A Policy Investigation Model for Long-term Software Evolution Processes

Paul Wernick and Tracy Hall

Centre for Empirical Software Process Research
Department of Computer Science, University of Hertfordshire
College Lane, Hatfield, Hertfordshire AL10 9AB, England
tel. ++1707 286323/284782; fax ++1707 284303

{p.d.wernick, t.hall}@herts.ac.uk

1 Objective of this Research

The work presented here is intended to develop a simple calibratable simulation model of a long-term software evolution process. This base model will act as a test-bed for the examination of effects of long-term software process 'improvement' proposals, since it will be possible to modify it to reflect the effects on the process of any particular proposed change.

The use of an existing simulation in this way is not itself novel. Existing work on the same lines (Tveldt and Collofello 1995; Haberlein, 2003) use Abdel-Hamid and Madnick's model (1991) as a base. However, they are examining the effect on a single development project as against the longer-term effect over many years and many releases considered here. In addition, the latter is a complex model, with many variables and a considerable number of inputs to calibrate. The base model presented here is intended to be as simple as possible whilst allowing the crucial variables to be manipulated. The simplicity has a number of advantages:

- the base model is easier to calibrate, requiring fewer inputs to be quantified;
- in the same way, it should be easier to reflect proposed process changes, since fewer changes to the model structure and/or fewer calibration inputs need to be taken into account;
- there is less potential for the model to fail to reflect reality after changes to go wrong when simulating a process change, since this inevitably results in taking a model outside its known behaviour envelope; and
- the evaluation and interpretation of results of proposed process changes is more easily understandable.

On the other hand, some of the subtleties in process changes may be missed with this approach.

2 The Base Model Described

2.1 Model Structure

The model presented in this section represents the structures, effects, inputs and outputs of a long-term software evolution process. To allow the adoption of different methods and approaches to be simulated it is important that the base model is simple. A sufficiently simple base model will allow the simulation of generalised software development and evolution activities without any bias for or against any particular method, toolset or approach.

Our proposed base model is shown in Figure 1. It is derived from ideas, structures and values from our previous software evolution simulation models. In particular, it incorporates feedback structures representing both the generation of new requirements (Chatters *et al.* 2000) and the correction of faults in previously-implemented requirements (Wernick and Lehman, 1999).

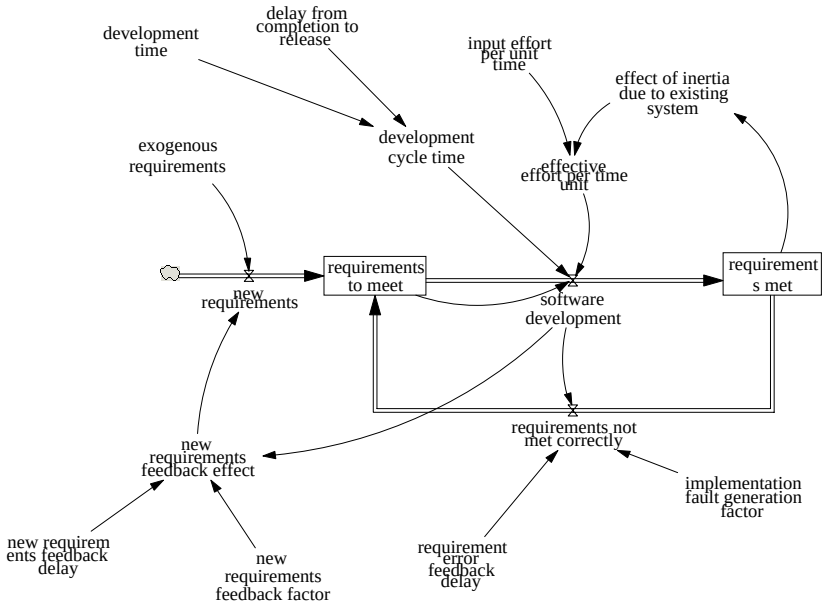

Figure 1: the base model structure

The model works as follows. The software process is viewed as a mechanism to convert requirements which need to be met into requirements which have been met and fielded to users. The rate of software development, more specifically the rate at which work to be done enters the delay representing the software production process, is the lesser of the amount of work available to be started and the resource available to start that work. This starting rate is subjected to a third-order time delay function to represent the time taken to perform the development work. It is further delayed as completed requirements have to wait until the next release of the software is delivered to its users. The effective effort available is reduced over time by an inertia-like effect (Wernick and Hall, 2002) in a manner related to the size of the existing system (Turski, 2002).

As a result of fielding the software, two things happen. First, developers are told that some requirements have not been met properly, typically due to bugs in the fielded software and/or mistaken interpretation of the users' requirements by the developers (*cf.* Wernick and Lehman, 1999). Secondly, new requirements arise due to the possibilities of additional uses of the system specifically enabled by the previously-fielded software (*cf.* Chatters *et al.*, 2000). In addition, there can arise exogenous events, i.e. changes in the environment within which the system is used and for which the system has to be modified. For the model calibration, the value for exogenous events has been set to 0, but an input for these has been included in the model for completeness.

2.2 Model Calibration

The calibration inputs for the evolutionary growth described in the model are based on actual figures for the evolution of the VME mainframe operating system as described in Chatters *et al.* (2000). This work on VME is useful to our model building as both input values and outputs against which to check them are available. The time delays used in the model are averages of the variable delays used in the VME model. These delays are

- time taken for the conversion of requirements into code ready to be released: 8 months;
- delay from the completion of this until next release is delivered to users: 5 months, and
- delay for user adoption of the new release, and for the feeding back of new requirements or of errors requiring fixing: 8 months.

In the absence of actual data:

- the size of the system at the start of the simulation run is taken arbitrarily to be: 200 requirements units;
- the initial input workload of recognised but unfulfilled demand for new requirements has been set arbitrarily: 40 requirements units; and
- the input value of effort available to turn requirements into met requirements, is set initially to a constant value of 1 requirement unit per month; the development team for the VME system was constant over the time simulated in the base model. This is multiplied by a factor reflecting the effects of the existing system size on the ability to evolve it. This results in a reduction of effective throughput over time, due both to the inertia of the existing system and to a reduction in system-wide knowledge (Wernick and Hall, 2002). The overall effect on effort due to system size is calculated as a multiple of the inverse cube (Turski, 2002) of that size.

For the VME model the flow of completed requirements was multiplied by a factor of 0.6 to generate the flow of new requirements. In the model described here, we have multiplied the output of the software production process delay function by 0.64 for the generation of new requirements and 0.16 to produce the rate of error feedback. The ratio of new to incorrect requirements is thus 4:1 in accordance with Pressman's (2000: 849) conclusion that fixing mistakes comprises 20% of maintenance work, the other 80% being system adaptation and enhancement for users or future use.

Finally, the simulated time over which the model runs is 156 months, reflecting the period of time over which the VME data is available.

2.3 Selecting the Reference Mode

Product metrics which might be selected as a reference mode for a simulation of long-term software evolution include physical system size (cf. Chatters *et al*, 2000) and the number of units of requirements implemented and delivered (cf. Wernick and Lehman, 1999). We decided to use the latter, since it is a more direct reflection of the ability of a system to do useful work. It also avoids issues related to specific methods of code size measurement**.**

The output trend for requirements met and fielded is shown in Figure 2. This shows the typical growth in product size over time reflecting a falling-off in the rate of newly-implemented requirements. This represents a similar reduction in the rate of addition of useful features to the system (*cf.* Wernick and Lehman, 1999; Chatters *et al.*, 2000). An equivalent trend for actual growth in VME from the same arbitrary starting point, in terms of number of arbitrary requirements units each of which is calculated as the same number of new modules added to the system, is shown on the same axes in Figure 2. The smooth trend of the simulation output contrasts with the more bumpy actual software product evolution trend, due to the abstraction in the model of detail in the real-world software evolution process which causes higher-order dynamic behaviour.

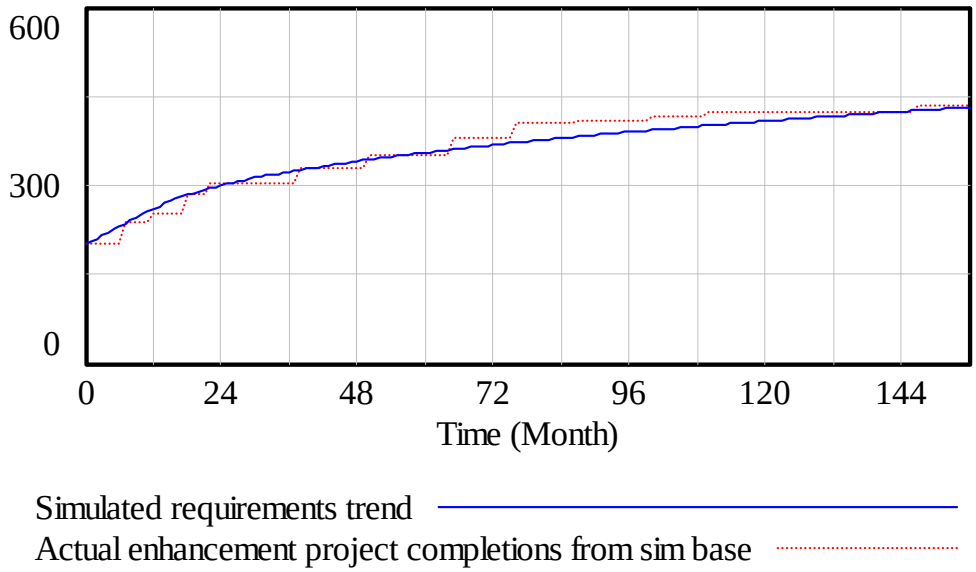

Figure 2: requirements met over time by the base model against normalised actual values

2.4 Policy Inputs

The effects on the long-term behaviour of a software evolution process of a specific improvement initiative is reflected in the model through changes to the values of a small number of input variables. Initiatives currently being proposed which might be examined in this way include pair programming and software inspections.

The policy inputs to the base simulation are:

- effectiveness of staff working, reflected in the rate of starting work on new enhancements/fixes
- speed of development cycle, reflecting for example the ability to speed up the development process from requirements to code ready to ship and/or reduced delays from completion of an enhancement to incorporation in a fielded release
- maintainability of the system, represented as the inertia of the existing system making any change more difficult, and reflected to the size of the exiting system (Wernick and Hall, 2002)
- effectiveness of the process in correctly identifying requirements and converting these correctly into system functions delivered to users

In the current simulation model, changes to these policy variables are hard coded; it will be possible to amend the model to allow its behaviour to be switched between the base case and that incorporating the change.

The rate of generation of new requirements due to feedback is assumed to be entirely due to forces outside the software process, and thus is not amenable to change due to any process changes

3 How to Use the Base Model

The base model is intended to be used as follows. It should if necessary be recalibrated against the real world process for which a process improvement is proposed; otherwise, the current calibration can be used. The base reference mode should then be derived. The model policy inputs should then be modified to reflect the anticipated effect of the process change. The simulation can then be run and the outputs compared. The equations for the base model will be included in a final version of this paper to facilitate its use by others.

An example of this process, investigating the long-term effect on software product size of the introduction of pair programming, is currently in progress.

4 Two Caveats

Two issues are causes of uncertainty in the current model and its application. These are:

- the causal mechanisms and actual effect of *inertia* in the system being simulated, which is modelled as acting to reduce the ability to evolve the system over time. This uncertainty, and the difficulty in calibrating some aspects of a simulation model which reflect abstract phenomena, have been previously noted by Haberlein (2002) in the context of calibrating a value for ‘trust’; and
- the assumption, implicit in the model when amended for any proposed process improvement resulting in an increase in the rate of adoption and use of the system, that this increase in functionality can and will be adopted and employed by the system’s users and will result in a corresponding increase in the rate of change demand fed back into the software process.

5 Future work

The most important work required on the base model is that part intended to reflect the effect of the inertia of the existing system in slowing further evolution. The current simulation structure is a simple mathematical representation of a theory (Wernick and Hall, 2002) of this phenomenon. Future research will hopefully identify the causal strucure underlying this representation, and allow a calibration based on real-world metrics rather than the current calibration to fit.

References

Abdel-Hamid TK and Madnick, SE (1991) *Software Project Dynamics – An Integrated Approach*, Prentice-Hall; Englewood Cliffs; New Jersey.

Chatters BW, Lehman MM, Ramil JF and Wernick P (2000) Modelling A Software Evolution Process, *Software Process: Improvement and Practice*, **5** (2–3), 91–102.

Forrester JW, *Industrial Dynamics*; Productivity Press; Cambridge, MA (1961).

Haberlein T (2003) A Framework for System Dynamic Models of Software Acquisition Projects; *Proc.* ProSim 2003; Portland, OR.

Tveldt JD and Collofello JS (1995) Evaluating the Effectiveness of Process Improvements on Software Development Life Cycle Time via System Dynamic Modelling; *Proc.* COMPSAC ’95; 318–325.

Turski WL (2002) The Reference Model for Smooth Growth of Software Systems Revisited"; *IEEE Trans. Software Engineering*; 28(8); 814 – 815.

Wernick P and Hall T (2002) Simulating Global Software Evolution Processes by Combining Simple Models: An Initial Study, *Software Process: Improvement and Practice*; **7**, 113–126.

Wernick P and Lehman MM (1999) Software Process Dynamic Modelling for FEAST/1, *J. Systems and Software*, **46** (2/3); 193–202.

Williams L, Kessler RR, Cunningham W and Jeffries R (2000) Strengthening the Case for Pair Programming; *IEEE Software*; **17** (4); 19–25.