Getting the Best out of Software Process Simulation and Empirical Research in Software Engineering

Paul Wernick and Tracy Hall
Systems and Software Group, School 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

Abstract

This position paper sets out our views on the need to use simulation and quantitative experiments in combination in order to maximise the benefit of both to software engineering research. Each approach should be used to overcome weaknesses in the other in attempting to predict the behaviour of software processes when new or modified processes, tools or techniques are employed. We also express our concern at the frequently-encountered use of the term 'experiment' to describe quantitative simulation-based investigations.

1. Introduction

In this paper we describe what we believe to be a weakness in current approaches to the use of simulation in software engineering. This is the often-unstated uncertainty of the results obtained from what-if simulation runs. We give reasons for this uncertainty, and suggest that this problem can be to a large extent resolved by the application of empirical experiments in combination with simulation-based investigations. Such an approach will improve the reliability and usefulness of both techniques.

2. What is software process simulation?

As a first step, we define the term 'simulation' as used in software process research. Since we believe that the issue we are raising is applicable to all types of software process simulation, we define this key term as broadly as possible.

A software process simulation is a simplified abstracted model, enactable on a computer, of a real or proposed software development or evolution process, usually producing results reflecting real situations or the expected results for proposed process changes. These results are often expressed in quantified terms. The model is most usually based on a graphical or textual structure intended to represent elements of the real or hypothesised process and the interactions between them.

3. Why simulate software processes?

Simulation of software processes is now a well-established and successful field of software engineering research and practice.

In the context of the development of simulation frameworks Pfahl [9] provides us with a number of likely reasons for this:

- "Focused improvement of techniques and tools, and associated conduct of controlled experiments and case studies, thus accelerating the generation of interesting new empirical evidence about the efficiency and effectiveness of development techniques and tools
- Standardized representation and packaging of empirical evidence about local effectiveness and efficiency of techniques and tools in varying contexts, facilitating the systematic exploration of the impact on (global) project performance at low cost.
- Improved knowledge transfer, education, and training through visualization of the impact of local effects on global performance."

We also note the successful use of software process simulation in making explicit theories of why software processes perform as they do (see, for example, [3]).

Simulation also allows the elimination of potentially confounding factors found in empirical studies, such as student performance and the particular circumstances of a real-world situation. This can particularly be the case when a new or modified process is being used for the first time and not all of the specifics of a situation which may influence the outcome can be identified and factored out in an experimental protocol.

Second International Workshop on Realising Evidence-Based Software Engineering (REBSE'07)
0-7695-2962-3/07 $20.00 © 2007 IEEE

Finally, simulation enables the prediction of the long-term effects over many years and many software releases of a process change, a prediction which would otherwise be unavailable when the decision to adopt or reject the change has to be made.

As Setamanit *et al.* [11] note, "In software engineering it is easy to propose hypotheses; however, it is very difficult to test them … Controlled experiments are costly and time consuming … and are nearly impossible to conduct. In addition, the isolation of the effect and the evaluation of the impact of any given factor within a large, complex, and dynamic project environment … can be remarkably difficult."

However, we suggest that a purely simulation-based approach to experimentation in software engineering has weakness which, we believe, can be addressed by the use of simulation and empirical techniques in combination.

4. What can we legitimately claim to be able to do with simulation models?

Setamanit *et al*. [11] claim the following status for their work. "With available empirical data software process simulation models can be constructed and calibrated so that they reflect real world behavior quite accurately. Such models can then be used as an experimental platform to investigate the situation/system and evaluate new hypotheses and theories. By varying individual parameters or combinations thereof, the magnitude and strength of the impact on variables of interest can be measured … Simulation models enable controlled experimentation that allows the researcher to identify factors that profoundly impact the outcome. It is far less costly and less time-consuming to perform experimentation using simulation models."

We are concerned at the thinking shown by this second quotation.[1] We agree that simulation models can be constructed whose quantitative outputs reflect observed behaviours, and that the model should be tested in sensitivity analyses to ensure that its outputs do change in expected ways when the values of its input are changed. In addition, it is certainly cheaper and less risky to build and run a simulation of a software project than to test a hypothesised process 'improvement' by using a real-world project.

However, we feel that the advantages of simulation can be overstated by suggesting that a researcher can use such a model to identify unambiguously "factors that profoundly impact the outcome" of a software development project" without noting strong reservations which we outline below. We believe that this also raises a deeper question about the nature of simulation, which is: *are speculative simulations actually* ***experiments*** *in the sense of the term as commonly used?*

If these investigations are not 'experiments' in the sense in which the word is usually employed and might reasonably be understood by the expert general reader, are modellers in danger of asking for too much credence to be placed in the outputs of simulation runs unless they support their conclusions with more traditional experiments? In other words, might it be useful to use empirical experiments designed specifically to address this weakness in a purely simulation-based approach by testing the hypotheses developed in modelling in an environment closer to the real world?

5. Why should the result of 'experiments' using simulation models be treated with caution?

What aspects, currently unavoidable in the way simulation models are developed and used in software engineering, might place limitations on the use and usefulness of simulation-based 'experiments', and therefore cause us to look more critically at what has actually produced the given output? For instance, what might caused us to have less faith in such an output than the result of a scientific experiment conducted with a tool which has a long history of empirical support, such as an optical astronomical telescope?

An unavoidable danger arises from giving as much credence to these results as to those of an 'experiment'. This is the risk that the assumptions we make in excluding things (as a result of our need to abstract and simplify in building a simulation model) might not hold up under changed conditions of a 'what-if' 'experiment', such as that which we have ourselves previously presented on pair programming [13]. This might be, for example, due to a failure to realise that a specific assumption or simplification, allowable under existing real-world conditions, may no longer apply under the conditions necessary for the speculative simulation run, *i.e.* when an assumption underlying the model structure or its explicit or implicit abstractions becomes vital to the success of the run, or, as Dewar [4] terms it, 'load-bearing'.

For similar reasons, the reassurance gained from testing the model by checking its calibration against real world reference modes and the justifications for the values used for simulation inputs may also be endangered by taking the model outside existing known 'safe' limits. The commonly used expert

[1] We stress that we have used this paper only as an example of a mind-set which we see as common in the software process simulation community, and intend no criticism of Setamanit and her colleagues, or of the research they do.

estimates for parameter quantification may lose their justification when conducting speculative simulation runs, particularly if modellers do not warn the experts that the latter's estimates are to be employed in this extra-normal fashion and give them an option to revise their opinions. This situation inevitably arises when these expert values are taken from the literature without referring back to the experts who provided them in the first place.

As a further example, the use of values obtained from student experiments, often on comparatively small-scale tasks by comparatively inexperienced developers, as input parameters for models purporting to represent the work of experienced developers in the real world, may work when a model is initially calibrated but may fail in unexpected and unpredictable fashions when the model is taken outside known calibrations into the realm of prediction.

However, expert estimates and results of small-scale academic experiments are sometimes the only source of values for simulation modellers to use to calibrate their models.

6. The Status of Experiments using Simulation Models

The question of the status of simulation-based 'experiments' has led us to reflect on the status of the simulation models themselves, which needs to be examined before the speculative simulation runs can themselves be considered.

What is the ontological status of the models we produce? To what extent can we justify the idea that the models represent the real world in some way and that they form a sufficiently stable basis and a sufficiently complete abstraction for the results of simulation-based investigations such as ours into pair programming [13] to be applicable in the real word? Do we need to make explicit the assumptions underlying these investigations?

Montgomery defines an *'experiment'* as "… a test or series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output responses." [7: p.1]. The repeated execution of simulation models with differing values of input parameters certainly meets this definition. However, we feel that Naylor *et al.*'s concern that "… computer simulation experiments are in effect experiments with a mathematical model …" [8: p.3] rather than with the real-world situation being simulated is still a valid issue despite the length of time since this statement was made.

Further, we suggest that non-specialists, including many likely users of the results of our simulation work, when faced with the term 'experiment' are liable to think in terms of hard science such as physics or chemistry. Are, then, we simulation modellers doing what we would identify with what people generally call 'experiments' when they refer to 'science'? We suggest *No,* at least not in the physics sense. If we are not conducting 'experiments', what claims can we make for our results, and might we be giving the wrong impression to outsiders by using the term 'experiment' with its connotations of scientific rigour and certainty of outcome?

Certainly, the performing of such an investigation using a simulation model does not seem at first sight to have the fundamental stability of a physical experiment intended to support the level of certainty of outcome as, say, mixing an acid whose properties are well-known with an alkali with equally well-known properties in a school laboratory, in repetition of an experiment conducted in chemistry classes many times before, most of those cases after the original, research-driven 'experiment' was conducted. In the case of the chemistry experiment, the repetition, and the support given by the underlying theoretical and procedural base of chemistry (Kuhn's [6] 'disciplinary matrix') lend an air of close-to-certainty in advance of our 'experiment'.

Even when such an experiment is conducted in a laboratory for the first time, without the certainly derived from a history of many previous repetitions, there is support for the reason for conducting the experiment (the theory to be tested), its procedures and protocols, and how to interpret its results, all of which can also be drawn from the discipline of chemistry [6].

Do we simulation modellers intend to claim that we can rely on any equivalent sources of comfort for repetitive or new results when we run our simulation to see what will happen when, say, we graft some observed values obtained from experiments with student subjects into pair programming on a simulation model of an existing software process [13]? Whilst the process and the tools can be selected based on a past of successful applications, the problem domain, that of software process, is greatly lacking in the theoretical base which might allow outcomes to be predicted in novel situations.

The underlying nature of simulation 'experiments', and thus the status of claims made on this basis, is a question requiring deeper consideration than we are able to give it in this position paper, so at this stage we are raising the question as one which software process simulation modellers need to take into account in the future.

For the present, we suggest that this weakness in a purely simulation-based approach can be to some extent mitigated by support provided by the use in

combination with it of series of practical experiments, each of which is designed explicitly to address one or more questions raised by the simulation.

7. What might we do to minimise the problems with simulations?

In addition to using experiments, how else might we improve the dependability of our simulation results? We suggest that the following guidelines would help improve the validity of simulation models.

- Use well-defined processes, such as that due to Ahmed *et al.* [1], to develop and check simulations.
- Use well-defined graphical representations, such that modelled variables and influences reflect actual (logical or physical) entities and actually-observed influences. In some cases this may require reliance to be placed on expert opinion in order to capture subtle influences and effects, but when this is done it should be stated explicitly when reporting the work. Further assurance should be gained by having problem domain experts agree that the model structure is reasonable before quantifying its variables
- Use well-defined parameters, i.e.
  - numbers that make sense in the real world [5: p.6];
  - numbers that make sense in the light of expert opinion of our theories or our own experience of research in the area, even if they are difficult to measure in the real world (such as the effect of the existing system on our ability to change it [13] (cf. [5: p.6]). However the need to be able to quantify parameterised inputs to models must also be taken into account, and if this results in values having to be estimated by expert estimates or indeed by curve fitting to see whether a model can reproduce the shape of a trend this must be acknowledged as an issue to be resolved (cf. [3]). However, we note the current state of uncertainty concerning the validity of many metrics in software engineering, and the need in gathering values to calibrate simulation models to quantify aspects of the process or product whose quantification is itself difficult, problematic or controversial
  - When extracting values for simulation model parameters, take care, but only as much care as the model requires; do not introduce over-'accurate' parameter values
- Place explicit limits on claims made on the basis of speculative simulation runs.
  - Simulation modellers should no longer refer to these runs as 'experiments'; as we have noted they are not experiments in the sense of the term in generally-accepted use. They are investigations, but not experiments in the physical science term of the word
  - If model output values are to be cited, limit the number of significant figures in the outputs to the lower of whatever the model's underlying accuracy, and that of its inputs, will bear.
- Accept that there are severe limitations to the validity of results of purely simulation-based work, but treat this fact as an issue to be addressed rather than as a reason for dismissing them.
- Attempt to quantify the risks, errors or uncertainties in the outputs of the model. This is often achieved in current practice by sensitivity analyses which show how the output of interest changes as each input parameter is varied. This gives an idea of the extent to which errors in the numeric values of these parameters might affect the quantified outputs, but not the degree to which the modeller's confidence in their model structure has been affected when amending that structure to reflect the conditions under investigation. The latter may well turn out to be a much more difficult task than sensitivity analyses with respect to input parameters.
- Check *post facto* to see how well the models predictions are reflected in more realistic situations than the mathematical world of a simulation environment. Here, the models and their predictions are to be seen as theories in themselves, and subjected to more realistic tests. These tests consist of the design of experiments intended to refute [10: p.276] the predictions of the simulation models, conducting the experiments under controlled conditions, comparing the results of model predictions and experimental results, and a search for reasons for the inevitable differences found. The final test of predictions, to operationalise them in real-world processes, would follow only after this trial.

The use of simulation and experimentation in combination is considered in the next section.

8. Combining empirical and simulation approaches

How might we exploit the strengths of simulation and empirical approaches to maximise the benefits to be obtained from both? We propose, as Pfahl [9] hints when he notes the need for "associated conduct of controlled experiments and case studies" that the

changes made to simulation models which can already reproduce real-world results when 'what-if' simulations are conduced be considered less as experiments and more as theories to be tested, with the quantitative and qualitative outputs from them seen not as results to be applied directly but as speculative results which need support from closer-to-real-world experimentation.

We therefore suggest that empirical experiments should be designed on the basis of speculative simulation conditions, and conducted under conditions otherwise as close to the real world as possible, in order to test the simulation predictions in a practical environment. In addition to testing simulation results in a more rigorous fashion than can be achieved in the simulated world, this approach would provide a strong theoretical basis for the protocols used in the experiments, showing explicitly why *these* experiments need to be conducted, specifying both their various inputs and the expected outputs and how to measure them. It might also assist in the identification of confounding factors in the empirical experiments, and perhaps in the quantification of their effects.

The process of simulation and experimentation is likely to need to be iterated, as experimental results are fed back into simulation runs in the form of modified model structures and changed parameter values more directly focussed on the questions which the simulation had raised. Subsequent simulation runs may themselves require changes to the experimental design and protocols or the reinterpretation of existing experimental results.

This iterative process might be viewed as an example of the Shewhart or Deming plan/do/check/act cycle, with the simulation model providing the plan for a specific experiment (the 'do'), then a checking of the experimental results followed by action both to improve the simulation model and, if justified, to improve the real-world situation being studied. However, in order to maximise the benefit from this approach we strongly believe that appropriate weights be given to both simulation and experimental aspects, and not to weigh one unduly at the expense of the other. This may require simulation and empirical researchers to combine their expertise and submerge their egos in a combined investigation, in the same way that programmers are required to submerge their egos in eXtreme Programming [2: p.59]. It will also demand the acceptance by both parties of shared definitions for key technical terms in the joint work. Perhaps the most crucial of these terms where a shared definition may not have yet been agreed is the word 'experiment'. If this is not achieved, then simulation and empirical workers will inevitably fail to communicate clearly as they work at cross purposes.

We show our approach in diagrammatic form in Figure 1.

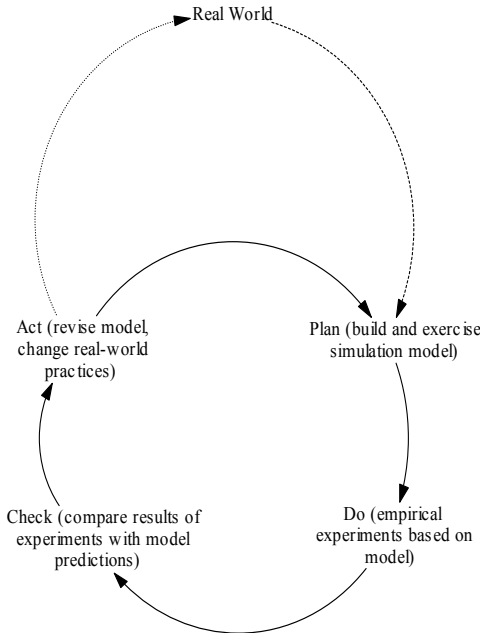

Figure 1: simulation and empirical software engineering research combined as Shewhart/Deming cycle

Our proposed combined approach would also make it easier to explain to non-specialists why specific experiments need to be conducted. This support may provide stronger justification to sceptical senior managers for the high cost of these experiments. It may therefore also help address another weakness common to much current experiential work in software engineering, which is that of the potentially unrealistic results obtained from of the common use of comparatively inexperienced students as subjects in environments intended to reflect real-world conditions. The ability to use simulation as the basis of experimental design, and the ability to show potential results in the form of simulation outputs, may help sell to managers the idea of using in experiments scarce and expensive resources in the form of their most experienced developers, rather than relying on results obtained by using students. Such a change in approach must inevitably improve the reliability, applicability and acceptability of experimental results.

9. Conclusions

In response to the problems we have identified with current simulation reporting practice, we could simply say, 'Adopt the practices we have outlined above and

it'll all be OK!' However, more seriously, we urge all simulation practitioners, when reporting their work either publicly or in private, to emphasise these uncertainties in speculative simulations, and refrain from using words such as 'experiment' which might influence people unaware of the issues which we have outlined here to have more confidence in the results of the simulations which we are crafting than our models can bear.

Finally, we still firmly believe that simulation has an important role in software engineering research and practice. It enables proposed processes changes to be examined at far lower cost than any real-world intervention. When strengthened as we suggest by combining it with focussed empirical work, simulation remains far cheaper, quicker, and more generalisable than the current practice of immediately 'testing' in real projects new ideas by adopting, for example, a proposed process change on the basis of anecdotal evidence, blind faith and an appeal to current fashion [12: p.157] which are sometimes all we appear to be able to offer to industry!

References

[1] R. Ahmed, T. Hall, P. Wernick and S. Robinson, "Evaluating a Rapid Simulation Modelling Process (RSMP) through Controlled Experiments", IEEE International Empirical Software Engineering Conference, Noosa Heads, Australia, Nov 2005.

[2] K. Beck, *eXtreme Programming Explained*, Addison-Wesley, Indianapolis IN, 2000.

[3] B.W. Chatters, M.M. Lehman, J.F. Ramil and P. Wernick, "Modelling A Software Evolution Process", *Software Process: Improvement and Practice*, **5**, 2–3, pp.91–102, 2000.

[4] J.A. Dewar, *Assumption-Based Planning: A Tool for Reducing Avoidable Surprises*, Cambridge University Press, Cambridge UK, 2002.

[5] A. Jedlitschka and D. Pfahl, *Reporting Guidelines for Controlled Experiments in Software Engineering*, report ISERN-REPORT ISERN-05-01, Fraunhofer Institute for Experimental Software Engineering, 2005.

[6] T.S. Kuhn, *The Structure of Scientific Revolutions*, Second Edition, University of Chicago Press, Chicago IL, 1970.

[7] D.C. Montgomery, *Design and Analysis of Experiments*, Wiley, 2005.

[8] T.H. Naylor, D.S. Burdick and W.E. Sasser, "The Design of Computer Simulation Experiments", *in* T.H. Naylor (ed.), *The Design of Computer Simulation Experiments*, Duke University Press, Durham NC, 1969, p.3–35.

[9] D. Pfahl, "Software Process Simulation Frameworks in Support of Packaging and Transferring Empirical Evidence," position statement, *Empirical Software Engineering* (Nº 06262), Dagstuhl Castle, Germany, 26-30 Jun. 2006.

[10] K. Popper, "The Problem of Demarcation", *in* N. Warburton (ed.), *Philosophy: Basic Readings*, Routledge, London, 1999, p.275–286.

[11] S. Setamanit, W. Wakeland and D. Raffo, "Exploring the Impact of Task Allocation Strategies for Global Software Development Using Simulation", *in* Q. Wang et al. (eds.), *Proc SPW/ProSim 2006*, LNCS 3966, Springer-Verlag, 2006, pp.274–285.

[12] P. Wernick, *A Belief System Model for Software Development: a framework by analogy*, PhD thesis, Computer Science, University College London, 1996.

[13] P. Wernick and T. Hall, "The Impact of Using Pair Programming on System Evolution: a Simulation-based Study"; *Proc. IEEE International Conference on Software Maintenance* (ICSM) 2004, IEEE.