

System Dynamics Models of Software Evolution Processes

Manny Lehman

Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
England
mml@doc.ic.ac.uk

Paul Wernick*

Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
England
pdwl@doc.ic.ac.uk

1. Background and Context

The work presented here forms part of a wide-ranging study of the role and impact of feedback in software evolution. The objectives of this investigation are as follows [8]:

- provide objective evidence that feedback phenomena and the consequent system dynamics have substantial impact in the software process;
- model global process feedback structures and mechanisms and identify their properties;
- demonstrate that feedback phenomena can be exploited in both managing and improving industrial processes; and
- develop foundations for a theory of software process and software evolution.

The investigation includes both statistical analysis of data from software processes [10], and the building of System Dynamics (SD) [4, 5] models of those processes. It seeks to combine results from both these strands, to determine the degree to which they support the hypothesis [8] that, as for other complex feedback systems, the dynamics of real world software development and evolution processes will possess a degree of autonomy and global stability.

The processes being investigated are dynamic in nature, incorporating both forward paths and feedback loops. The processing and control mechanisms associated with these feedback loops involve people, individually and in groups as managers/implementers. All observe, interpret, communicate, decide and act or refrain from acting on the basis of their overall perception, instructions, habits and other inclinations, experience and biases [15]. Much of the feedback control is indirect, unplanned, or even unconscious. Some of the feedback mechanisms are non-deterministic. Modelling or simulation

must therefore replace the analytical tools of control theory, and SD is therefore considered appropriate for the investigation. Convincing support for the hypothesis requires that the analysis and its associated predictive models must be quantitative. Use of computer-based SD environments permits models to be calibrated and their quantified results compared with actual data. Vensim [14] is being used for the current studies.

This paper describes initial models developed for early FEAST/1 process structure studies. Each model describes a specific real world industrial software process and its effects on its products. Whether a generic model over several systems within an organisation, or over more than one organisation, can be developed is, as yet, an open question. At this stage, it has not been possible to identify the truly relevant aspects of the global software process and its organisational and wider environments, or to determine an appropriate level of abstraction, for generic models to be of use. Current efforts are therefore directed towards the building of instance-specific SD models, as described here, for specific industrial collaborators, processes and products. Each model is designed to reflect the actual evolution process associated with a single product. Later research will seek to abstract elements common to all from these (and other similar) models, and to construct one or more generic models.

The approach taken to the modelling reported here is based on two fundamentals, *viz.* an emphasis on identifying control points in the process, and simplicity in the models. The search for control points in the global software process has identified as candidates the discrepancies, or gaps, between actual and desired attributes or measures of products and processes (cf. [12, p.308 *et seq.*], [2, p.59, fig. 8]). The value of each gap alters over time as a result of changes in either the product or process or users' or developers' expectations of that product or process. These changes are often carried through the global software process by information feedback. Simplicity is also being emphasised in model building, the goal being to build and calibrate simple models and to understand their high-level behaviours, particularly those elements related to the dynamics of software evolution. These models seek to replicate observations made on real world processes, employing the smallest possible number of entities,. Simplicity in models eases the tasks of identifying, calibrating and analysing the

* now at Department of Computer Science, Birkbeck College, University of London. Email: pwernick@dcs.bbk.ac.uk

effects of process feedback loops. The strategy for model building has therefore been top down.

The models described here are to be seen as work in progress. The next step is intended to yield models calibrated against actual process and product data, describing in qualitative and quantitative terms current software processes and their effect on the evolution of the products they generate.

2. Long-Term Evolution of a Commercial Product : 20 years of the VME Kernel

The first model described relates to evolution of the kernel of VME, ICL's mainframe operating system [6]. The kernel includes low-level hardware interfacing routines, and higher-level routines to support the management of the computer's operations.

VME has been the operating system for ICL's mainframe computer range for more than 20 years, during which time it has been evolved to cope with all the changes which have taken place in the world of computing whilst maintaining its basic structural integrity. Over that time, the ICL organisational culture has remained basically stable, despite changes in organisational structure and ownership. Many of the staff who have worked on VME over its life to date are either still working for ICL or are otherwise still available for discussion or informal data trawling. In addition, 20 years' of system size and change data is available. A statistical study of the evolution of VME has previously been published [7]. VME therefore appears to be well suited to an examination of software product evolution.

The model (Figure 1) has been derived, in conjunction with ICL staff, with little structural change from a simple influence diagram of the kernel's growth. It is already providing ICL with useful insights into their process. Its structure is completed, and calibration to actual data is expected to be completed before the ICSE Workshop. It describes the sources, and ways of addressing, the gaps between supply and demand for power, capacity and other attributes of the kernel software, which have formed the main areas of concern for users and thus sources of pressures for enhancements to the product. Since the objective here is to develop a *single* model structure to explain the evolution of that product over its life, the long history of VME, and the changes in process, management structures, ownership, etc. over that time require a very high level of abstraction. It demands the factoring out of details of elements which have changed over that time, even if they are included as changing parameters. An example of this is release interval, the value of which has changed over time but whose effects on the process need to be taken into account in an unchanging model structure.

The model has two main loop systems. The loops to the left of the 'new capability creation rate' reflect the effect of previous releases of the kernel on demand for changes in subsequent releases. Exogenous pressures for changes will also be included as these are identified from actual data. This part of the model is three-valued (using Vensim's array facilities – [14], p.26 *et seq.*), maintaining separate values for each of the three

types of attributes, to allow for differences in demand, cost of implementation, etc. It incorporates the assumption that improving the kernel results, over time, in demands for further improvements, in addition to other exogenous demand for improvements. The loops to the right reflect the effects of existing software on future developments, including inertia of the existing product, and obsolescence of code as, for example, older hardware is no longer supported. It also includes the calculation of source code database size, which is the output parameter for model calibration. These two loops drive the software evolution process, in the sense that the gap between supply and demand for changes to the system is intended to be eliminated by system evolution over a time set by management. The achievement of this goal is limited by available resource and the ability to make changes to the system. The model is a continuous time model, using a weekly time base, and the release process is abstracted into that continuous timescale, rather than each release being modelled as a specific event. The calculations that emulated the production of software incorporate a delay to reflect the time required to implement desired changes. The release interval, which has changed considerably over the lifetime of the kernel in response to exogenous management decisions, is used in the model to calculate average delays through the user feedback process, from changes being made to the recognition of users' demands for further changes as a result of using the enhanced system.

As stated, the output metric being used to test the operation of the model is the size of the source code database. The actual data is measured in 'holons', which are broadly equivalent to source code modules. The source code database forming the basis of the calibration started life about 20 years ago as a transfer of existing code for the then-current product from a previous code database. The size of the delivered product is a subset of the source code database content, and would be preferable as an output metric. However, actual values for this are not available over the system's life span, whereas changes in the source code database size can be traced from the data in the database. Note that the size of the source code database is generally an *unmanaged* output, not deliberately controlled by the developers. It may therefore reflect actual process dynamics.

Once calibrated, the model will represent a complete, simulatable SD model. Quantification and calibration of the model requires both the calibration of input and output variables, and the development of expressions to calculate values for the levels and flows which form the model. The latter has been completed, and the former is currently in progress. It is expected to be complete by the time of the ICSE Workshop. Derivation of input and output variables will require various approaches. Some, such as change requests received, are obtained from system data files. Others can be inferred or compiled from the stored data. Yet others can only be estimated by observation of process, e.g. delays between release and feedback of change requests. Still others can only be estimated on the basis of the 'gut feeling' of those familiar with the process.

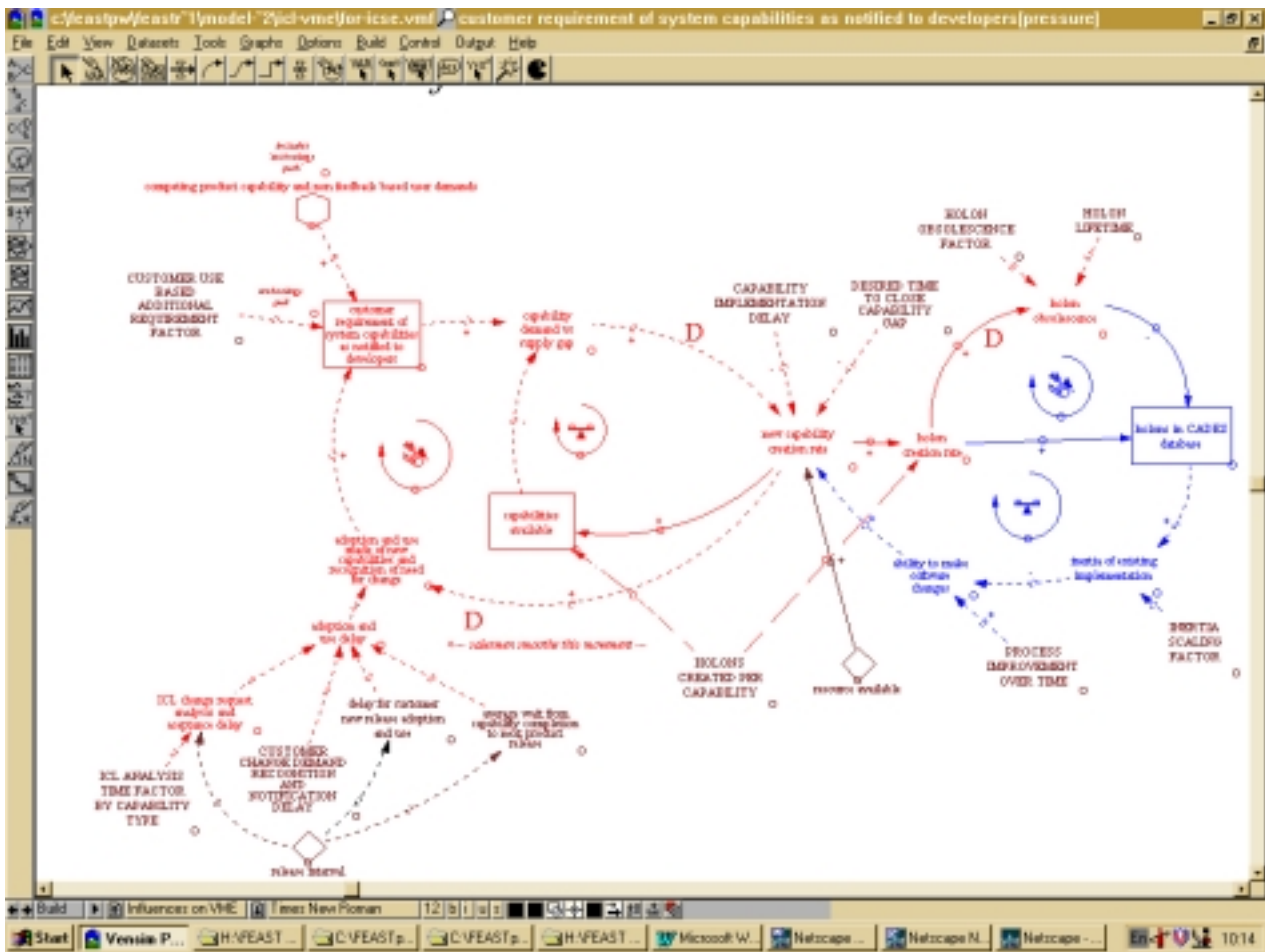


Figure 1: Evolution of The VME Kernel

3. Parallel Evolution of Specification and Product : Matra BAe Dynamics ('BAe')

This model (Figure 2), constructed with the assistance of BAe staff, places an embedded software process in the context of its product and financial constraints. The environment of the project is somewhat different from that described above. The goal is to research, design and implement the embedded software for a defence system being developed under a fixed price contract, with a fixed timescale for project completion. Payments are made under the contract on the basis of progress achieved as demonstrated in field trials. There is only one customer, the contracting authority.

This model, like that described above, is a continuous-time model, with the specific release and field trial process and schedules (except for the first release) abstracted. As before, there is an emphasis on identifying the effects of, and controls which close, the gaps between desired and actual process attributes. Here, gaps are defined with reference to the goal-oriented nature of this process, working towards an initially ill-defined definition of system behaviour. The model reflects gaps between desired and actual positions for the goal, both as the former moves in response to learning during the

development and trialling process, and as progress towards the current goal is evaluated in field trials. The effect of the looming final hard deadline on developers' ability and willingness to change the product completed to date has had to be taken into account. The effect of existing work on a software product in making further changes more difficult is reflected here as increased cost and slower completion. The need for developers to demonstrate progress towards the goal in order to receive stage payments necessitates the incorporation of BAe's financial objectives in the model, in the form of a cash flow gap and its sources and influences.

The current state of the model is that of an influence diagram, rather than a completed SD model. No quantification work has yet been carried out, either in developing equations for levels or flows, or to seek data for specific input and output variables. Questions currently remain as to the correct units of measure for each variable, and how variables' quantifications must be related to each other. However, it is expected that further analysis will provide sufficient insights to allow the quantification to succeed, and this analysis will be performed after the model structure has been agreed with the collaborator. Even at its present stage of development, the model is providing useful insights to the collaborator.

work on inspections [11], have also resulted in more complex models than those presented here.

There is a similarity in approach between the models described here and the high-level causal diagram underlying the Draper model [13, p.9], but their emphasis is on the internals of the process as shown later in their report, and therefore they model these aspects in greater detail than is done here. A comparison with their work also shows that the models presented here are independent of any specific software development process model; the Draper work is based explicitly on the waterfall model [2, p.4]. This suggests that it may be easier to develop generic models of the software process from simpler models, as here, than from complex models. The goal-gap mechanism underlies [2, fig. 5], and they have, as here, produced a model covering evolution of requirements over multiple product releases. However, their emphasis has been on specific aspects of the process, aiming to improve product 'quality', rather than a more generally-based examination of the long-term evolution of a software product.

6. Current Status of Work and Next Steps

The current state of this work is as follows. Each model is being examined by respective collaborator staff to ensure that its structure reflects their software evolution process. The VME model is currently being extended to the form of a complete SD model, only requiring calibration parameters and expected outputs to be calculated in order to allow simulation runs to be performed. It is intended that a complete SD model be described at the workshop. The BAe model is currently in the form of an unquantified influence diagram. The means of quantifying all elements, together with additional variables, constants, equations and other elements required for a full SD model, have yet to be identified. Progress towards this is expected by the workshop. In both cases, when the models are quantified, sensitivity analysis mechanisms reflecting the route used to obtain data for each variable will be employed to examine the importance of the influence of that variable on the model's behaviour.

The models presented here are preparing the ground for the next major stage in the FEAST research programme, namely that of the analysis of feedback influences. They will be used in that work when they are calibrated and demonstrate behaviour equivalent to those observed in (or, for ahistorical scenarios, expected of) actual real world software evolution processes.

7. Acknowledgements

We are grateful to the FEAST/1 industrial collaborators (Matra BAe Dynamics, ICL, Logica and MoD DERA), and particularly to Les Barker, Dave Nuttall, Mike Atterton, Bob Born and Paul Gilbert of Matra BAe Dynamics and Brian Chatters of ICL High Performance Systems for help in developing the models described. Thanks are also due to the other members of the FEAST/1 team, Juan Ramil, Dewayne Perry and Wlad Turski, and further thanks to Juan Ramil for

commenting on an early draft of this paper. The work reported here has been supported under EPSRC grants numbers GR/K86008 and GR/L07437.

8. References

- [1] Abdel-Hamid T. and Madnick S.E. Software Project Dynamics – An Integrated Approach. Prentice-Hall, Englewood Cliffs NJ, 1991
- [2] Aranda R.R., Fiddaman T. and Oliva R. Quality Microworlds. American Programmer, 6, 5 (May 1993), 52–61
- [3] Boehm B.W. Software Engineering Economics. Prentice-Hall, Englewood Cliffs NJ, 1981
- [4] Coyle R.G. System Dynamics Modelling. Chapman and Hall, London, 1996
- [5] Forrester J.W. Industrial Dynamics. Productivity Press, Cambridge MA, 1961
- [6] An Introduction to VME. ICL Literature and Software Operations, December 1989
- [7] Kitchenham B. System Evolution Dynamics of VME/B. ICL Tech. J.; May 1982; 42–57
- [8] Lehman M.M. and Stenning V. FEAST/1: Case for Support. ICSTM – DoC EPSRC Proposal. March 1996
- [9] Lehman M.M. Why COCOMO Works. Invited talk in absentia, COCOMO Symposium, October, 1996
- [10] Lehman M.M., Perry D.E., Ramil J.F., Turski W.M. and Wernick P.D. Metrics and Laws of Software Evolution – The Nineties View. Proc. Metrics '97 (Albuquerque, NM, 5–7 Nov., 1997)
- [11] Madachy R.J. System Dynamics Modeling of an Inspection-Based Process. Proc. ICSE 18, IEEE, 376–386
- [12] Roberts N. *et al.* Introduction to Computer Simulation, Productivity Press, Portland OR
- [13] Smith B. *et al.* The Software Process Model. Tech. Report IR&D #349, Draper Laboratory, Cambridge, Mass., 1993
- [14] Vensim Reference Manual, Version 1.62. Ventana Systems Inc., Belmont, MA, 1995
- [15] Wernick P. A Belief System Model for Software Development: A Framework by Analogy. PhD Thesis, Department of Computer Science, University College London, April, 1996
- [16] Winder R. and Wernick P. The Inductive Nature of Software Engineering and its Consequences. Proc. BCS Information Systems Methodologies Specialist Group/UK Systems Society Conference (Edinburgh, 1–3 September 1993) 431–443