

Challenges facing Data Collection for Support and Study of Software Evolution Processes

Position Paper

Submitted to ICSE 99 Workshop on Empirical Studies of Software Development and Evolution
Los Angeles, May 18, 1999

JF Ramil

Dept. of Computing
Imperial College

180 Queen's Gate, London SW7 2BZ

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

jcf1@doc.ic.ac.uk

<http://www-dse.doc.ic.ac.uk/~jcf1/>

MM Lehman

Dept. of Computing
Imperial College

180 Queen's Gate, London SW7 2BZ

tel +44 171 594 8214 fax +44 171 594 8215

mml@doc.ic.ac.uk

<http://www-dse.doc.ic.ac.uk/~mml/>

Introduction

A recently completed study [leh96] investigated the FEAST hypothesis. This asserts, *inter alia*, that software processes are feedback systems. The project was based on gathering, analysis, modelling and interpretation of empirical data reflecting the evolutionary behaviour of five evolving software systems from four industrial collaborators. An outcome of the project was a better understanding of *E*-type [leh85] software evolution processes. Considerable insight into a role of metrics in software evolution studies was also gained. The investigation will be continued in a successor project [leh98a].

Evolutionary behaviour relates to attributes of relevant software processes, their products and relevant domains. Attributes such as system size, its complexity, effort applied, change rate (eg handled, handlings [leh85, leh98e]) amongst others, over time or pseudo-time (release sequence number, for example) [cox66], reflect aspects of the evolutionary behaviour. Measurement or estimation of these attributes has provided a basis for the study of software evolution dynamics [leh85,98d]. System size, for instance, played a dominant role in FEAST/1 studies [leh97b,98b,d]. The investigation of evolutionary behaviour requires studies based, at least in part, on system and process history, as reflected by historical data.

FEAST/1 has revealed, however, that the collection and storage of software process historic records is not a common industrial practice. This is in contrast to the situation in other fields, such as manufacturing, in which continuous process monitoring and record is seen as widespread practice and a topic of research. There is a need in the software process for enhanced data gathering and analysis, reducing human intervention and focusing human attention, not on the repetitive tasks, but on the interesting and rewarding aspects [mor98]. For study of the software evolution process, there is clearly a need for similar tools, leading to a more disciplined and inexpensive collection and analysis of historic data, for example. Industry should find such acquisition attractive, since it offers immediate benefits, a basis for better-informed decisions in the context of software evolution management. This acquires particular relevance in the context of evolution of large business-critical software. Academia would have at their disposal richer and more reliable data sources that are vital to validate, refine and evolve theories and models of software evolutionary behaviour and evolution approaches and techniques. But if this has not been accomplished yet, if empirical data collection still requires major efforts, it has not been by chance. There are major challenges to overcome.

Some questions of interest are the following: What are the challenges and the possible approaches to achieve a more effective data collection for the study of software evolution processes? How can the *observability* of software process phenomena be enhanced?

After summarising the FEAST/1 results and FEAST/2 plans briefly, this position paper addresses in outline some aspects of the above questions.

FEAST/1 Results and FEAST/2 Plans

The FEAST/1 (*Feedback, Evolution and Software Technology*) project (Oct 96 - Sept 98) investigated the FEAST hypothesis which states that: "*E*-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems. They must be treated as such to achieve significant improvement for other than the most primitive processes" [leh96]. FEAST/1 collected metrics data from five software systems being evolved by their respective organisations¹ covering periods of, in some cases, up to 15 years of evolution. Detailed goals were stated in the project proposal [leh96]. Results [leh97a,b,98b,c,d] can be summarised as follows:

- Support for the FEAST hypothesis [leh98b]
- Strong similarities in growth trends of different systems [leh98b]
- Black box [leh97b,98b] and system dynamics [eg cha99] models of software evolution
- Support for laws of software evolution [leh85,97b,98b], requiring only minor refinements of the latter
- Improved insight into metrics of software evolution [leh98c]
- Seeds for development of software evolution planning, management and control tools [leh98a,ram98]

A successor project, FEAST/2, due to start in March 1999 involving also a new industrial collaborator², plans to broaden and advance the study. Its proposal includes, *inter alia*, a list of topics for further research [leh98a].

The Role of Data

The above goals were achieved despite the intrinsic difficulty in locating and obtaining historical data relevant to the systems studied and the ubiquitous need for cleansing and pre-processing of the data ultimately located. Typical sources of raw data were configuration management systems and, more directly, source code files (change logs in program headers). Only in one of the systems studied there was in place a systematic mechanism for registration of changes to the source code. Even in this case substantial pre-processing and implementation of scripting tools in Perl was required [cha99].

Statistical experimentation in software evolution dynamics is severely constrained. Input manipulation has to satisfy certain mathematical conditions, such as *sufficiently exciting* [wel91], so that reliable information about process dynamic characteristics will be obtained. This is achievable and a common practice in certain technical domains (eg chemical plants), but is severely restricted in socio-technical domains such as the software process. Most of the times it is even unthinkable. Cost, resource, company policy and even ethical reasons and constraints play a major role. One has to rely mainly on observation of real process data to validate theory, models and interpretations. This at times imposes serious limitations. During FEAST/1, for example, a number of questions which arose could not be accurately answered because lack of appropriate data. This was the case when trying to explore the factors underlying a discontinuity, a 'change point', in the VME Kernel system growth trend [leh98b]. This situation wishfully suggested the idea of somehow establishing a process recording 'black box', analogous to the well-known device installed on airliners. This software process 'black box' would record the relevant attributes, in some sense defined, for later study and enquiry. Unfortunately the analogy appears to break down very quickly since one would surely wish to consult such a 'black box' not only after an incident (process and/or product break-down?) but at any time or even continuously to enhance software process monitoring and related decision making. Does it really deserve more serious consideration? What are the challenges and implications behind this suggestion?

¹ ICL, Logica, Lucent Technologies and Matra - British Aerospace Dynamics

² BT Labs

Some Challenges in Data Collection

The software process is a process of intensive information transformation [leh84]. Transformations are performed by humans on *documents*³ using *tools* (eg text editors) or directly by the tools (eg compilers). Since these activities typically are, in one way or another, computer supported the focus here is data collection addressing what is reflected by the content of the computers being used in the software evolution process⁴. Computer-supported data collection could be achieved by establishing a communication link between the process mechanisms and a data-logger⁵ which would systematically collect and record data. This seems to be technically feasible taking into account the current state of networking and distributed systems technology. A major challenge would be, however, to agree, for instance, in a set of standards, to decide what to capture, at which sampling rate and degree of granularity and what would be the role and nature of human participation in the data collection process.

A list of metrics of interest, identified and defined by a top-down procedure, is one feasible approach. The list would, for example, be generated using the goal-question-metrics (GQM) paradigm [bas84]. One has to realise, however, that implicitly or explicitly underlying any list of measures is a theory. This underlying theory suggests what is relevant (to be measured) and what is not (to be left out of the observation process). This leads to the danger that relevant aspects of the phenomena will be left out of the list, never captured, since they have been considered irrelevant 'from scratch'. This problem is not so evident or significant in the monitoring of other processes, such as manufacturing, in which a well-accepted underlying theory of the process will exist and will suggest a measure set. But even recognising the danger of omitting significant measures, the list may clearly be extended and refined, as understanding progresses and other parts of the process are recognised as being relevant, or as process portions initially monitored appear to have ceased to be relevant. An additional problem of the top-down approach is that it may end up with metric definition which are highly process dependent. As changes in the software process occur many metrics may become invalidated in their assumptions or irrelevant. One could expect process changes to happen many times over the several years of system life.

Alternatively, one may adopt a bottom-up approach. Were it possible, one could run the historical process in a 'slow-motion' fashion and obtain, from this more genuine raw data, almost any metrics one would like to define *a posteriori*. Clearly a 'brute force' solution based on storing every single event and change of state in the computers used for evolution is basically impractical. Given that approach, gigabytes, or even terabytes, of raw data would accumulate very quickly. Even at a coarse level of granularity in the definition of state, this approach does not seem to be either cost-effective or even technically feasible (yet) for nontrivial processes. More realistically, one could store snapshots of all (or at least the most relevant, including, of course, source code) documents at regular intervals (say, monthly), and more particularly at the moment of release, when the system, at least, is uniquely defined by the released code and documentation. Decreasing storage costs make this option attractive. Tools will be required to extract from this sequences of documents the desired, *a posteriori* defined metrics. In the light of FEAST/1 experience this appears to be a helpful approach. Some emerging technologies such as

³ Documents refer here to the different models of the application, including requirements, specifications, high-level and low-level designs and code [leh84], including development and user documentation. Tools refer, in general, to all support tools and CASE tools being used.

⁴ It is recognised that important process activities, such as people's mental activities, meetings and so on, are not so directly recorded. For the moment it is assumed that, in the context of software evolution phenomena, information within the machine of whatever form that is applicable in the evolution process will, sooner or later, reflect these activities.

⁵ This data-logger will follow relatively simple procedures. Collection of metrics which require direct human intervention and interpretation, such as function point counts, are not considered here, though one could argue that the data-logger could be provided with more advanced capabilities. It could be configured to simulate human expert features. One should not reject the use of intelligent agents technology for this purposes in the not too distant future.

semi-structured databases [suc98] (emerging very much as a response to needs imposed by web-based documentation issues⁶) and intelligent document processing [mal97] may help to obtain measures even from document formats which lack of a well defined structure.

Following a different approach, one could attempt to establish a set of *primary* measures⁷, from which other attributes could be derived. That is, having recorded the primary measures associated with a given process, one could reconstruct evolutionary behaviour depicted by them and by other *secondary* measures. The latter would be obtained as some combination of the primary measures. Here we have also the problem that a set of primary measures implies a theory, and hence, constraints on the observability of the process, though one could argue that such constraints will be less than in the case of top-down defined measures. This deserves more careful examination.

It is hoped that the above discussion has suggested *prima facie* the challenges ahead. Space limitations have not allowed here the consideration of other related aspects, such as computer supported data analysis and recognition of patterns and trends [eg mor98] and the opportunities opened by enhanced data collection in the application of data mining [leh98a] and in process simulation [leh98a,ram98].

Computer-Supported Software Process Data Collection and Monitoring

The current focus in academia and industry has been on using computers and tools *actively* in the process. Many aspects of the process may benefit from this. Our contention here is that a relevant, yet not fully explored role of computers and support tools is a more 'passive role', *computer-supported software process data collection and monitoring*. This has to be achieved transparently without imposing additional constraints to those inherent to the process or demanding extra data collection effort from those involved (eg developers, testers, integrators). Human creativity and process change should not be constrained, when possible, by other considerations, such as observation, measurement and decision support needs. We suggest that minimising human intervention for purposes of data *collection* and focusing human effort on data *analysis* and *interpretation* is the way forward. The use of computers as suggested, however, raises ethical issues. For example, should, and in affirmative case to which extent, the private workspace of developers and others involved be subject to automatic monitoring? This and other related issues would need appropriate consideration.

Final Remark

This paper has stated the position that data collection mechanisms addressing the monitoring of software process evolutionary behaviour can yield benefits to industry and academia. The former will benefit by having a basis for better-informed decisions, the second by having at their disposal empirical data for more reliable and effective theory and model validation. In this context, this position the paper has outlined some of the related challenges and approaches.

Much work is still needed to fully understand the underpinning mechanisms, the sources of the regularities, such as those observed in FEAST/1, for example, and the more appropriate modelling paradigms [eg ram98] to encapsulate them. A long term goal could be an integrated decision support framework for software evolution and its process, together with more accurate process performance

⁶ Some organisations are now beginning to experience an increasing need to update large and complex internet and intranet sites as they rely on them more and more to support their operations. This phenomenon shares some commonalities with the needs for continuing software maintenance experienced since the beginning of the computer age. An interesting cross-fertilisation of ideas and techniques between the two fields can be expected [bre98].

⁷ Since in principle primary measures do not depend on measures of a lower order they are related to the *zeroth level concepts* discussed in [mal97].

estimates and a fuller understanding of and command on the global process dynamics. Any progress in software evolution process monitoring is seen an important contribution towards the above goal.

Acknowledgements

Grateful thanks are due to Dr. Stephen McKearney (BT Labs and Univ. of Bournemouth, UK) for having introduced the semi-structured databases topic to one of the authors. Dr. Dan Suciú (ATT Research) replied very kindly an enquiry and gave useful indication on the use of semi-structured databases. The authors acknowledge financial support from the UK EPSRC, grant number GR/K86008 (FEAST/1). One of the authors (JFR) acknowledges financial support from BT Labs, Martlesham, UK (Short-Term Research Fellowship).

References

- [bas84] Basili VR and Weiss D, *A Methodology for Collecting Valid Software Engineering Data*, IEEE Trans. Softw. Engineering, SE-10 (6), 728-738
- [bre98] Brereton P, Budgen D and Hamilton G, *Hypertext: The Next Maintenance Mountain*, Comm. ACM, Dec. 1998, Vol. 31, No. 12, 49 - 55
- [cha99] Chatters BW, Lehman MM, Ramil JF, Wernick P, *Modelling a Software Evolution Process*, submitted for publication, Jan. 1999
- [cox66] Cox DR and Lewis PAW, *The Statistical Analysis of Series of Events*, Methuen, London, 1966
- [leh84] Lehman M M, Stenning V and Turski W M, *Another Look at Software Design Methodology*, Dept. of Comp., Imp. Col., Res. Rep. 83/13, June 1983. Also, Software Eng. Notes, v. 9, no 2, April 1984, 38 - 53
- [leh85] Lehman MM and Belady LA, *Program Evolution, - Processes of Software Change*, Academic Press, London, 1985,
- [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/~mml/feast>
- [leh97a] Lehman MM, *Process Models - Where Next?*, "Most Influential Paper of ICSE 9 Award", Proc. ICSE 19, Boston, 20 - 22 May 1997, 549 - 552
- [leh97b] Lehman MM, 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, 20 - 32. Also as: *Process Improvement - The Way Forward*, in Elements of Software Process Assessment and Improvement, K El Eman and N H Madhavji (eds.), IEEE CS Press, 1999
- [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, 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, 208 - 217
- [leh98d] Lehman MM and Ramil JF, *Feedback, Evolution and Software Technology - Some Results from the FEAST/1 Project*, Invited Keynote Lecture, 11th Int. Conf. Softw. Eng. And its Applic., Preprints, Vol. 1, Paris, December 8-10, 1998, 1 - 11
- [mal97] Malerba D, Semeraro G and Esposito F, *A Multistrategy Approach to Learning Multiple Dependent Concepts*, in Nakhaeizadeh G and Taylor CC (eds.), Machine Learning and Statistics, Wiley, New York, 1997, 87 - 106
- [mor98] Morrill JP, *Distributed Recognition of Patterns in Time Series Data - From Sensors to Informed Decisions*, Comm. ACM, May 1998, 41, 5, 45 - 51
- [ram98] Ramil JF and Lehman MM, *Fuzzy Dynamics in Software Project Simulation and Support*, Proc. EWSPT'6, Weybridge, UK, 16 - 18 Sep. 1998, LNCS 1487, Springer Verlag 1998, 122 - 126
- [suc98] Suciú, Dan, *An Overview of Semistructured Data*, SIGACT News, Dec. 98, vol. 29 , no. 4 , 28-38
- [wel91] Wellstead PE and Zarrop MB, *Self-Tuning Systems - Control and Signal Processing*, Wiley, Chichester, UK, 1991