Process Modelling - Where Next

M M Lehman
Department of Computing
Imperial College of Science. Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
+44 (0171) 594 8214
mml @doc.ic.ac.uk

ABSTRACT
After limited interest prior to the 1980s, the software process attracted the attention of a small group of researchers and practitioners as evidenced by a series of International Process Workshops commencing with the first in 1984. General interest had, however, to wait until Osterweil's now classic paper being honoured today and the present author's response were delivered at ICSE 87. This brief paper seeks to move forward from the positions presented then to introduce three process related issues, software process improvement, feedback in the software process and business process improvement. The first is, currently the principal focus of the software process community. The second, it is believed, should be. The third is equally relevant. Individually and collectively these issues appear, to this author at least, to make the majority view of the current focus of process modelling in general and process programming in particular largely irrelevant.

Keywords
Process: modelling, process programming, process improvement, feedback, feedback systems system dynamics

PROCESS MODELLING AND PROGRAMMING

The ICSE 97 program committee will not have found it difficult to select Lee Osterweil's classic paper *Software Processes are Software Too* [1] as the most influential paper of ICSE 87. As evidenced by a number of earlier publications [2, 3, 4, 5] and the first three of the now regular International Process Workshops [6, 7, 8] this paper did not pioneer the concept of process studies and process modelling. But, as predicted, his fresh approach and the underlying philosophy captured and retained the imagination and fascination of individual and groups of researchers in unparalleled fashion. Whether the resulting R & D effort produced major new insight, understanding, and progress in planning, designing, controlling and improving the process is a matter for debate. But this in no way detracts from the originality of Osterweil's thinking or the influence, for better or worse, that his paper has had on the directions that software engineering research has taken in the last decade.

ICSE 97 Boston MA USA

As is well known, I was a doubter from the start. I must express my special appreciation to the program committee of ICSE 97 for, nevertheless, associating me with this award.

In the short time available to me today I am not able to reopen the earlier debate. That is recorded in the ICSE 87 proceedings [1, 9] and there is little in what I said then that I would wish to change; though something could be added. I restrict myself to a brief outline of three developments that have surfaced in recent years. Individually and collectively they imply a change of direction for software process modelling and relegate the original concept of process programming to an interesting technique with conceptual implications but one that cannot be expected to have major impact on future development.

SOFTWARE PROCESS IMPROVEMENT

Following on the pioneering work of the SEI [10], model based software process improvement has become a major research and applied software engineering activity in both academia and industry. In the application of process modelling to process improvement, as with the SEI CMM models for example, a wide range of activities at many levels are considered. Process programming based modelling, on the other hand, focuses, primarily, on technical development, definition, control, direction and sequencing of activities. It concentrates on individual and small group activity where individual actions can be well specified, rather than on organisational activity. As discussed below, this aspect of process plays only a limited role in process improvement, except at low levels of process achievement. In primitive processes the introduction of a new method or tool has more than a local impact; the introduction of a method or tool, for example, can yield visible improvement in some aspect of the overall process. Once these have been introduced at these levels and a degree of maturity achieved, further improvement requires attention to a whole gamut of issues, managerial, organisational and high level technical (eg the addition of additional process steps, the use of metrics). Improvement of individual steps, better development methods, improved support tools, whatever their local effect, have little impact at the global level [11].

One is forced to conclude that once one has advanced beyond primitive processes, CMM levels one and two for example, the principal current procedural process modelling approaches have little, if anything, to contribute to the search for process improvement. What is required in any

particular context is high level understanding and representation of the needs of the business and of relevant organisational processes in that context. One must achieve thorough insight into the manner in which these act and interact. One must learn how to design, control and modify individual and joint action to produce the desired global output and impact. One must assess the effectiveness of the process as experienced and evaluated by the world outside the development, marketing and support organisations.

Detailed design of constituent steps in general, and technical steps in particular, is not the most critical issue in seeking to achieve the desired overall processes or their improvement. Local fine tuning cannot be expected to make a major contribution to global effectiveness. It is a well known property of complex systems that local optimisation usually causes global sub-optimisation. And even without this effect the impact is small. After all, a fifty per cent efficiency improvement in an activity that represents, say, five per cent of the activity or resource required to produce the product from start to finish, makes at best a two and a half percent impact, often very much less. The essential lesson to be derived from current improvement approaches is that one must develop a global view and comprehensive insight as to how, through their processes, organisations achieve and maintain quality products. This goal demands models of the processes and techniques to achieve and exploit them that are quite different to the majority of those current in the process modelling community at large.

FEEDBACK IN THE SOFTWARE PROCESS

A second major development that signals a change in direction for process modelling arises from the realisation that the software process is a complex, multi-loop, multi-level feedback system [11]. This was first recognised following a 1969 study of the IBM programming process [3, 12] which led eventually to a study of the evolution of OS/360 and to the conclusion that the system's growth was regulated by a self stabilising feedback process [13]. From this it followed that understanding and improving the process required it to be treated as a feedback system [14]. Moreover, in these systems, intrinsically, humans play a major controlling role. This greatly complicates disciplined analysis of such systems.

Some years ago, in asking why the global industrial process still leaves so much to be desired despite the many advances in software technology, these observations were recalled. Now, an intrinsic property of systems that include negative feedback loops and mechanisms, however controlled, is that the impact of changes to individual forward path mechanisms outside the loop is attenuated in proportion to the amplification in the loop. This phenomenon is not simple to interpret in the software process context. Nevertheless, it may explain why global characteristics of software development processes are not responsive to changes in individual forward path steps or to the introduction of new ones except in primitive processes where negative feedback control is weak or absent. Where negative feedback is present it is likely to constrain forward path improvements such as the use of high level languages, structured programming, new development paradigms, formal methods, disciplined requirements analysis and specification, CASE support and so on to explain why all these are limited in their global impact. It leads to the paradoxical situation that the more advanced a process, the more technical, management, organisational and user derived feedback control is applied the less benefit will accrue from local *technological* change *unless feedback controls are adapted to the circumstance created by such change.* Processes not employing feedback control can yield significant improvement as a consequence of technological advances alone. Mature processes are likely to respond to localised improvements only if feedback mechanisms are also adjusted. Adjusting only the latter may, in fact, itself yield significant improvement to global process characteristics.

This observation and its implications were expressed in a hypothesis, the FEAST hypothesis, as follows, [11, 15]: *As complex feedback systems, E-type [13, 16] software processes evolve strong system dynamics and with it the global stability characteristics of other feedback systems. Consequent stabilisation effects are likely to constrain efforts at process improvement.* It was restated in a project proposal [17] as:- *As for other complex feedback systems, the dynamics of real world software development and evolution processes will possess a degree of autonomy and exhibit a degree of global stability.* The resultant FEAST/1 project in the Department of Computing at Imperial College is now investigating this hypothesis.

This short presentation is not the place for a detailed review of the project or to present our results to date. It is, however, appropriate to remark that early results [18, 19] are encouraging. So far the analysis has concentrated on the evolution of the Logica plc FW financial transaction system now in its tenth release. The data indicates that the evolution of this system has characteristics similar to those of OS/360 [16]. The growth trend ripple as in figure 1, for example, is reminiscent of that of OS/360 as in figure 2. It was, of course, this ripple that first suggested the presence of feedback control. Moreover, the laws of software evolution as previously stated [14, 16, 20, 21] are upheld [22], or rather, not negated, by the data. There is also strong evidence [18], that a controlling internal dynamics develops over the early releases, as in figure 3. The "E" parameter of that figure is the constant of Turski's inverse square growth model [18]. The plot shows that data from the first six releases suffice to determine the value of E. Thereafter, the internal dynamics dominates further growth and the model provides a growth trend predictor accurate to better than 5%. This is a remarkable result [18] that greatly increases confidence in the validity of the FEAST hypothesis.

That all but the most primitive *E*-type software processes constitute a feedback system is indisputable. This, by itself implies that they cannot be satisfactorily modelled using the techniques widely in vogue in the process modelling community. The R&D challenge is, therefore, to discover and develop more appropriate techniques. As a first step, the FEAST/1 project, which involves also four major industrial collaborators, is using black box analysis of real industrial processes. Figures 1 and 3 represent early results

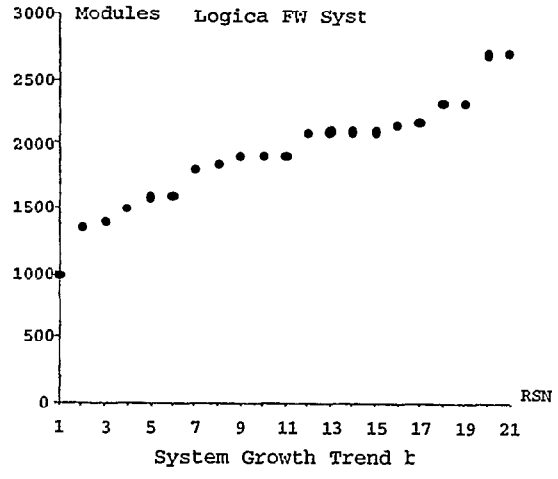

Fig. 1. 1990s System - Logica FW

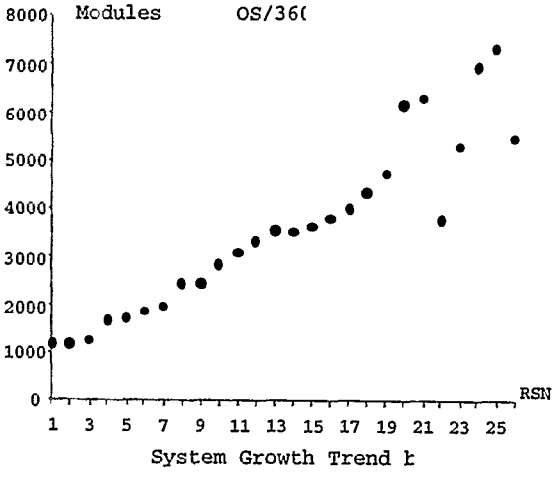

Fig. 2. 1970s System - OS/360

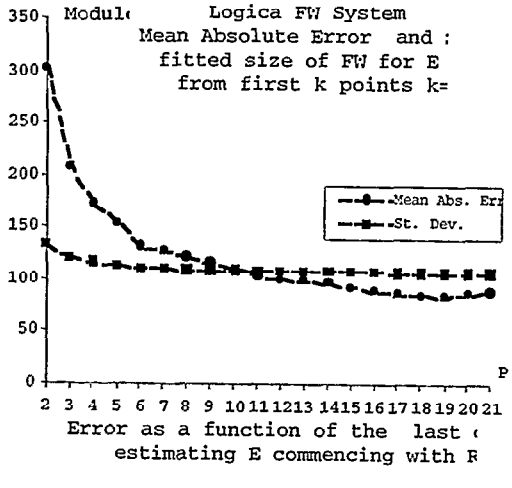

Fig. 3. - Development of the Logica FW System Dynamics

from that analysis whose aim is to demonstrate feed-back-like behaviour and the presence of system dynamics effects. White box studies based on systems dynamics modelling [24, 25] will then seek to identify actual feedback control mechanisms, assess their impact on the global process and on the impact of changes to them, and identify, implement and measure or otherwise evaluate potential improvements [17]. Multi agent modelling techniques will also be explored in this context.

BUSINESS PROCESS IMPROVEMENT

One further brief observation must be made. In general, *E*-type software systems are not developed for their own sake. They are required to address, in the most effective manner, a need in some domain. In seeking to use computers in the late fifties and early sixties US banks and insurance companies for example, the first business organisations to attempt major, if not total, automation, soon recognised that it was not effective to simply install computers as electronic clerk- replacements. In introducing them one must ask the question *"How shall our business be conducted now that computers are available?"*. Only recently has this awareness spread more widely; have organisations begun the search for overall *business process* improvement as a disciplined and integrated activity that also includes computer supported processes. In addressing this question it was soon discovered that for businesses operating *legacy* computer systems the freedom to change is severely constrained unless the computer software is reliably, responsively and economically adaptable. Modelling and improving a software process, a process including *ab initio* development, fault fixing, adaptation and extension (ie. software evolution), or modifying and extending its products must be done in the context of the total business including its clients, not as a self contained exercise. And this is also true for the software industry where software evolution is, basically, just another business process. There is, therefore, little point in modelling the technical software development process in ever greater detail. The *total* process must be modelled in its business environment to include all relevant activities and their feedback mechanisms. This requires that the latter are identified and understood, and that appropriate techniques are available.

CONCLUSIONS

This very brief analysis indicates why many of the current approaches to process modelling and the objectives they have been directed at have largely outlived their usefulness. More appropriate approaches and techniques must now be identified, explored, developed and applied. I believe that feedback control dominates the software process both in its technological aspects and in its organisational context. Mastery of feedback mechanisms demands adequate quantitative models individually calibrated against real world software evolution environments. Eventually it may be possible to develop generic models but that lies in the distant future. A widespread concerted effort at global software process modelling is required, one that takes into account the facts as outlined. Current modelling approaches and techniques will need to be augmented by others that promise hope of success in domains that are significantly wider and more complex than those currently considered.

ACKNOWLEDGEMENTS

As the FEAST core group, Dr Dewayne Perry and Professors Vic Stenning and Wlad Turski have played a major role in developing and exploring the concepts and results presented here [26]. More recently Dr Paul Wernick and Juan F Ramil have joined the FEAST/1 team and have also made significant contributions. It is also appropriate to mention the many participants in the three International FEAST Workshops (1994/5) who in their critical appraisal and creative criticism of the work of the core group helped ensure FEAST/1 funding, launch and further progress. Acknowledgement is also due to the EPSRC for grants numbers GR/K86008 supporting FEAST/1 and and GR/LO7437 that permits the continuing involvement of Professor Turski and Dr Perry.

REFERENCES

1. 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*, 2 - 13
2. Bennington H D, Production of Large Computer Programs, *Proc. Symp. on Advanced Computer Programs for Digital Computers, sponsored by ONR*, June 1956, Republished in *Annals of the History of Computing*, Oct. 1983, 350 - 361
3. Lehman M M, The Programming Process, *IBM Res. Rep. RC 2722, IBM Res. Centre, Yorktown Heights, NY 10594*, Sept. 1969. Also. in [*13*], 39 - 83
4. Royce W W, Managing the Development of Large Software Systems, *IEEE Wescon*, Aug. 1970, 1 - 9
5. Boehm B W, Software Engineering, *IEEE Trans. on Comp., v. C-5, n. 12*, Dec. 1976, 1226-1241
6. Potts C (ed), *Proc. of the Softw. Process Worksh.*, Egham, Surrey, UK, Feb. 1984. *IEEE cat. n. 84CH2044-6, Comp. Soc.*, Washington D.C., *order n. 587*, 27 -35
7. Wileden J C and Dowson M (eds), *SE Notes, Special Issue on The 2nd International Workshop on the Software Process and Software Environments*, Coto de Caza, Cal., 27-29 March 1985, *Softw. Eng. Notes, v. 11, n. 4*, Aug. 1986
8. Dowson M (ed), Iteration in the Software Process, Proc. *3rd Int. Proc. Worksh., IEEE Comp. Soc. Press*, March 1987
9. 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*, 14 - 16
10. Paulk M C, Curtis B and Chrisi M B, Capability Maturity Model for Software, Version 1.1, *Softw. Eng. Tech. Rep., CMU/SEI-93-TR*, Feb. 24 1993
11. Preprints of the (first) FEAST Workshop, M M Lehman (ed.), *Dept. of Comp., ICSTM*, June 1994
12. Lehman M M and Belady L A, Program Evolution, - Processes of Software Change, *Academic Press*, London, 1985, 538 p.
13. Belady L A and Lehman M M, An Introduction to Growth Dynamics, *Proc. Conference on Statistical Computer Performance Evaluation*, Brown U. 1971, *Academic Press*, 1972, W Freiberger (ed.), 503 - 511
14. Lehman M M, Laws of Program Evolution - Rules and Tools for Programming Management, *Proc. Infotech State of the Art Conf., Why Software Projects Fail*, - April 9 - 11 1978, 11/1 - 11/25
15. Lehman M M, *Feedback, Evolution And Software Technology, Pos.* Paper - *ISPW9, Proc. 9th Int. Softw. Process Workshop*, 5 - 7 Oct. 1994, publ. by *IEEE Comp. Soc.*, 1995
16. Lehman M M, Programs, Life Cycles and Laws of Software Evolution, *Proc. IEEE Special Issue on Softw. Eng., v. 68, n. 9*, Sept. 1980, 1060 - 1076
17. Lehman M M and Stenning V, FEAST/1 - Feedback, Evolution And Software Technology; Case for Support, *EPSRC Research Proposal, Dept. of Comp., ICSTM., London SW7 2BZ*, Nov. 1995/March 1996, 11 p.
18. W M Turski, A Reference Model for the Smooth Growth of Software Systems, U. of Warsaw, March 1996, *IEEE Trans. Softw. Eng.*, v. 22, n. 8, Aug. 1996, 599 - 600
19. Lehman M M, Process improvement - The Way Forward, Invited Keynote Address, *Proc. Brazilian Softw. Eng. Conf.*, 14 - 18 Oct. 1996, 23 - 35
20. Lehman M M, Programs, Cities, Students, Limits to Growth?, Inaug. Lect., May 1974. Publ. in *Imp. Col of Sc. Tech. Inaugural Lect. Ser., vol 9, 1970, 1974*, 211 - 229. Also in *Programming Methodology*, (D Gries ed.), *Springer Verlag*, 1978, 42 - 62
21. Lehman M M, On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle, *J. of Sys. and Softw., v. 1, n. 3*, 1980, 213 - 221
23. Lehman M M, Laws of Software Evolution Revisited, Pos. Paper, *EWSPT96, Oct. 1996, to be published by Springer Verlag*
24. Abdel-Hamid T and Madnick S E, Software Project Dynamics: An Integrated Approach, *Prentice Hall, Englewood Cliffs*, 1991, NJ 07632, 264 p.
25. Madachy R J, System Dynamics Modelling of an Inspection Process, *Proc. ICSE 18*, Berlin, 25 - 29 Mar. 1996, *IEEE Comp. Soc. ord. n. PR07246, IEEE Cat. n. 96CB35918*, 376 - 386
26. Lehman M M, Perry D E and Turski W M, 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. 107-115.