

**Process Improvement - The Way Forward**  
**M M Lehman**  
**Department of Computing**  
**Imperial College of Science, Technology and Medicine**  
**London SW7 2BZ**  
**mml@doc.ic.ac.uk**

IEE Colloquium, 21 March 1995, Digest No. 95/056, Colloquium title *Are Software Development Technologies Delivering their Promise?*

### **Background to the Search for Software Process Improvement**

The last twenty years or so have seen an order of magnitude growth in the functionality of computer applications and, hence, in the size and complexity of the software systems implementing them. It is, nevertheless, well established that processes transforming initial statements of requirements into installed software products still suffer from serious shortcomings. The continuing problems faced in system development and maintenance have led to strong pressure for systematic improvement of what has become known as *the process*. The invention of symbolic, assembly and high level languages in the fifties initiated the search for process improvement [ran94]. Public discussion of the many problems facing industrial software development at the 1968 NATO Garmisch conference [nau69] triggered a more general effort. Since then there have been significant advances in the provision of means for such development. Major R & D investment has produced numerous concepts, methods, techniques and tools. High level languages, structured programming, abstract data types, formal methods, non-procedural programming, object orientation, CASE, support environments exemplify innovations that were each, in turn, expected to overcome, once and for all, the many problems that have for so long frustrated consistent, cost effective, on-time development of functionally satisfactory and reliable software. None did.

### **Current Weaknesses**

Despite local benefit that innovation has brought to individual development steps too many problems still haunt industrial software development [gib94]. Introduction of improved methods, techniques and tools has not yielded a consistent capability for planned, on time, controlled-cost development of quality software that is functionally satisfactory when delivered. Nor has it resulted in major productivity growth, cost reduction or faster response to user needs. It has proven equally difficult to achieve major improvement in maintaining systems satisfactory as user needs and expectations change in evolving operational domains. Software *evolution* from concept to first installation and from release to release still relies on processes that are far from satisfactory [gib94].

### **The Software Process and the Drive for its Improvement**

Direct interest in the process as such has existed since at least the late sixties [leh69, roy70]. Wider interest was encouraged by the first International Software Process Workshop [pot84]. Despite doubts about the specific direction being recommended [leh87], Osterweil's keynote address to the ICSE 9 conference [ost87] triggered major activity in process modelling based on programmatic concepts. Most recently the focus of interest, particularly in industry, has turned to process standards and maturation as exemplified by the ISO 9000, TickIt and Spice process related standards and the SEI CMM model. Some industrial and commercial organisations have launched their own efforts both to obtain process quality certification and through R & D improvement programs such as that described at ICSE15 [maj93]. But progress is painfully slow. This at a time when the rapid spread of computer application and its ever deeper penetration into every facet of human activity implies ever increasing dependence, individual and collective, on maintaining implementing software and related software processes satisfactory in changing operational and maintenance environments [leh91].

### **Why the Slow Progress?**

The limited improvement provided by innovations such as those listed above can be individually explained, innovation by innovation. The failure of formal methods, for example, to yield generally visible global impact may be related to perceptions that their successful use requires mathematical skills and that the beneficial impact of their adoption is primarily process-local. CASE tools frequently yield less benefit than expected because, being acquired one at the time, they cannot easily be used together, may not even be able to communicate directly. Integration of their outputs often requires more effort than the effort saved through their introduction. Hence, their net contribution to productivity improvement say, is small or even negative.

The extent of the failure of innovation to yield major global benefit, however, suggests an overall constraint inhibiting process improvement. Is there perhaps a common phenomenon resisting or

even blocking improvement? If such can be identified a path to major improvement will have been opened. Techniques and tools to enable industry to exploit the opportunities it offers will follow.

### Information flow in the Software Process

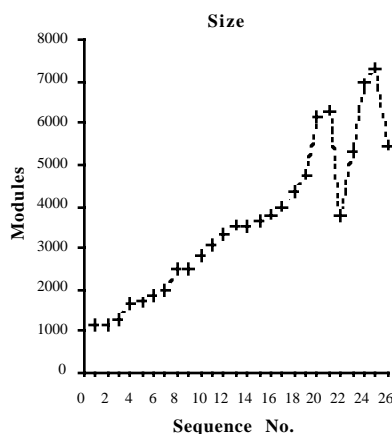
Processes whereby software is evolved *ab initio* or enhanced and extended, involve extensive feed around of information. The flow will be both internal to and between various activities and agencies (sub-organisations). Each of the latter will have its own responsibilities and objectives. Their joint action and interaction produces information, insight and enlightenment that drives, guides and controls the process. Some flow will influence process steps that follow the information-generating activity. Other will be fed back to *control* the sub-process from whose output it was derived. The process constitutes a complex information flow network. Since output from some process elements controls their subsequent behaviour the process constitutes a multi loop feedback system [leh94]. A forthcoming publication by Lehman, Perry and Turski will discuss the role and likely impact of feedback controls in processes whereby *E-type*<sup>1</sup> applications and software systems are evolved. For the moment it is sufficient to observe that for over four decades industrial software *process* evolution has been driven, at least in part, by organisational needs and ambitions. To ensure organisational and financial stability with planned growth feedback based *checks* and *balances* controls, for example, will have been employed<sup>2</sup>. Feedback control develops naturally in industrial process evolution in general and in *E-type* software process evolution in particular.

### The Software Process as a Feedback System

A characteristic property of all feedback systems is that the stabilising effect of negative feedback reduces the global impact of changes to individual forward path elements. In the limit, externally observable system behaviour will not visibly change in response to forward path changes unless and until excessive positive feedback causes system instability. To obtain the full impact on system behaviour from forward path changes requires adjustment of feedback paths and mechanisms.

As a feedback system, the software process must be expected to display this characteristic property. Thus even if there were no other feedback controls in the processes, checks and balances alone would tend to constrain the global impact of forward path improvement of internal processes. In practice there may be many other controls inhibiting process improvement as a side effect.

The significance of feedback in the software process and its role in determining its dynamics has long been recognised. It was referred to in passing by several speakers at the Garmisch Conference [nau69]. At about the same time it was briefly discussed in the 1969 *Programming Process* [leh69] report and, in more detail, in subsequent papers [bel72, leh85]. As an example brief mention may be made of the early identification of the feedback stabilised and controlled growth characteristics of OS/360 and other systems [leh80] as illustrated in the growth plot reproduced in the figure below.



As observed at the time of publication with reference to the first 19 data points [leh72] "... the plot ripple is typical of a self stabilising process with positive and negative feedback loops. From a long-range point of view 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 budgets varying, increasing numbers of users reporting faults or desiring new capability, varying management attitudes towards

<sup>1</sup> *E-type* software is defined as being required to *satisfactorily* address a real world problem in contrast to *S-type* software that is required to be *correct* in the full mathematical sense with respect to a fixed specification [leh78, 94b]

<sup>2</sup> Others were developed or evolved to support the search for improved product quality and process productivity. A third process attribute, responsiveness, whilst of equal societal significance has not been so avidly pursued.

system enhancement, changing release intervals and improving methods....". The last seven data points included in the plot reinforce the evidence for feedback control like behaviour by showing the consequences of excessive positive feedback, instability that resulted in the breakup of OS/360.

More recently, Abdul-Hamid and Madnick devoted a book to aspects of software process dynamics, concentrating on management aspects [abd91]. Weinberg [wei92] and Senye [sen93] provide extensive discussion of related themes. But to the best of the present writer's knowledge the first published suggestion that the characteristic property of feedback systems is a likely impediment to software process improvement did not appear till 1994. It is difficult to understand why the penny took so long to drop but then hindsight always raises that question.

### **Slow Progress in Process Improvement**

The innovations mentioned in the opening paragraphs all represent changes to the forward path of the software process. In general their introduction into practice was rarely, if ever, accompanied by review and adjustment of process feedback controls. It should, thus, not come as a surprise that their impact was limited. Is this not simply an instance of the stabilisation normally associated with negative feedback? For the moment, this observation is a conjecture to be verified in theory and practice. Once it has been verified one may develop means for its exploitation.

### **Project FEAST**

This analysis suggests that major software process improvement requires continuing review of feedback mechanisms. To date this aspect of improvement has been largely ignored. Hence, identification and adjustment of feedback mechanisms in current processes may itself be expected to produce improvement. In addition, the development of appropriate modelling and other techniques with their support tools will ensure that, in the future, relevant feedback controls will be identified and adjusted in tune with each forward path improvement to yield maximum benefit.

A comprehensive investigation to confirm the conjecture and to develop means for controlling and exploiting feedback is clearly required. The first step is the development of a conceptual and theoretical framework that includes means for the representation and evaluation of global processes. The investigation must also seek approaches to systematic process improvement, means to support such improvement and means for assessing the global impact of changes.

A study that will examine the role of feedback control in the process, process improvement and the exploitation of advancing technology to this end has been initiated. If successful it will ensure and sustain future advances in software evolution (development *ab initio*, enhancement, extension) process, yielding methods, tools and metrics for the effective evaluation, support and further development of process technology. An international project, FEAST, is now underway. Three workshops over the last nine months involved people from industrial, academic and research organisations in Canada, Finland, France, Norway, Poland, Portugal, UK and USA. The main focus so far has been on the identification and definition of basic concepts, the adoption of outline definitions, preliminary examination of project issues and objectives and consideration of how best and most profitably the investigation should proceed. This initial effort has been funded in part by the UK Department of Trade and Industry and in part by participants' organisations. Present support ends in March 1995. The rate of progress thereafter will depend on the further funding obtained.

The project is challenging but feasible. First practical results should be available within two years say. But in view of the difficulty of the issues under study the main body of results is likely to require 3 to 5 years to achieve. The degree of success and the rate at which it is achieved will clearly depend, at least in part, on the funding obtained. The calibre of people attracted to and participating in FEAST suggests that significant progress can be anticipated.

### **The Way Forward**

That the software process constitutes a feedback system with both positive and negative feedback, is indisputable, though what constitutes positive and what negative feedback requires clarification. That, as a feedback system, the process will possess the external stability property of such systems is plausible. It is, in fact, most unlikely not to be true. The involvement of people, however, in most, if not all, of the feedback loops as observers, communicators, decisors, evaluators and implementors, raises questions as to whether the system dynamics, its responses to changes in the forward path or feedback mechanisms, can be rigorously and systematically modelled, predicted or controlled. The progress made in system dynamics and in control theory applied to economic and organisation modelling suggests that limited progress can be achieved. There are, in fact, grounds for believing that the software process may be more amenable to control because of moderating and stabilising influences of both the application in its domain of operation and the program code.

The FEAST investigation is intended to explore and exploit the insight described above. What

seems clear is that for major process improvement the total global process, its mechanisms and characteristics must be considered whenever local improvements are investigated. Merely changing forward path technology can only produce limited benefit.

### Final Note

In papers at IFIP Congress '86 Brooks [bro86] and Turski [tur86], respectively, pointed out that one must expect neither a silver bullet nor a philosophers' stone to solve the software engineering problem once and for all. The FEAST conjecture is not an exception. If it can be exploited, it may make a significant contribution to improvement of the software evolution process. It must be seen just as that, no more.

### Acknowledgements

This paper presents initial observations and concepts developed during the formative stages of the FEAST project during 1994 and early 1995. The clarification that resulted from numerous discussions with Professor Wlad M Turski and Drs Dewayne Perry and Vic Stenning has produced a well founded view that was presented to and discussed at the third FEAST workshop [fea95]. Their insight, contributions and support as well as that of the participants in the three FEAST workshops is gratefully acknowledged.

### References

- [abd91] Abdel-Hamid T and Madnick S E, *Software Project Dynamics - An Integrated Approach*, Prentice Hall, Englewood Cliffs, NJ 07632, 263 p.
- [bel72] Belady L A and Lehman M M., *An Introduction to Program Growth Dynamics*, in Statistical Computer Performance Evaluation, W Freiburger (ed), Academic Press, New York, 1972, pp. 503 - 511
- [bro86] Brookes F P, *No Silver Bullet - Essence and Accidents of Software Engineering*, Information Processing 86, Proc. IFIP Congress 1986, Dublin, Sept. 1-5, Elsevier Science Publishers (BV), (North Holland), pp. 1069 - 1076
- [fea95] Preprints of the Third International FEAST Workshop (Lehman M M ed.), Imperial College, Oct. 1994
- [gib94] Gibbs W W, *Software's Chronic Crisis*, Scientific American, Sept. 1994, pps. 72 - 80
- [leh69] Lehman MM, *The Software Process*, IBM Report no. RC 2722, Yorktown Heights, 1969, also in [leh85]
- [leh78] Lehman M M, *Laws of Program Evolution - Rules and Tools for Programming Management*, Proc. Infotech State of the Art Conf., Why Software Projects Fail, - Apr. 9 - 11 1978, pp. 11/1 - 25
- [leh80] Lehman M M, *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Special Issue on Software Engineering, vol. 68, no. 9, Sept. 1980, pp. 1060 - 1076
- [leh85] Lehman MM and Belady L, *Program Evolution - Processes of Software Change*, Academic Press, 1985
- [leh87] Lehman M M. *Process Models, Process Programs, Programming Support - Invited Response To A Keynote Address By Lee Osterweil*, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March 2 Apr. 1987, IEEE Comp. Soc. pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 14 - 16
- [leh91] Lehman M M, *Software Engineering, the Software Process and their Support*, IEE Softw. Eng. J., Spec. Iss. on Software Environments and Factories, Sept. 1991, vol. 6, no. 5, pp. 243 - 258
- [leh94] Lehman MM, *Feedback, Evolution and Software Technology*, Preprints of the First International FEAST Workshop, Imperial College, June 1994
- [leh94b] Lehman MM, *Properties of S and E-Type Systems*, Preprints of the Second International FEAST Workshop, Imperial College, Oct. 1994
- [maj93] Major J, *Keynote Address*, ICSE15, Baltimore, 17 - 21 May 1993
- [nau69] Naur P and Randell B, *Software Engineering - Report on a Conference, Sponsored by the NATO Science Committee*, Garmisch, 1968, Scientific Affairs Division, NATO, Brussels 39, 1969
- [ost87] Osterweil L, *Software Processes are Software Too*, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March - 2 Apr. 1987, IEEE Comp. Soc. Pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 2 - 13
- [pot84] Potts C (ed), *Proceeding of the Software Process Workshop*, Egham, Surrey, UK, Feb. 1984. IEEE, cat. n. 84CH2044-6, Comp. Soc., Washington D.C., order n. 587, pp. 27 -35
- [ran94] Randell B, Ringland G and Wulf B, (eds.), *Software 2000 - a View of the Future*, Output of a Forum Sponsored by ICL and the Commission of the European Communities, 1994, D2D, Cavendish Rd., Stevenage, SG1 2DY, UK
- [roy70] Royce W W, *Managing the Development of Large Software Systems*, IEEE Wescon, August 1970
- [sen93] Senye P M, *The Fifth Discipline - The Art and Practice of the Learning Organisation*, Century Business, London, 1993
- [wei92] Weinberg G, *Quality Software Management*, Dorset House Publ., NY, NY, 1992
- [tur86] Turski W M, *And No Philosophers' Stone Either*, Information, Processing 86, Proc. IFIP Congr., Dublin, Sept. 1 - 5, 1986, Elsevier Sci. Pubs, London, pp. 1077 - 1080