A Brief Review of Feedback Dimensions in the Global Software Process

Goel Kahen M M Lehman
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ
Tel +44 20 7 594 8214; +44 20 7 594 8216
Fax +44 20 7 594 8215
{gk, mml}@doc.ic.ac.uk

Abstract

FEAST, an ongoing study of the role of feedback in the software process, was prompted by various factors including the need to identify the mechanisms underpinning observed phenomena in a series of metrics-based studies of evolving software systems conducted during the seventies and eighties. Evidence to date indicates that feedback loop mechanisms play significant role in determining the performance and dynamics of software processes. To improve understanding of the evolutionary behaviour of software systems and to exploit feedback in the context of process improvement it is necessray to improve knowledge of the origin and sources of feedback phenomenon. This is also a prerequisite for a systematic definition of control and policy mechanisms for the management of such processes. This paper refers to some of the many dimensions that appear to relate to the issue of feedback in the global software process. It is argued that empirically assessing and modelling the presence and importance of those different dimensions in industrial software processes can bring significant progress to the FEAST investigation.

Keywords: Feedback, Management Control, Software Evolution, Software Process, Process Improvement

Introduction

Feedback is one of the fundamental processes that is contained in almost all manageable models. Feedback mechanisms and interactions are now conceptualised and considered one of the major influences underpinning software evolution phenomena. Prompted in the first [5] by observations and interpretation of software growth dynamics and at the early stage of this new conceptualisation, it was suggested that an organisation is developing and maintaining a large program should be regarded as a system in the 'system theoretic sense' [6]. It was argued that the "system behaves as a self stabilising feedback system… The process leads to a process dominated by feedback…. with long range trends…. and invariances" [14]. This implied that software processes constitute feedback systems [15, 18, 20, 21]. Studies of the evolution of a number of software systems during the seventies and eighties led to the recognition of patterns and similarities which were encapsulated in a set of laws of *E*-type[1] software evolution [13, 14, 15, 16, 17, 19, 23]. This work led to a recognition of, *inter alia*, the important role of the global software process and of user satisfaction [18]. It also led to the observation that visible improvement of software process has to be assessed at the global system level. The most recent of the eight laws so far identified is termed feedback

[1] *E* -type systems that is systems that support applications operating in the real world [17]. The common properties of such systems include intrinsic need for continuing evolution, a loosely defined requirement or expectation that users are satisfied with the system as is. An *E*-type system is judged by the results it delivers as it executes in the real world, its performance, its behaviour in execution, the functionality it delivers, its ease of use and so on. In this they differ from *S*-type systems where the criterion of acceptability is that of mathematical correctness, relative to an absolute specification.

system. It was restated and extended as the FEAST, *F*eedback, *E*volution *A*nd *S*oftware *T*echnology, hypothesis [18]. It states that *"E-type evolution processes are multi level, multi loop, multi agent feedback systems and must, in general, be treated as such to achieve major process improvement for other than the most primitive processes."* The initial examination of the hypothesis was undertaken by a series of international workshops [18, 21]. Most recently it has been investigated by the EPSRC funded FEAST/1 project (1996-1998) [19]. It has been followed by the on-going FEAST/2 (1999-2001) project [22].

The FEAST studies were prompted by the need to identify the mechanisms underpinning observed phenomena. Evidence to date indicates that feedback loop mechanisms play an important role in determining the performance and dynamics of software processes. Much is still to be done to systematically exploit understanding of such mechanisms and applying it, for example, in process design. Up to the present, the study has not identified the many dimensions that affect feedback in the global process. This paper is an initial attempt to examine some of these to permit their systematic investigation. Such dimensions are derived, at least in part, from the management science view of feedback in management control.

The Proposed vs the Actual Software Process

Apart from, perhaps the individual programming process smallest or most primitive, software processes are complex systems in their own right. Our ability to define, understand and manage them without proper concepts, rules and tools is limited. Current software process knowledge is mostly of a technical nature, based, for example, on the abstract world of computer science and computer science-based software engineering techniques. The latter tends to restrict what in some software improvement models has been termed the key process areas (KPA) [25]. Beyond the KPA towards, for example, software process management issues, there is a need for appropriate concepts, rules and tools. Progress in this area is hampered by many challenges such as that like many other industrial processes involve, and to some extent are determined by human, organisational and other factors. How to conceptualise and abstract the influences of the latter poses major challenges.

In the conceptual world, ideas can be interpreted as mental models that reflect human understanding. One here must distinguish two categories: models related to the *proposed* or desired system behaviour and models related to the *actual* or practical situation. In mental models, the first category appears to dominate the second. Thus, it is not surprising that the investigations, for example, in software process description languages have found so little application in practice [24]. In order to understand and master software processes it is necessary to consider both categories of models. The degree of mastery of software processes will relate to progress in both areas.

System behaviour can be examined, for example, from the perspective of *system dynamics* [9], as in the FEAST projects. Other methodologies, such as *soft systems* [7, 8] can also contribute to a better understanding of the real software process phenomena. When investigating the real world, measurement and systematic empirical study must play a major role. One must also consider that the real world also involves the views, desires, expertise, goals, criteria and plans that exist in the conceptual world, the mental models, of those involved, such as developers, users, supporters, managers and others. Furthermore, one must consider the role of factors, such as, for example, the organisational and managerial policies, strategies and so on, the forces and constraints placed on the process by its environment. In sum, many factors, both within and external to the software process determine the behaviour and performance of that process. To conceptualise and identify the individual influences of these may be challenging but appears to be necessary if one is ever to achieve a discipline of feedback control for software processes.

The Many Dimensions of Feedback in the Global Process

In general, feedback loop mechanisms has been classified into two different types: *reinforcing* or positive (i.e. amplifying) and *balancing* or negative. While the former is seen as a driver for change the latter operates, *inter alia*, whenever there is goal-seeking behaviour. The behaviour that results from a reinforcing loop is either accelerating growth or accelerating decline. Balancing feedback mechanisms on the other hand, force the system in which they are located, systems to seek stability in order to moderate or reduce the discrepancy between desired and actual behaviour.

E-type software processes aim to maintain stakeholder satisfaction with software behaviour in execution within a changing real world domain. Many usage domains involve a business (competitors, markets, regulations) and rely on a set of technologies. As the business environment accelerates its rate of change and technology also changes [11], the software process must react to accordingly to maintain stakeholder satisfaction. In an ideal situation, the software system would be immediately adapted to the changing environment. In reality the software process is driven by feedback loops which propagate, for example the discomfort perceived by the stakeholders as a consequence of the mismatch between software capability and functionality and current business demands. Such propagation of concerns involves many reinforcing loop mechanisms in users' and developers' organisations. But these are subject to the control of balancing mechanisms. The latter are required to guarantee organisational smooth operation and long term survival.

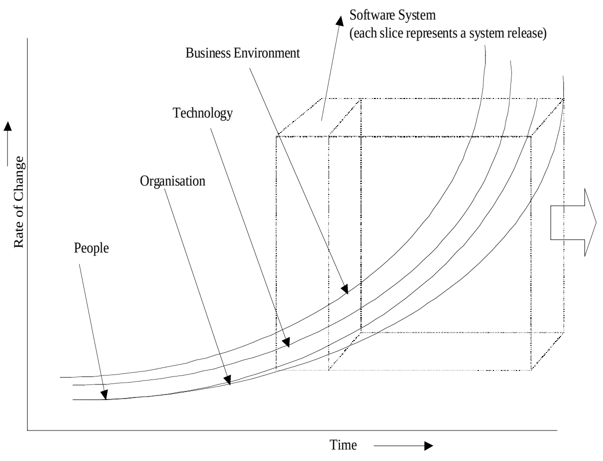

Figure 1: The major drivers of the software evolution process

One may distinguish four role playing constituents: people, organisational (managerial and structural aspects), (business) environmental and, technological (own and related aspects). In some circumstances these are referred to the organisational, technological, information and human aspects (see, for example, [10, 11, 12]). Figure 1 illustrate the process of such dynamism. The systems viewpoint requires to recognise that the human actors are an integral part of the feedback process. They are not standing apart from it. Impacts generated by changes in these four elements provide different types of feedback that drive the software evolution process. In this context, each element (i.e. people, organisation, technology and business environment) contributes its own weight and impact to the entire process and contribute to the global feedback. In order to understand and possibly manage the software process one needs to define and understand the potential effect of each type of feedback and their interrelationships. One may start by looking individually at each of these four elements within software evolution historical data and trends.

We believe that in addition to the four elements identified above as the sources of major feedback loops, there are two other major sources. The latter exist as inherent part of the software evolution process. One feedback loop involves the usage domain and users' reactions regarding the last and previous software releases. Another (feedback) loop runs internally within the technical software process and conveys internal reactions to the last software release and to the work in progress. While the latter addresses problems that arise, for example from the appropriateness or otherwise of software functionality by itself, the former ensures that changes in the environment including both business or technological environments have not rendered the recommended or implemented software less than fully adequate or even obsolete (see Figure 2). Note that such changes may affect software already implemented, software that is in process of being implemented or that planned. Another point to make here is that the two feedback loops of Figure 2 are essential to ensure an adequate process in which, the external one being considered part of the global software process. In other words, the (feedback) loop from system output to system input is part of the system and, must therefore be included in the measurement of software system effectiveness. As Beer [4] pointed out the first principle of organisational control is that the controller is part of the system under control. Regarding this point, some cybernetic control mechanisms are already well understood and their study may help to improve software evolution management. Furthermore, the controller grows with the system, and, if we look back through time, we see that the controller itself evolved with the system.

The software process can be studied in terms of the proposed content of a release and its actual context. The proposed content of a release relates to the question of how well the *needs* are identified and defined. The software context relates to the degree to which the tools, techniques and approaches to meet those needs are defined. The ‘needs’ factor is not stable (due to reinforcing feedback); as the software system has to be adapted to changing needs and changing environment. Based on both of these changes, software content and software context must also change leading to increasing software and software process and product complexity. In addition, uncertainty will bring more complexity to the software process because of needs misunderstanding (i.e. ill-defined feedback), and also because of ill-defined tools and techniques. This is caused by the lack of appropriate of control mechanisms or malfunction of the latter.

A significant portion of the software process and software metrics literature has focused on *endogenous* mechanisms (i.e. internal to the technical software process) formal and informal. These are devised and established by software developers and managers without the involvement of others. The control variables, in particular, include endogenous technical and financial software process properties. But the importance of exogenous (i.e. external to the software process) variables is undeniable. It is crucial to recognise that the behaviour of any software, as an organism, is regulated by a combination of both exogenous and endogenous control mechanisms. In fact, the behaviour of software process is function of both environment-initiated (i.e. exogenous) control mechanisms as well as self-initiated technical (i.e. endogenous) control mechanisms. Thanks to the cybernetic theory we now know that the behaviour of many organisms depends first and foremost on its own self-regulatory mechanisms and only indirectly upon exogenous regulatory mechanisms [2]. One may conclude that, the effectiveness and efficiency of exogenous control mechanisms depends crucially on the extent to which they are able to appropriately shape the self-regulatory behaviour of the organism being controlled. Exogenous control mechanisms, in and by themselves, are almost always indirect and incomplete. In order to understand the dynamics of a system one must study the overall feedback loop structure with its mechanism. This to a significant extent is believed determine the overall process performance[2].

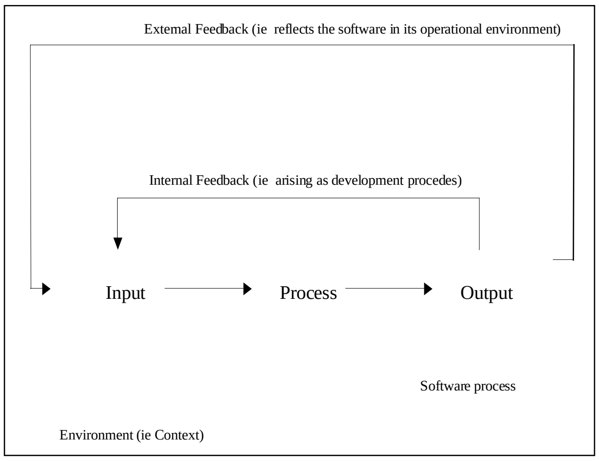
External Feedback (ie reflects the software in its operational environment)
Internal Feedback (ie arising as development procedes)
Input
Process
Output
Software process
Environment (ie Context)

Figure 2: Two major feedback loops in the software process

Feedback as Reflected in Organisational and Managerial Literature

The role of various forms of feedback in decision making processes has been considered in organisational and managerial literature in general and, in decision making studies in particular (e.g. [1, 3, 26, 28]). The provision of feedback in this respect is that it has the potential to alleviate some of the difficulties associated with the limited cognitive ability of decision makers and over insights into the processes involved in their decisions. In this context, feedback is defined as information about the decision-making process or its outcome [1]. This framework makes it possible to analyse the components of judgmental accuracy in the process of decision making. It involves three basic elements: individual decision maker's judgement system, criteria and prediction by the individual decision maker of the criteria variables. Accordingly, four types of feedback are typically distinguished in this framework:

1. *Outcome* feedback: involves informing the individual decision maker about actual values of the criterion variable.

2. *Cognitive* feedback: provides information about the individual decision maker's judgement policy [e.g. information about the weights and functional forms relating task cues to one's (i.e. software developer of manager) actual predictions of the environmental variables].

3. *Task properties* feedback: provides the individual decision maker with information about statistical properties of the prediction task, typically information about judgement policies required to make optimal predictions, in some sense

4. *Model predictions* feedback: arises from the decision maker's stated judgement model and actual predictions to provide the decision maker (i.e. software developer or manager) with insight into his or her own judgement model. This type of feedback has been practically ignored in prior research.

These four forms of feedback in decision making process may enable the agents involved in the software process to combine quantitative data in ways that enhance their decision making ability in control development of comprehensive metrics.

In brief, we believe that in the study of feedback mechanisms, the following two questions need also to be addressed:

1. Does an adequate, in some sense, monitoring system for the global process exist? Can it be defined?

2. Would such a monitoring system be part of a discipline of feedback control that may improve software evolution process performance by means of, for example, improve responsiveness to changes in the business environment and in the technological elements?

It is now clear that further research is needed. In this respect, an exploratory study should be conducted in order to provide the empirical basis. It is hoped that the discussion in this paper was able to reflect some of the challenges and elements to be considered in advancing the study of the feedback phenomenon.

[2] See, for example, a companion paper [27] that discusses the potential impact of feedback control in software cost estimation.

Closing Remarks

Feedback loop mechanisms play a major role in software evolution. Feedbacks are either positive or negative. The degree to which a feedback relation is positive or negative depends on the function and parameters. Depending on the related elements (business environment, technological, organisational and humans) feedback mechanisms vary. Understanding these forces will open the way to practical model-based software evolution management, control and support for process improvement. Previous research within the FEAST/1 and FEAST/2 projects has focused on the general concept of feedback dynamics. However, major elements responsible for the feedback phenomenon and their different impacts in the software process must be identified, and quantified. In the light of such understanding, then, one can define appropriate feedback-system based metrics, an essential tool to improve software process management. Such metrics could be use to monitor and ultimately manage the software evolution process and also could provide the basis for strategy definition and corrective local action for such process. It is argued that empirical assessment and modelling of the presence and importance of those different dimensions in industrial software processes can lead to significant progress in the FEAST investigation.

Acknowledgements

Financial support from the UK EPSRC, grant number GR/M44101 (FEAST/2 Project), is gratefully acknowledged.

References

[1] Arunachalam, V. & Daly, B. *An Empirical Investigation of Judgment Feedback and Computerized Decision Support in a Prediction Task,* Accounting Management and Information Technology, Vol. 6, No. 3, 1996, pp. 139-56.

[2] Ashby, W. R. *An Introduction to Cybernetics,* Chapman & Hall, London, 1956.

[3] Ahton, R. H. *Effects of Justification and a Mechanical Aid on Judgment Performance,* Organisational Behavior and Human Decision Processes, Vol. 52, 1992, pp. 292-306.

[4] Beer, S. *The Brain of Firm: The Managerial Cybernetics of Organisation…,* John Wiley & Sons, Chichester, 1981.

[5] Belady, L. A. and Lehman, M. M. *An Introduction to Program Growth Dynamics, in Statistical Computer Performance Evaluation,* in: W. Freiburger (ed.), New York, 1972, Acad. Press, pp. 503 – 511 [Reprinted as chapter 6 in Lehman, M. M. and Belady, L. A. (eds.), *Program Evolution - Process of Software Change,* Academic Press, London, 1985].

[6] Boulding, K. E. *General Systems Theory: The Skeleton of Science,* Management Science, Vol. 2, No. 3, 1956, pp. 197-208.

[7] Checkland, P. *Systems Thinking, Systems Practice,* J Wiley, London, 1981.

[8] Checkland, P. & Scholes, J. *Soft Systems Methodology in Action,* J Wiley, London, 1990.

[9] Forrester, J.W. *Industrial Dynamics,* MIT Press, Cambridge, Mass, 1961.

[10] Kahen, G. *Assessment of Information Technology…: Appropriateness, Economic and Local Constraints, IT Characteristics and Impacts,* Int. Journal of Computer Applications in Technologies, Vol. 5, Nos. 5/6, 1995, pp. 325-33.

[11] Kahen, G. *Strategic Development, Technology Transfer and Strategic Technology Assessment in Changing Environments,* Proceeding of the First International Conference on Dynamics of Strategy, Surrey, UK, 1996, pp. 366-84.

[12] Kahen, G. *Devising the Convergence Strategy for Productivity Improvement: Effectiveness based on the Human Element,* Int. Journal of Materials and Product Technology, Vol. 12, No. 1, 1997, pp. 18-26.

[13] Lehman, M.M. *Programs, Cities, Students, Limits to Growth?,* Inaugural Lecture, in Imperial College of Science and Technology Inaugural Lecture Series, Vol. 9, 1970, 1974, pp. 211-229. Also in Programming Methodology, (D. Gries. ed.), Springer Verlag, 1978, pp. 42-62.

[14] Lehman M. M. *Laws of Program Evolution—Rules and Tools for Programming Management,* Proceedings of the Infotech State of the Art Conference, Why Software Projects Fail, 1978, pp. 11/1-11/25.

[15] Lehman M.M. *On Understanding Laws, Evolution, and Conservation in the Large Program Life Cycle,* Journal of Systems and Software, Vol. 1, No. 3, 1980, pp. 213-221.

[16] Lehman M.M. *Life Cycles and Laws of Software Evolution,* The Proceedings of IEEE Special Issue on Software Engineering, Vol. 68, No. 9, 1980, pp. 1060-76.

[17] Lehman, M. M. & Belady, L. A. *Program Evolution: Processes of Software Change,* Academic Press, London, 1985.

[18] Lehman, M.M. *Feedback in the Software Evolution Process,* keynote address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics, Dublin, Ireland, Sept. 7-9, 1994, and in Information and Software Technology, special issue on Software Maintenance, Vol. 38, No. 11, 1996, Elsevier, 1996, pp. 681-686.

[19] Lehman, M. M. and Stenning, V. *FEAST/1: Case for Support,* Department of Computing, Imperial College, London, UK, 1996, Available from links at the FEAST project web site http://www-dse.doc.ic.ac.uk/~mml/feast

[20] Lehman M.M. *Laws of Software Evolution Revisited,* Proceedings of EWSPT'96, Nancy, LNCS 1149, Springer Verlag, 1997, pp. 108-124.

[21] Lehman, M. M. *Feedback in the Software Process,* Position Paper, SEA Workshop: Research Directions in Software Engineering, Imperial College, London, 1997, April 14 – 15th.

[22] Lehman M. M. *FEAST/2: Case for Support,* Department of Computing, Imperial College, London, UK, 1998, Available from links at the FEAST project web site http://www-dse.doc.ic.ac.uk/~mml/feast

[23] Lehman M. M., Perry, D.E. and Ramil, J.F. *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution,* 1998, in Proceedings of the Fifth International Metrics Symposium, Metrics'98, Bethesda, Maryland, Nov. 20-21.

[24] Osterweil, L. *Software Processes Are Software Too,* Proceedings of the 9th International Conference on Software Engineering, 1987, pp. 2 - 12

[25] Paulk, M. C. et al *Capability Maturity Model for Software,* Version 1.1, Software Engineering Institute Report CMU/SEI-93-TR-24, 1993.

[26] Raffia, H. *The Art and Science of Negotiation,* Harvard University Press, Cambridge, 1982.

[27] Ramil, J.F. (2000) "'Why COCOMO' Works Revisited or Feedback Control as a Cost Factor", submitted to FEAST 2000 International Workshop on Feedback in Software and Business Processes, July 10-12, Imperial College, London

[28] Te'eni, D. (1991) "Feedback in DSS as a Source of Control: Experiments with the Timing of Feedback, Decision Sciences, Vol. 22, pp. 644-55.