

Can the Legacy Syndrome be Avoided?-Legacy Systems in the Context of Software Evolution

Taking a Global View of Legacy Systems, 7th SEBPC Legacy Workshop, Oxford, 30 August 1999

G Kahen M M Lehman J F Ramil

Dept. of Computing

Imperial College

180 Queen's Gate, London SW7 2BZ

tel +44 171 594 8216 fax +44 171 594 8215

{gk,mml, ramil}@doc.ic.ac.uk

<http://www-dse.doc.ic.ac.uk/projects/feast>

Computerisation involving, *inter alia*, software acquisition and deployment, is generally driven by an intention to solve a problem, to do things better in some sense, to modify or improve some aspect(s) of some behaviour(s) in the application domain. Increasing the speed or effectiveness of an operation, improvement of functionality or usability, decreasing operational costs have all been used for many years to justify investment in automation. More recently, business has focused on decreasing time-to-market and, more generally, in minimising response times. Businesses have also been seeking ways to increase their adaptation rate to and exploitation of changes in the operational environment, in the market place, in legislation or in technology for example. In this context, any significant mismatch between the software and its domain may bring a heavy penalty, loss of business opportunities or market share, for example. Thus, as business processes become ever more dependant on computers, the success of such improvement efforts is bound up with changes to the operational software by acquisition, upgrading or change. Meeting the goal of continuous adaptation is very likely to be hampered by the rate at which software can be safely and reliably adapted.

More fundamental than the need for software change driven by the need or desire for product and process improvement are the pressures for change that arise in *E*-type systems [leh85], that is, systems that are embedded and being used in an ever-changing real world domain. Such systems require continuing *adaptation* to changing needs and conditions to ensure the maintenance of user satisfaction. Several studies of a number of actively evolving software systems, most recently in FEAST/1¹ [leh96,98d,99], have demonstrated the presence of regularities in the behaviour of *E*-type systems evolution [leh85]. These regularities and other common evolutionary features have been captured and encapsulated in eight laws of software evolution [leh85]. These same sources also provide evidence in support, after minimal refinement, of six of these laws. The other two are neither supported nor negated [leh98b]. While the significance and impact of the laws cannot be considered here in detail, there appears to be a strong relationship between the drift towards becoming a *legacy systems* and a maintenance strategy that ignores the laws. For example, it follows from the first law, *continuing change* [leh85,98a], that for satisfactory system evolution the *software adaptation rate* has to be matched to the rates

¹ FEAST/2 [leh98a], a continuation of FEAST/1, started in April 1999.

of changes in the domain². Given a legacy system of undue complexity or with inadequate or incomplete documentation, this may not be possible.

The rate at which software can be adapted is determined and constrained by many factors. It is related, for example, to software characteristics such as growing complexity and other ageing effects [leh85,par94]. Equally it is a function of process characteristics such as maturity [ele99], and productivity. It is also related to the characteristics of the *global* process, that is, a wider system that includes the activities, technical and non-technical, of developers, users, marketeers and all others involved in bringing the product into active use. In some cases trade-offs between the adaptation rate and quality factors such as fault rate, flexibility, reliability and so on may be investigated in the search for improvement. But any immediate rate increase is very likely to produce negative effects at some time in the future as problems develop and as a consequence of the increased rate of working. It might be suggested that a higher software adaptation rate can be achieved by increasing the number of personnel involved in the maintenance process. Brooks' "adding manpower to a late software project makes it later" [bro75] suggests the contrary. And even if it were possible, the achievable size of the software team is likely to be limited by budget-related or technical and managerial considerations.

The precise extent to which the constraints imposed by, for example, personnel communication [bro75] and by feedback loops within the global and technical processes [leh94] become dominant to play an a major role in constraining the adaptation rate, varies according to circumstances. In general, however, there will be a *gap*³ between the functionality as required and the functionality as provided by the operational software. Legacy systems exemplify the extreme case in which, for many reasons, a system is not effectively changeable anymore. One class of legacy systems (or systems approaching that condition) are those in which the software adaptation rate has become significantly smaller than the rate required to satisfy stakeholder demands, that is the *required* rate. The challenge is to delay the onset of this situation.

An ability to estimate actual and required adaptation rates for a system is likely to find practical application, particularly in the context of large business-critical systems. The first may be derived from software evolution metrics such as *handled* and *handlings*⁴ [leh85,98c], that can be obtained, for example, from sources such as configuration management systems, code repositories and so on. That is, their availability need not depend on a prior metric collection and retention program [ram99] though having had one would help. Procedures needed to estimate and forecast the *required rate* are not yet available. They could be based, for example, on records of users' requests over time or over releases. From such data, estimates of the number of elements needing to be created, changed or removed can be obtained. From them, one can derive the required handled or handlings rate. Alternatively, estimation of the required adaptation rate may be based on an assessment of the volatility of the application domain using, for example, techniques similar to those applied in market forecasting [kot93]. It must however be noted that there is dynamic feedback linkage between the change in the application domain and change of

² Since the acts of installation and operation change the domain, achieving such a rate is further compounded. This represents an illustration of a more general linkage between the changing software and a changing domain. The implications of this linkage cannot, however be analysed within the limited scope of this paper.

³ The functionality gap during the software life cycle is discussed in [ara93].

⁴ We cannot discuss the definition of the metrics handled and handlings rates nor the choice of elements, LOCS, objects, files, modules, sub-systems, classes and so on, for example, on which they can be based. The interested reader is referred to [leh85,98c].

the software and of their respective rates of change. Thus the assessment of required rate is not straightforward but involves feedback-system like analysis [cha99,leh94,96,98a,98d].

The bottom line remains however that if it can be effectively established, the *rate difference* provides an indicator of rate divergence, to be used primarily as a monitoring device. Since both rates are likely to change over time the suggested indicator as defined will be dynamic in at least two dimensions. Nevertheless once monitored, and when relevant factors influencing the rates have been identified, it will provide the basis for a tool to apply a degree of control and management on the rate of change characteristics. Process improvements and attempts to avoid the *legacy syndrome* and its effects may become, at least partly controllable using the suggested indicator as a guide.

The suggested indicator clearly contrasts with others, such as *maintainability*. The latter has been based mainly on directly measurable software product attributes [gra97] and/or its immediate technical process [fen97]. In contrast, the indicator suggested here emphasises the importance of monitoring the relationship between the software process and the software operational domain from a global perspective. The distinction is crucial.

Summary

In summary, our position is that evolving a software system at a rate of change compatible with that of the application domain is vital if the decay leading to the display of legacy syndrome symptoms is to be avoided or, at least, delayed. We argue that factors within the global software process, that is, beyond the immediate technical software process have to be considered when addressing this issue. An indicator for the degree of rate match has been proposed.

Areas for further work, directly or indirectly related to the legacy system issue, and to methods for its monitoring and avoidance include the following:

- i. metric-based retrospective case studies of the evolution of software that have degenerated to legacy status, including identification and characterisation of patterns in metrics that led into that condition
- ii. case studies based on system dynamic modelling of the systems referred to in i, with the intent to explore underlying dynamic factors and mechanisms that have produced the legacy condition
- iii. characterisation and modelling of the application domain in terms of its volatility and change rate using market forecasting techniques [kot93], for example
- iv. practical methods for metrics-based monitoring, early warning and avoidance of the legacy syndrome, such as the rate difference indicator suggested above, with applicability, for example, to large business-critical software and their domains
- v. development of empirical models for controlling maintainability based on software adaptation rate, effort and cost in the context of software evolution.

Acknowledgements

Financial support from the UK EPSRC, grant numbers GR/K86008, GR/M44101, GR/L07437 and GR/L96561, is gratefully acknowledged.

References

- [ara93] Aranda RR, Fiddaman T and Oliva R, *Quality Microworlds: Modeling the Impact of Quality Initiatives over the Software Product Life Cycle*, American Programmer Vol. 6, No. 5, May 1993, pp 52–61
- [bro75] Brooks FP, *The Mythical Man-Month*, Addison-Wesley, Reading, MA, 1975, also as 20th Anniversary Edition, Addison-Wesley, Reading, MA, 1995
- [cha99] Chatters BW, Lehman MM, Ramil JF, Wernick P, *Modelling a Software Evolution Process*, ProSim'99, Softw. Process Modelling and Simulation Workshop, Silver Falls, Oregon, 28-30 June 99
- [kot93] Kotler P and Armstrong G, *Marketing: an Introduction*, Prentice-Hall Int., 1993
- [ele99] El Emam, K and Madhavji NH (eds.), *Elements of Software Process Assessment and Improvement*, IEE Computer Soc., Los Alamitos, CA, 1999
- [fen97] Fenton NE and Pfleeger SL, *Software Metrics - A Rigorous & Practical Approach*, 2nd edition, ITP, London, 1997
- [gra97] Granja-Alvarez JF and Barranco-Garcia MJ, *A Method for Estimating Maintenance Cost in a Software Project: A Case Study*, Software Maintenance: Research and Practice, Vol. 9, 1997, pp 161 - 175
- [leh85] Lehman MM and Belady LA, *Program Evolution - Processes of Software Change*, Academic Press, London, 1985
- [leh94] Lehman MM, *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
- [leh96] Lehman MM and Stenning V, *FEAST/1: Case for Support*, Department of Computing, Imperial College, March 1996, <http://www-dse.doc.ic.ac.uk/projects/feast>
- [leh98a] Lehman MM, *FEAST/2: Case for Support*, Dept. of Comp, Imp. Col., Jul. 1998, available from FEAST web page, see ref. [leh96]
- [leh98b] Lehman MM, Perry DE and Ramil JF, *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution*, Proc. Metrics'98, Bethesda, Maryland, Nov. 20-21, 1998, pp 84 - 88
- [leh98c] Lehman MM, Perry DE and Ramil JF, *Implications of Evolution Metrics on Software Maintenance*, Proc. International Conf. on Software Maintenance (ICSM'98), Bethesda, MD, Nov. 16-20, 1998, pp 208 - 217
- [leh98d] Lehman MM and Ramil JF, *Feedback, Evolution and Software Technology - Some Results from the FEAST/1 Project*, Invited Keynote Lecture, ICSEA, 11th Int. Conf. Softw. Eng. And its Applic., Preprints, Vol. 1, Paris, December 8-10, 1998, pp 1 - 11
- [leh99] Lehman MM and Ramil JF, *Metrics-based Process Modelling with Illustrations from the FEAST/1 Project*, August 1999, to appear in Bustard D. (ed.), System Modelling for Business Process Improvement, Artech House, Norwood, MA.
- [par94] Parnas DL, *Software Aging*, Invited Plenary Talk, ICSE'16, May 16-21, Sorrento, Italy, 1994, pp. 279 - 287
- [ram99] Ramil JF and Lehman MM, *Challenges facing Data Collection for Support and Study of Software Evolution Processes*, Position Paper, Proc. ICSE 99 Workshop on Empirical Studies of Softw. Development and Evolution, Los Angeles, May 18, 1999, pp 28 - 32