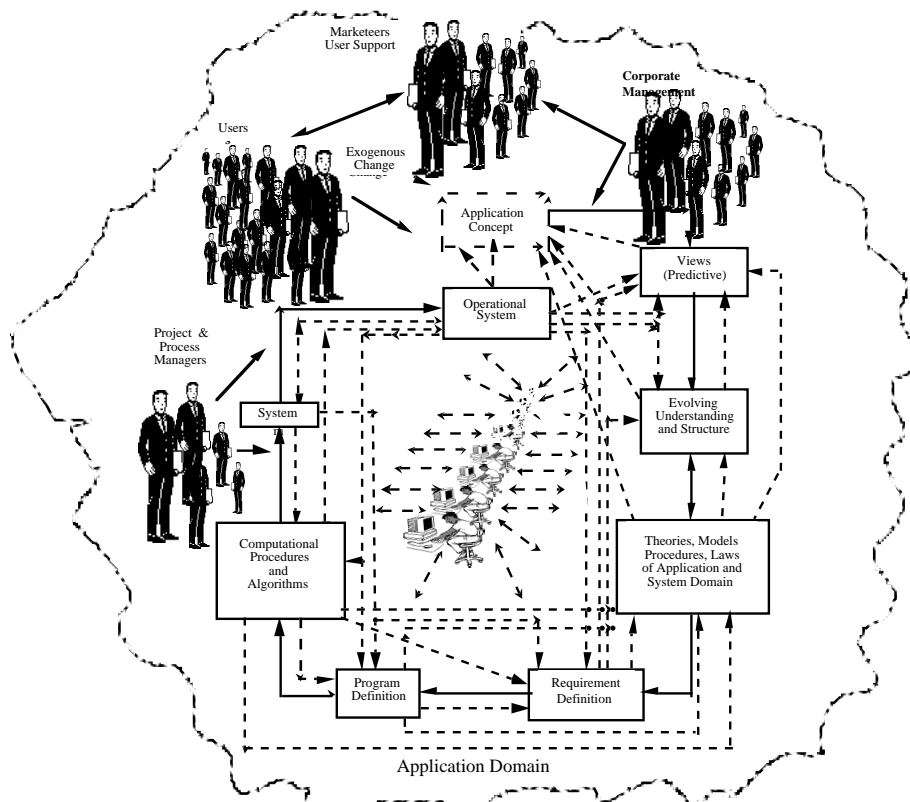


Figure 7 The Global Process

This picture provides a more solid, phenomenological, basis for the observation that *E*-type software evolution processes are complex multi-agent, multi-level, multi-loop, feedback system. It was the observations and inferences summarised in the preceding paragraphs that led to the formulation of the FEAST hypothesis and to the FEAST/1 project. That project is seeking to lay the foundations of a longer and more widespread investigation. This will develop models to be used for reasoning about, evaluating, predicting the behaviour and improving the software and system evolution processes that maintain computer application systems effective over their operational lifetimes. The work will also provide a foundation and a framework for a theory of software evolution.

The previous sections have talked, glibly, of feedback loops and control mechanisms. Despite the introduction of many tools at both the management and technical levels of software processes the vast majority, if not all, process planning and control mechanisms involve humans. In the first place and throughout the process they will have made explicit assumptions and will have also imposed implicit assumptions on the basis of other actions. As the process is followed, all its players receive information, observe, interpret their inputs, take decisions, communicate and act or fail to act on the basis of their interpretations, understanding, viewpoints and previous experience. The actions of the individual participant are therefore not precisely predictable. The behaviour of the process at the individual level is non-deterministic. At a higher level, however, the process behaviour is the consequence of the aggregate assumptions, decisions, activities and interactions. These have a degree of interdependence but, in many ways, be regarded as pseudo independent and pseudo random. It is this high level aggregate that largely determines the global, externally perceived, attributes of the process. Thus it is not surprising that one may usefully apply computational, statistical and other techniques to model and analyse process patterns and behaviours. In addition, and as referred to above, the FEAST/1 investigations are beginning to reveal the strength of the dynamics of the total process system once the system evolved through several releases. Local, individual or group action can cause deviations from the inbuilt trends and patterns, deviations or disturbances that may be visible from outside the process, for example a delay in system release or a glitch in system functionality. But a variety of feedback controls, explicit and implicit, will ensure process recovery and its return to the established patterns of behaviour.

This brief analysis has made no reference to situations where a process goes seriously wrong. There have been many, too many, instances of premature termination or even failure to deliver. That situation can be analysed in similar terms but is not attempted here. Nor has the topic of feedback analysis based software process improvement been addressed. In the long term that and the associated development of a theory of software process are likely to prove by far the most important outputs of FEAST/1 and successor investigations.

7. Acknowledgements

Grateful thanks are due to Dr Paul Wernick and to Juan F. Ramil for their continuing effort within the FEAST/1 project. Most of the results quoted here are due to their work. Thanks are also due to Professor Wlad Turski and to Dr Dewayne Perry who have made major contributions to the generation of the concepts from which FEAST/1 was developed and who, as Senior Visiting EPSRC Fellows, have made major contributions to the investigation via email and during their periodic visits. Also to the industrial collaborators who have devoted significant resources to recovering the data which is essential to our investigation and have participated actively in modelling, interpretation and analysis of the data and its relationship to process behaviour over the lifetimes of the various systems. EPSRC has supported the work through grants GR/K86008 (FEAST/1) and GR/L07437 (SVFs)

8. References - papers included in [leh85] are identified by an *

- [abd91] Abdel-Hamid T and Madnick SE, *Software Project Dynamics - An Integrated Approach*, Prentice Hall, Englewood Cliffs, NJ 07632, 263 p.
- [bel72] Belady LA and Lehman MM, *An Introduction to Program Growth Dynamics*, in Statistical Computer Performance Evaluation, W Freiburger (ed), Academic Press, New York, 1972, pp. 503 - 511
- [dij72] Dijkstra E W., *Notes on Structured Programming*, In Dahl, Dijkstra and Hoare, *Structured Programming*, Academic Press 1972, pp. 1 - 82
- [fri98] Friedman Y and Sandler U, *Dynamics of Fuzzy Systems*, to app. in Int. J. Chaos Theory and Application, inv. paper, 1998.
- [leh69]* Lehman MM, *The Programming Process*, IBM Res. Rep. RC 2722, IBM Res. Centre, Yorktown

Heights, NY 10594, Sept. 1969

- [leh74]* id., *Programs, Cities, Students, Limits to Growth?*, Inaugural Lecture, May 1974. Publ. in Imp. Col. of Sc. Tech. Inaug. Lect. Ser., vol 9, 1970, 1974, pp. 211 - 229. Also in *Programming Methodology*, (D Gries ed.), Springer, Verlag, 1978, pp. 42 - 62
- [leh78]* id., *Laws of Program Evolution - Rules and Tools for Programming Management*, Proc. Infotech State of the Art Conf., Why Software Projects Fail, - Apr. 1978, pp. 11/1 11/25
- [leh80a]* id., *On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle*, J. of Sys. and Software, v. 1, n. 3, 1980, pp. 213 - 221
- [leh80b]* id., *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Special Issue on Software Engineering, v. 68, n. 9, Sept. 1980, pp. 1060 - 1076
- [leh85] Lehman MM and Belady LA, *Program Evolution, - Processes of Software Change*, Acad. Press, London, 1985, 538 p.
- [leh89] id, *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
- [leh90] id, *Uncertainty in Computer Application*, Tech. Let., CACM, v. 33, n. 5, May 1990, pp. 584 - 586
- [leh94] id, *Feedback in the Software Evolution Process*, Keynote Address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics, Dublin, 7 - 9th Sept. 1994, Workshop Proc., Information and Software Technology, sp. is. on Software Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp. 681 - 686
- [leh96a] id, *Laws of Software Evolution Revisited*, Proc. EWSPT'96, Nancy, 9 - 11 Oct. 1996
- [leh96b] Lehman MM, Perry DE and Turski WM, *Why is it so hard to find Feedback Control in Software Processes?*, Invited Talk, Proc. of the 19th Australasian Comp. Sc. Conf., Melbourne, Australia, 31 Jan - Feb 2 1996. pp. 107-115.
- [leh96c] Lehman MM and Stenning V, *FEAST/1: Case for Support*, ICSTM, March 1996
- [leh96d] Lehman MM, *Process Improvement - The Way Forward*, Invited Keynote Address, Proc. Brazilian Software Engineering Conference, 14 - 18 October 1996, pp. 23 - 35
- [leh97a] Lehman M M, Perry DE, Ramil JF, Turski WM and Wernick PD, *Metrics and Laws of Software Evolution - The Nineties View*, Proc. Metrics '97, Albuquerque, NM, 5 - 7 Nov., 1997. Also as *Process Improvement - The Way Forward*, in *Elements of Software Process Assessment and Improvement*, IEEE CS Press, 1998
- [leh97b] Lehman MM, *Process Models - Where Next?*, "Most Influential Paper of ICSE 9 Award", Proc. ICSE 19, Boston, 20 - 22 May 1997, pp. 549 - 552
- [leh98] Lehman MM and Wernick PD, *System Dynamics Models of Software Evolution Processes*, Proc. Int. Wrkshp. on the Principles of Software Evolution, ICSE'98, Kyoto, Japan, April 20 - 21, 1998
- [lev90] Levary R R, *System Dynamics with Fuzzy Logic*, Int. J. Sys. Sci., 1990, v.21, n.8, pp. 1701-1707
- [mad96] Madachy R J, *System Dynamics Modelling of an Inspection Process*, Proc. 18th Int. Conf. on Softw. Eng., Berlin, 25 - 29 March 1996, IEEE Comp. Soc. ord. n. PR07246, IEEE Cat. n. 96CB35918, pp. 376 - 386
- [tur96] Turski WM, *Reference Model for Smooth Growth of Software Systems*, IEEE Trans. on Soft. Eng. v.22, n.8, Aug. 1966
- [ven95] *Vensim Reference Manual*, Version 1.62, Ventana Systems Inc., Belmont, MA 1995
- [wer98] Wernick PD and Lehman MM, *Software Process Dynamic Modelling for FEAST/1*, ProSim'98, Proc. Int. Wrkshp. on Softw. Proc. Simulation Modelling, June 22 - 24, 1998, Silver Falls, Oregon
- [wir71] Wirth N, *Program Development by Stepwise Refinement*, CACM, v.14, n.4, Apr. 1971, pp. 221-227
- [zur67] Zurcher F W and Randell B, *Iterative Multi-Level Modelling - A Methodology for Computer System Design*, IBM Res. Rep. RC 1938, Nov. 1967, IBM Res. Centre, Yorktown Heights, NY 10594 and *Information Processing 67*, Proc. IFIP Congr. 1968, Edinburgh, Aug. 1968, pp. D138 - 142

å

mml585
31/3/98