Empirical Studies Applied to Software Process Models

DAVID RAFFO
School of Business Administration, Portland State University, USA

TIMO KALTIO
Nokia Research Center, NOKIA Group, Finland

DEREK PARTRIDGE
Department of Computer Science, University of Exeter, UK

KEITH PHALP
School of Design, Engineering and Computing, Bournemouth University, UK

JUAN F. RAMIL
Department of Computing, Imperial College, UK

Introduction

Organisations involved in developing, maintaining and enhancing software experiment constant changes in all the relevant domains and face increasing demands for improvements in cycle time, cost-effectiveness and product quality. Software process modelling (Curtis et al., 1992) is emerging as an effective tool for *inter alia* evaluating changes made to a software project or development organization and to help manage these changes. But its implications extend beyond the software process. Business process models, for example, are also used to represent end-user business processes to support requirements elicitation and analysis of activities. Research programmes such as Systems Engineering for Business Process Change (SEBPC, 1998) highlight the importance of modelling business processes in evolving and maintaining the software systems that support those processes. Hence, process modelling and simulation are becoming widely used within software engineering, for a variety of purposes and audiences, and using a variety of notations and tools. However, the relationship between empirical studies and software process modelling and simulation is relatively unexplored, as exemplified by the scarcity of empirical studies relating to the practical impact of process modelling and simulation. This paper addresses some relevant aspects of the multifaceted relationship between empirical studies and the building, deployment and usage of software process models. The paper draws on a variety of experiences and perspectives of process modelling and suggests techniques and questions for further investigation.

Simulation Model Uses and Types

Models and simulation are used as an aid to decision making, to aid in risk reduction, and to help management at the strategic, tactical, and operational levels. The many uses of software process simulation models have been clustered into six categories (Kellner et al., 1999):

- strategic management
- planning
- control and operational management
- process improvement and technology adoption
- understanding
- training and learning

Software process simulation models focus on some particular software development, maintenance, or evolution process. They can represent such a process as currently implemented (as-is), or as planned for future implementation (to-be). Since all models are abstractions, a model represents only some of the many aspects of a software process that potentially could be modeled—namely the ones believed by the model developers to be especially relevant to the issues and questions the model is used to address. When developing software process simulation models, identifying the purpose and the questions and/or issues management would like to address is central to defining the model scope and data that need to be collected. Common purposes of simulation models are to provide a basis for experimentation, predict behavior, answer what-if questions, teach about the system being modeled, and so forth. Such models are usually quantitative, although this is not always the case (e.g. knowledge based systems simulation).

When Simulation Models are Necessary

Simulation models (or any other modeling method) are an inexpensive way to gain important insights when the costs, risks, or logistics of manipulating the real system of interest are prohibitive (Abdel-Hamid and Madnick, 1991). Simulation models are generally employed when the complexity of the system being modeled is beyond what static models or other techniques can usefully represent. Complexity is often encountered in real systems and can take for example the following forms:

System uncertainty and stochasticity—Simulation models are needed when the risk or uncertainty of the system is an essential feature to consider. Simulation models provide a flexible and useful mechanism for capturing uncertainty related to complex systems without the restrictive constraints required by other analytical models (Raffo, 1996).

Dynamic behavior—A dynamic model is appropriate when system behavior changes over time and when accounting for (or controlling) key variables representing these changes is important. Dynamic simulation models are very flexible and support modeling of a wide variety of system structures and dynamic interactions (Madachy, 1994).

Feedback mechanisms—Feedback occurs in the software process at many levels and in a variety of ways. In feedback systems, behavior and decisions made in the process generally have counterintuitive impact. Thus feedback must be taken into consideration to achieve a degree of mastery on process changes and manipulations (Lehman, 1994; FEAST, 1999). For complex feedback systems and their models, analytic approaches may not be useful or even possible. Simulation offers a feasible alternative.

Types of Simulation Models

In general, three simulation paradigms that have been applied to software development processes are (Curtis et al., 1992): discrete event, state-based, and continuous (i.e. System Dynamics).

The term "discrete models" is used to refer to a group of models that advance time as a result of a discrete event. It is assumed that nothing of importance to the model happens between event times, therefore, the model can advance the simulation clock from event to event with no loss of information. This type of model is efficient and particularly appealing when the process is viewed as a sequence of activities. Discrete models are often used to model manufacturing processes where items or entities move from station to station and have processing done at each station. Discrete models easily represent queues and can delay processing at an activity if resources are not available. In addition, each entity may be described by unique attributes. Changes to the attributes by the activities can provide much of the value of a discrete model. The duration of each activity may be sampled from random distributions allowing the model to represent process uncertainty. This allows the simulation to capture the effects of variation in the entities on each activity.

In software development, there has been a long history of viewing the development process as a sequence of discrete activities. To enable more precise management, process models like the Waterfall Model (Royce, 1970) or the Spiral Model (Boehm, 1988) have been proposed as a generic set of activities that can be used to organise and track the progress of the development effort. Tausworthe's Work Breakdown Structure (Tausworthe, 1980) is a clear example of a systematic description of the development process as a sequence of discrete activities. More recently, the Capability Maturity Model (Paulk et al., 1993) stresses the importance of a description of the process as a detailed sequence of repeatable activities.

Because a discrete model allows each entity to contain unique values for attributes, the model can capture the variation in code difficulty and programmer capability. If we wish to model the effects of increased complexity on effort and error rates, or if we want to model the impact of different programmer capabilities on productivity, a discrete model allows us to represent these individual differences as attributes.

Finally, a discrete model allows us to represent the interdependence that occurs between activities in a project. Activities in a development process may be delayed when a programmer is diverted to another task. If a model can capture these dependencies at a sufficiently detailed level, it may show ways to alter the process to reduce risk or increase efficiency.

The distinct advantage of state-based simulation offers a rich representation of the events that drive the software process to progress. This capability is very significant for promoting process understanding and being able to communicate salient features of process. State-based models have a strong process focus where attributes are attached to process steps in the model rather than to artifacts as artifacts and resources are handled at a more abstract level than is typically done in discrete event simulation. At the same time, however, most applications using the state-based paradigm to date have been modelled using a discrete clock (see Kellner et al., 1999, for an overview of recent work in the field).

Continuous simulation models (often referred to as System Dynamics) use differential equations to describe the state of the system, the events that occur, and the artifacts that are generated. Time advances in small increments. Work is accomplished, artifacts are transformed, and resources are consumed during each small increment of time. The key feature of continuous simulation is that the differential equations used to describe the system (as well as the forces and feedback loops prevailing in it) are continuously updated. This enables a variety of factors to be represented using continuous simulation that are difficult to capture using other paradigms. Examples include: worker fatigue, the effect of schedule pressure and so forth (Abdel-Hamid and Madnick, 1991; Madachy, 1994).

For an overview of the types of simulation modeling techniques that have been applied to software development processes as well as current research being conducted in this area, interested readers are referred to the proceedings of the International Workshop on Software Process Simulation Modeling (ProSim'98 and ProSim'99) as well as a recent special issue of the Journal of Systems and Software (*JSS* 46(2/3)).

What Are the Empirical Issues Related to Software Process Simulation

In this section, we identify important empirical issues encountered in the analysis of

- Process data used as direct input to simulation models and/or used to assist model building
- Model output data used to support management decisions about process alternatives
- Model structure in the context of evaluating the efficiency of a process
- The effectiveness of process models in supporting process change and improvement

This section also reports on the status of current work being done to address these and identifies relevant statistical and analytical techniques.

Empirical Analysis of Inputs

In organisations subject to many technical and economic pressures the need for process improvement is clear. Simulation models can play an important role achieving such a goal. Its usefulness, however, could be limited to the degree to which the simulation models relate to the real world process being modelled. The use of real world (empirical) data as model inputs appears to be, therefore, an important component in increasing model validity. The importance of the empirical data becomes evident when one considers that this data, sometimes, is the only available directly traceable link between the simulation model and the reality being modelled (Ramil and Lehman, 1999a).

The word input in this section and in process simulation is used in a wide sense. Empirical input data can serve many purposes within the model building and usage process, for example, to provide

- Realistic inputs to the model, so that the model can simulate the response of the process to realistic conditions
- Real world process behaviour which can be compared with model outputs to assess, for example, model prediction capabilities
- Means for calibration of model parameters to real world conditions
- A source of reference modes (Roberts et al., 1983) for model builders to identify modes of behaviour that may suggest them the presence of relevant behaviours and/or suitability of particular model structures.

Empirical input data for software process simulation models may originate from many sources, historical and present, for example

- Past executions of the process to be simulated
- Data from other processes within the same organisation
- Industrial benchmarking data
- Human expertise

The data may be qualitative or quantitative, though in most cases quantitative data seems to be the predominant type. This section will focus on quantitative data. Some issues raised in connection with the use of qualitative data in the context of software process simulation have been discussed elsewhere (Levary, 1990; Ramil and Lehman, 1998).

The richness and complexity of the software process and its products is reflected in the numerous types of data that can be collected from it and in the many formats data can take. Methods have been proposed for process instrumentation and for the systematic collection of data and metrics, such as the GQM (goals-question-metrics) paradigm (Basili and Weiss, 1984). Well understood principles are provided for those starting a measurement programme from scratch. Those collecting data with process simulation purposes in mind can apply already established data collection principles (Fenton and Pfleeger, 1997). Simulation models can also be used as vehicles to suggest new or alternative metrics and indicators. That is, the model building process can serve as driver of the process instrumentation, indicating critical process areas that may require monitoring and helping in the definition of process metrics.

For those who undertake process simulation without existing data collection programmes in place, options are still open, since useful data sources may exist. Many of the software technology tools incorporate databases that can turn to be useful sources of data. For example, data from a configuration management system offered a suitable source, which together with input from a process expert, enabled the calibration of a dynamic model in the FEAST/1 project (Chatters et al., 1999; FEAST, 1999). General issues and guidelines concerning the utilisation of existing data in empirical studies of the software process have been discussed in Cook et al. (1998). These authors recognise, both the cost-effectiveness of the use of existing empirical data but also its limitations. They mention that

- available data sources may not be an accurate surrogate for the attribute of interest

- historical data can only show the presence of correlations, being not possible to confirm causal relationships by manipulation of the independent variable

An extreme example of the first case occurs when the process attributes represented in the simulation model are not reflected by the available data. It has been suggested that the definition of *primitive* or *primary* indicators could help in solving this issue, but that remains to be investigated (Ramil and Lehman, 1999b). While this remains to be seen, lack of appropriate metric sources inevitably will force model builders to rely on external data sources and/or in human experts. It is mentioned in passing that simulation models can be built to generate inputs, such as when a simulation model is built to predict maintenance requests originated by a given group or type of users and usage (application domain). Whether the inputs generated therefrom can be considered valid or whether the data generation model can be considered part of a larger simulation model is not clear. It is reasonable to assume, however, that in some situations model builders may need to include as part of the simulation and/or related empirical input collection, processes beyond the immediate technical software process. This has been the case, for example, in FEAST/1, and its continuation, FEAST/2, in which the *global* software process, that is, the process involving the activity of developers, users, managers, support personnel, marketeers and others, is being studied (FEAST, 1999).

The extraction of data and metrics from unstructured data sources, including documents in natural language, represents another challenge. Progress has been being made in this field, for example, in the analysis and metric extraction from web-based documents. These results seem to be applicable, at least in part, in the extraction of data with process simulation purposes.

One of the conditions that is required for the effective use of existing data in simulation modelling is the absence of conflict between the assumptions underlying the model, on one hand, and those underlying the data, on the other. One example of such a conflict would be the case of a model involving conventional code generation, for example , code written by people using text editors. Conflict will necessarily occur if, to study this process, data from another process in which a significant portion of the code is automatically generated by tools, such as GUI builders, is used as empirical input. An antidote to this is the collection, together with the empirical data, of background and context information (meta-data) which will prove useful, in particular, in the later phases of model output interpretation.

As large as the variety of purposes that empirical inputs can serve in the context of process simulation model building is the variety of applicable data analysis techniques. These range from initial exploratory analysis and plotting (Tukey, 1977), filtering and smoothing (Tesoreiro and Zelkowitz, 1998) to regression modelling (Birkes and Dodge, 1993).

Inductive Learning and Process Modelling

In recent years the machine learning (Mitchel, 1997) subfield of inductive learning (Partridge, 1997), that is, automatic extraction of general information from a collection of specific instances, has spawned robust technologies. These appear to be appropriate for widespread use in software engineering, generally, and in the application of empirical data to process modelling.

With respect to the development of process models, these inductive techniques can be used to extract information from empirical data that derives from software processes being modelled. An example of this is the analysis of input and output data which provides a significance ranking of the selected input parameters with respect to their importance in generating the chosen outputs. A number of such techniques have been proposed and evaluated (Wang, Jones and Partridge, 1998, 1999).

Process modellers, while exploring and assessing the potential parameters for model building, can use such significance information. It can be determined whether any parameters appear to be information-less, in the context of the particular modelling task being undertaken and which parameters are highly relevant.

In addition, it is possible to use the products of inductive data mining such as trained neural networks as modules or components within the process simulation model. In this complex modelling domain it may be possible to collect (or construct) a set of instances of the behaviour of a model subfunction, and yet be well short of a functional specification. In such cases, inductive programming can be used to operate on the set of instances to generate an implementation module which can then be used directly to compute this subfunction within the process model. If decision-tree induction technology is used, then the decision tree generated is open to a rational analysis which might then shed light on the development of a functional specification for the module, so opening a route to conventional model building production via an inductive prototype.

Automatic Model Generation

Generation of dynamic models directly from empirical data are being investigated in control engineering, for example (Bradley and Easley, 1997). These techniques and the principles underlying them still await exploration in the context of software engineering and the software process. Though provoking, whether it is feasible or not remains to be seen. The overcoming of some of the challenges in software process data collection (Ramil and Lehman, 1999b) may also open up opportunities in this area.

Analysis of Simulation Outputs

When stochastic modeling and Monte Carlo simulations have been employed, simulation outputs of key performance measures of interest will contain a great deal of variability due to the variability of the inputs. Accordingly, the distribution of the output variables must be determined as a first step in analysis. If the output variables are normally distributed,

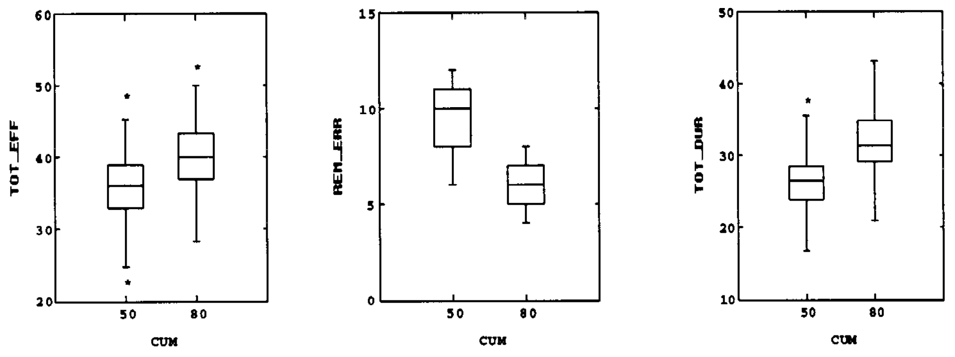

Figure 1. Comparison of the no inspection and inspection baseline cases. TOT_EFF = TOTAL STAFF EFFORT (IN PERSON-DAYS); REM_ERR = NUMBER OF REMAINING ERRORS; TOT_DUR = TOTAL PROJECT DURATION (IN DAYS); Legend: 50 = no inspection baseline, 80 = with inspection baseline.

parametric statistical tests can be used to analyze the results. Otherwise non-parametric tests may be required. The CHI Square tests (Kolomogorov–Smirnov) and probability plots are good tests for normality, as discussed and applied in (Raffo, 1996).

Also, the correlation among process performance measures should be tested in order that no surprises are encountered. Unlike in traditional regression analysis, it can be all right for certain output measures to be correlated. For example, effort and schedule are usually correlated. In fact, certain models may use effort to predict schedule. Having these output variables being correlated in a model is quite acceptable. Of course, both measures are of interest to managers.

Although the above analyses are used for understanding the properties of the model results from a statistical perspective, other analyses are used to manipulate the data to support decision making objectives of the model. For instance, it is generally quite valuable for managers to conduct extensive sensitivity analyses of meaningful business scenarios in order to gain insight into relevant risks. Typically, when conducting these analyses, the variability of the output measures will not be reduced in a statistical sense. However, the sensitivity analyses will provide insight into the potential benefit or costs associated with the various scenarios and thereby provide an assessment of the risk. As another example, in evaluating the impact of process changes, we have used the following four approaches to evaluate alternatives:

Simple Comparison of Performance Measure Deltas (Pareto Analysis)

When running a simulation model, the results of the process with the proposed process change and without the proposed process change are obtained and compared. We have found "box plots" to be a valuable technique for visually comparing distributions of key output variables, as illustrated in Figure 1. The differences between performance measure distributions are obtained and checked for statistical significance using t-tests. If all the

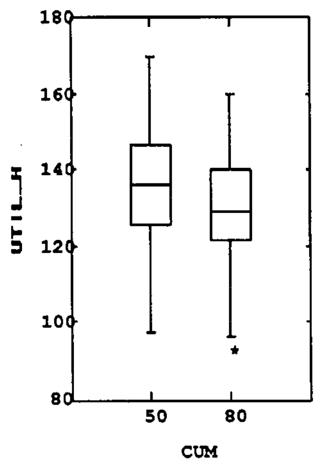

Figure 2. Utility function values for the ISPW-6 software process example UTIL_H = utility function values; 50 = "AS-IS" no-inspection baseline; 80 = "TO-BE" inspection baseline.

performance measures of interest (e.g., life cycle cost, product quality, project schedule, and so forth) are improved and the differences are statistically significant, then we have confidence that the proposed process change will offer an improvement. Figure 1 shows the results from a simulation model of the ISPW-6 software process example. The box plots show the difference in performance between the AS-IS process without inspections and the TO-BE process with inspections. Figure 1 shows that the TO-BE process with inspections performs better by having fewer remaining defects (REM_ERR). However, more staff effort and a longer duration are required.^{1} As a result, a tradeoff among the performance measures will be needed. A base case analysis as well as various sensitivity analyses were also conducted. A discussion of these analyses is beyond the scope of this paper, but see (Raffo, 1996), (Raffo and Kellner, 1999) for details. (In this model and in the figures, lower values indicate less total effort, fewer remaining defects, or shorter time to market.)

Comparison of Performance Measure Deltas Using a Utility Function

As described above, the simulation model is run and the results for the AS-IS and TO-BE processes are obtained (figure 1). The differences between their performance measures are then determined. Since these differences show conflicting results for a given process configuration (some performance measures worsen and others improve), a tradeoff by the decision-maker must be made. In this case, developing a utility function reflecting management's preferences can be a helpful way to assess the tradeoff and to reduce multiple performance measures down to one number. This overall measure can be checked for statistical significance and a decision can be taken. Figure 2, shows a box plot diagram of the utility function values developed to evaluate the results from the ISPW-6 model. The results show that given management preferences, the TO-BE with inspections was preferred in this case. The reader is referred to (Raffo, 1996; Raffo and Kellner, 1999) for details and examples. (In this model as shown in the figure, a lower utility function value is preferred.)

Comparison of Performance Measure Deltas Using Financial Measures Such as Net Present Value or Return on Investment

This approach is very similar to the approach using a utility function. Net present value or other financial measures are calculated using a certain kind of utility function where all performance measures are reduced to cash equivalents. The most difficult performance measure to reduce to a cash equivalent typically is schedule, because good estimates describing the dollar value associated with releasing the product earlier or later are difficult to obtain. Further details are provided in (Raffo, 1996). Also see (Harrison et al., 1999a, 1999b), (Raffo et al., 1999).

Comparison of Overall Performance Measure Values Using Data Envelopment Analysis (DEA) (Charnes et al., 1978, 1994)

In this approach, the performance measure results of a number of process configurations are obtained. They are then compared using a DEA objective function. This function may contain defects, effort, and schedule. DEA may be viewed as an optimization technique which finds a "productive frontier," i.e., a select set of process configurations that are potentially the most efficient given the input set. DEA works like this: For any given configuration, the DEA program determines a set of objective function weights that is most favorable to the given configuration. Suppose configuration "A" has one of the best (shortest) schedules, but more defects and higher costs than most of the other configurations. So, configuration "A" would choose the schedule parameter to be valued most in the objective function. The DEA program sets the weights for this objective function and then evaluates all other configurations according to this "schedule heavy" objective function. If configuration "A" is the best using this objective function, it is held as a candidate for the optimal set. If another configuration beats configuration "A" using configuration "A"'s objective function, the DEA program knows that configuration "A" is sub-optimal and discards it. The DEA program evaluates all the configurations in this manner and determines a productive frontier from which final selections can be made.

Empirical Evaluation of Process Models

Many authors have argued that process modelling and measurement should be complementary disciplines. Typically, it is argued that models can act as a framework for measurement, or that measures can add quantitative evidence to models. For example, many studies of the software process utilize process or product measures to argue that improvements have been made. However, a further complementary use of these technologies is to use measurement to help in the analysis of the process models themselves. Of course, one might argue that this quantitative dimension is already present in many modelling tools, for example, in constructing a system dynamics model one requires some form of quantitative input data. However, despite the proliferation of such ‘advanced’ modelling strategies, there are many situations where more conventional, purely diagrammatic (or static) models are used to model processes. These models are often used where understanding the process, and gaining process buy-in are seen as more important than the production of rigorous models, as is often the case in modelling business processes.

Business process modelling is increasingly used in conjunction with traditional software development (Phalp, 1998). Models are used to understand the context for change and as an aid to requirements and analysis activities (Yourdon, 1994; Booch, et al., 1999). However, the major use of business process modelling is to attempt to restructure the business process in order to improve some given aspect, e.g., cost or time. This restructuring may be seen either as separate activity or as a precursor to the development of systems to support the new or improved process. Hence, the analysis of these business models is vital to the improvement of the process, and as a consequence to the development of supporting software systems.

Business processes are typically described using static (diagrammatic) models and their analysis is typically qualitative, relying upon the experience of the modeller and the application of guidelines or heuristics (Miers, 1997). Hence, a potential use of measures (counts) would be to aid the analysis and comparison of these static process descriptions. This idea was proposed at the 1997 ICSE workshop on process models and empirical studies of software engineering (Phalp and Counsell, 1997) where the authors showed how counts could be applied to Role Activity Diagrams (RADs), a widely used process modelling notation (Ould, 1995). The authors suggested that a simple coupling measure could be used to provide useful guidance to the modeller, but noted the need for further empirical work to investigate the utility of the idea. Recent study has applied this coupling measure, a simple ratio measure, to Role Activity Diagrams describing ten prototyping processes. It is not, however, the intention to suggest adoption of this single measure, but rather to show the utility of this approach in general. It is likely that other simple counts, such as the number of interactions per role, the size of roles (actions + interactions), would be used along with the coupling measure. However, results from the study are encouraging, in that it appears that role types appear to exhibit similar levels of coupling across the different processes and organisations. That is, coupling levels were consistent with role types being from the same population. Furthermore, where roles did not adhere to tentative threshold values for coupling deviations could be explained by particular circumstances. This suggests that the coupling metric may be useful in helping to identify spurious or ‘outlier’ roles. That is, roles that exhibit particularly high (or low) levels of coupling for their role type within an organisation or site. This complements the qualitative analysis of such models, by allowing the modeller to quickly spot potential problems, or areas where the process may need restructuring. Hence, it is suggested that there is merit in applying simple counts to complement traditional forms of business process analysis. Moreover, this use of measures to aid the analysis of the models themselves shows a further complementary use of process modelling and measurement technologies.

Empirical Evaluation of the Effectiveness of Process Models in Supporting Process Improvement

In addition to supporting evaluation of process improvement alternatives and other management planning activities, another common use of process models is to support human enactment of processes. The deployment of an existing process can be a far more arduous task than defining and piloting a new, improved process. Depending on the size and maturity of the company, processes that have been previously defined may take root in the culture quickly or slowly. Software processes are never ‘ready’ or ‘finished.’ An organisation has to maintain process models over the years (Kaltio and Kinnula, 1998).

This usability of process models (including models of the process architecture, notations, quality of on-line implementation, contents and so forth) has an important effect on the success of deployment. Moreover, the process deployment can fail specially if an organisation doesn’t put enough emphasis into the promotion of process models and the infrastructure needed for process deployment and *process asset* management. Early results from a multi-case study conducted in Nokia Mobile Phones (henceforth NMP) shows clearly the importance of stable and well implemented infrastructure for software process asset management and deployment.

In 1995 the NMP’s software process improvement program evaluated the status and use of company’s existing software process documentation. The documentation was found to be very difficult to access and use, outdated, lacking in coverage and having contradictory instructions. For these reasons, the guidelines were not used for the most part. In essence, the defined process was not deployed. A decision was made to improve the status of process deployment. A new software process definition and approach for supporting deployment was developed and carried out. These have been in use since September ’96 and have been proven to be very successful in most of the SW R&D sites (Kaltio and Kinnula, 1998).

The focus in process architecture and design definition has been usability in human enactment. It has a modular, highly standardised and homogenous structure. The process library has been implemented as a hypertext solution in Lotus Notes, with access through Intranet Web. The NMP has more than 10 SW R&D sites. All the sites use same software process asset library, but there are big differences in user activity figures between sites. The sites that have well implemented infrastructure have clearly better user activity figures. The data from almost three years period suggest that focusing solely on improving the usability and contents of process representations doesn’t guarantee success in process deployment (Kaltio and Kinnula, 1998; Kaltio, 1999). The standardised structure provides though, based on same data, obvious benefits for process maintenance in a decentralised organisation. The structure and contents are easier to keep consistent, process area responsibilities can be clearly shared, more stable WHAT to do information can be separated from method, language, tool dependent HOW to do information, and so forth (Kaltio, 1999).

Infrastructure for process engineering includes organisation, people, technology and knowledge. Organisation consists of structure, roles and responsibilities, and communication lines. How responsibilities of different process asset management and deployment activities should be shared between different roles, and organisational units and layers, is related to the overall software development and software process improvement organisation.

Such an organisation needs people with necessary skills, motivation and time allocated to training activities. Special emphasis should be put into the training and coaching of the new members of the software process asset management and deployment organization. The personnel turnover and growth without proper training can easily cause deterioration of the infrastructure. Technology includes at least the media (e.g. process asset library) for making process models accessible by the users, and a mechanism for management of process assets. Some kind of database solution is a must for management of the process assets. Otherwise, the continuous improvement of those assets, especially in a decentralized process engineering organization, becomes an impossible mission. The database shall provide basic configuration management functionality at a minimum. Knowledge refers to the undocumented and documented information assets, which are used to guide the enactment and improvement of the process (Kinnula, 1999; Kaltio, 1998, 1999). Organisations need to systematically measure the results, activities and infrastructure for software process asset management and deployment. The following are some examples of issues that measurements needs to cover:

- Are process representations used and followed? (results)
- Do users provide feedback? (results)
- Are process representations maintained? (activities)
- Are users properly trained? (activities)
- Are roles in the organisation filled? (infrastructure)
- Is enough time allocated to perform the activities? (infrastructure)
- Do people in the organisation have necessary skills? (infrastructure)

The measurements need to reflect activity and infrastructure models for process asset management and deployment of an organisation. The measurements are as important for improvement of above-mentioned activities as they are for the improvement of the software engineering process itself.

Research in the field of process modelling needs to put more focus on, how to get real benefit out of process models in industrial settings, i.e. process deployment and process asset management. Research done in the area of software process asset management has concentrated on developing and describing specific methods, notations, models and tools for defining processes, and process-centered software engineering environments. Less research has been done to identify what it takes to maintain, deploy and re-deploy those process assets. For example need to be able to maintain process representations over the years is major requirement which is not always taken into account when building prototypes of on-line process manuals. However, these issues are partially covered in many papers having a wider perspective to software process improvement.

Also we need more usability studies of process models in such an empirical environment where the models have already been in use for a long period of time. Short term pilot projects and controlled experiments don’t reveal all the long-term benefits and problems of the used process modelling notation and on-line or off-line implementation. For example:

- how can we improve the presentation of process information in order to increase the understandability and reduce the effort required to create and maintain it?
- how does the usability of process documentation evolve when the latter is used for a long period of time?
- what is the main audience(s) for the process representation at different process maturity levels?

These questions can be best answered by conducting long term studies in the industry. Companies should be willing to invest on improving the management of the technologies and knowledge that has been captured and turned into process representations.

Conclusions

This paper has brought together contributions from researchers and practitioners who are applying process and simulation models to help to improve the software process. Hence, a variety of techniques and experiences are described, both for the construction and analysis of models.

A recurrent theme has been the complementary nature of models and measures. In brief, the authors show how models provide a framework for metric collection, and how metrics support analysis of the model itself, and of the process being modelled. For example, software process simulation models can be effective in providing a framework and focus for metrics collection programs. The results and information provided by these models can be useful to managers and motivate them to put additional value on data collection efforts. Integrating process and product metrics with process models can be an effective means for providing valuable information to project managers—to support the important planning and control decisions they face. Empirical study of inputs and outputs of simulation models, the models themselves, and the real-world data they reflect, are therefore, of vital importance.

Furthermore, techniques to aid the analysis of such models will be important in the adoption of these technologies. Again the authors describe a variety of (mostly quantitative) techniques, ranging from the application of simple counts to static models to more complex analyses of process simulations. The paper has shown how a variety of statistical and analytical techniques could be employed to evaluate various process alternatives. We have also described how utility functions, financial measures (NPV) and data envelopment analysis can be used to evaluate simulation model outputs. These techniques enable management to make better decisions about their processes and to better understand the impacts of these decisions on their projects.

In all cases the need to understand the purpose and intended use of the model, is seen as of paramount importance. Taking purpose into account guides the modeller to choose an appropriate modelling paradigm and to then filter input data and interpret output data accordingly. By making appropriate filtering decisions, greater opportunities to utilise existing data become possible. By making informed and appropriate interpretations of process simulation model results, which are cognisant of the limitations of the data, useful insights can be gained to support project decisions. Hence, by supporting real software development projects the utility of modelling and simulation technologies can be demonstrated.

The software process involves human performing activities and making decisions at many levels and in different roles. This implies an intrinsic limitation in the accuracy and faithfulness of the process models and its outputs, whatever the modelling technique or approach, since the system being modelled (either real or hypothetical) contains humans, who think, act and learn. (Lehman, 1976). Despite this intrinsic limitation, experiences to date suggest that benefits derived from the *construction* and usage of process models seem to outweigh these limitations, including the cost of model building. The word construction is emphasised since experience, for example, in simulation with research and development purposes it has been the experience that a significant part of what the model construction team learns actually happens during model construction (Lehman, 1996).

Process modelling offer a systematic approach to assess process performance and to process design. Its appropriate combination with the approaches of empirical studies can contribute to the long-term goal of increasing understanding and the degree of intellectual and industrial mastery over the software process.

Acknowledgments

Grateful thanks are due to Ms Siew F Lim for her help in the proofreading of this paper.

Notes

1. The reader should note that the ISPW-6 process example describes the modification of one change request from design through unit test. As a result, a majority of the savings in effort and time to market (schedule) derived from fewer remaining defects are not shown in this model's output.

References

Abdel-Hamid, T., and Madnick, S. 1991. *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs, NJ: Prentice-Hall Software Series.

Basili, V. R., and Weiss, D. 1984. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering* SE-10(6): 728–738.

Birkes, D., and Dodge, Y. 1993. *Alternative Methods of Regression*. NY: Wiley.

Boehm, B. M. 1988. A spiral model of software development and enhancement. *IEEE Computer* 21(5): 61–72.

Booch, G., Rumbaugh, J., and Jacobson, I. 1999. *Unified Modeling Language User Guide*. Addison-Wesley.

Bradley, E., and Easley, M. 1997. Reasoning about sensor data for automated system identification. In *Advances in Intelligent Data Analysis, IDA-97* (Liu, X. and Cohen, P., eds.). Berlin: Springer Verlag.

Charnes, A., Cooper, W. W., and Rhodes, E. 1978. Measuring the efficiency of decision making units. *European Journal of Operational Research* 2(6): 429–444.

Charnes, A., Cooper, W. W., Lewin, A. Y., and Seiford, L. M. (eds.) 1994. *Data Envelopment Analysis: Theory, Methodology and Applications*. Boston: Kluwer.

Chatters, B. W., Lehman, M. M., Ramil, J. F., and Wernick, P. 1999. Modelling a software evolution process. *ProSim'99, Softw. Process Modelling and Simulation Workshop*. Silver Falls, Oregon, June 28–30.

Cook, J. E., Votta, L. G., and Alexander, W. L. 1998. Cost-effective analysis of in-place software processes. *IEEE Transactions on Software Eng.* SE-24(8): 650–663.

Curtis, B., Kellner, M. I., and Over, J. 1992. Process modeling. *Communications of the ACM* 35(9): 75–90.

Drappa, A., and Ludewig, J. 1999. Quantitative modeling for interactive simulation of software projects. *Journal of Systems and Software* 46(2/3): 113–122.

FEAST—Feedback, evolution and software technology. 1999. Dept. of Computing, Imperial College. Web page: http://www-dse.doc.ic.ac.uk/∼mml/feast

Fenton, N., and Pfleeger, S. 1997. *Software Metrics—A Rigorous & Practical Approach*, 2nd ed. London: ITP Press.

Harrison, W., Raffo, D. M., and Settle, J. 1999a. Measuring the value from improved predictions of software process improvement outcomes using risk-based discount rates. *Proc. Workshop on Economics Driven Software Engineering, International Conference on Software Engineering*. Los Angeles, California, May.

Harrison, W., Raffo, D. M., and Settle, J. 1999b. Quantitatively assessing the value of metrics and information in predicting process improvement performance. *Proc. Pacific Northwest Software Quality Conference*. Portland, Oregon, October 1999.

Kaltio, T., and Kinnula, A. 1998. Deploying the defined software process. *Conference Proceedings, SPI 98—European Conf. on Software Process Improvement*. Monte-Carlo.

Kaltio, T. 1999. Efficient SW process asset management and deployment in a multi-site organization. Ph.D. dissertation, to be published in 2000.

Kellner, M. I., Madachy, R. J., and Raffo, D. M. 1999. Software process simulation modeling: Why? What? How? *Journal of Systems and Software* 46(2/3): 91–105.

Kinnula, A. 1999. Software process engineering in a multi-site environment: An architectural design of a software process engineering system. Ph.D. dissertation, to be published in 1999.

Lehman, M. M. 1976. Human thought and action as an ingredient of system behaviour. Imp. Col of Sc. Tech., CCD Research Report 76/12, also in *The Encyclopedia of Ignorance*, 397–354. Pergamon Press.

Lehman, M. M. 1994. Feedback in the software evolution process. Keynote address. *CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics*. Dublin. *Workshop Proc., Information and Software Technology, sp. is. on Software Maintenance* 38(11): 681–686. Elsevier.

Lehman, M. M. 1996. Personal communication.

Levary, R. R. 1990. System dynamics with fuzzy logic. *Int. J. Systems Sci.* 21(8): 1701–1707.

Madachy, R. J. 1994. A software project dynamics model for process cost, schedule and risk assessment. Ph.D. dissertation, Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, California.

Miers, D. 1997. Use of tools and technology within a BPR initiative. In *Business Process Re-engineering: myth and reality* (C. Coulson-Thomas, ed.). Kogan Page.

Mitchell, T. 1997. *Machine Learning*. NY: Mc Graw Hill.

Ould, M. A. 1995. *Business Processes: Modelling and Analysis for Reengineering and Improvement*. Wiley.

Partridge, D. 1997. The case for inductive programming. *IEEE Computer* 30(1): 36–41.

Phalp, K. T., and Counsell, S. J. 1997. Counts and heuristics for the analysis of static models. *ICSE'97 Workshop on Process Modelling and Empirical Studies of Software Engineering*. Boston.

Phalp, K. T. 1998. The CAP framework for business process modelling. *Information and Software Technology* 40(13).

Paulk, M. C., et al. 1993. Capability maturity model, version 1.1. *IEEE Software* 10(4): 18–27.

Raffo, D. M. 1996. Modeling software processes quantitatively and assessing the impact of potential process changes on process performance. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Raffo, D. M., and Kellner, M. I. 1999. Predicting the impact of potential process changes: A quantitative approach to process modeling. In *Elements of Software Process Assessment and Improvement* (K. El Emam and N. Madhavji, eds.). Los Alamitos, California: IEEE Computer Society Press.

Raffo, D. M., Settle, J., and Harrison, W. 1999. Estimating the financial benefit and risk associated with process changes. *Proc. Workshop on Economics Driven Software and Systems Engineering, ICSE'99*. Los Angeles, California.

Ramil, J. F., and Lehman, M. M. 1998. Fuzzy dynamics in software project simulation and support. *Proc. EWSPT'6*. Weybridge, U.K. LNCS 1487, Springer Verlag, 122–126.

Ramil, J. F., and Lehman, M. M. 1999a. Modelling process dynamics in software evolution processes—Some issues. *ICSE 99 Workshop on Software Change and Evolution*. Los Angeles, CA.

Ramil, J. F., and Lehman, M. M. 1999b. Challenges facing data collection for support and study of software evolution processes. *ICSE 99 Workshop on Empirical Studies of Software Development and Evolution*, 28–32. Los Angeles, CA.

Roberts, N. et al. 1983. *Introduction to Computer Simulation—A System Dynamics Modelling Approach*. Portland, OR: Productivity Press.

Royce, W. W. 1970. Managing the development of large software systems: Concepts and techniques. *Proc. IEEE WESTCON*, 1–9. Los Angeles.

SEBPC—Systems Engineering for Business Process Change. 1998. UK EPSRC. Homepage at: http://www.staff.ecs.soton.ac.uk/~ph/sebpc/

Tausworthe, R. C. 1980. The work breakdown structure in software project management. *Journal of Systems and Software* 1:181–186.

Tesoreiro, R., and Zelkowitz, M. 1998. A model of noisy software engineering data. *Proc. ICSE'20*, 461–464. Kyoto, Japan, April 19–25.

Tukey, J. W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

Wang, W., Jones, P., and Partridge, D. 1998. Pattern recognition features for neural networks. *Proc. ICAPR98*, 232–241. Plymouth, UK.

Wang, W., Jones, P., and Partridge, D. 1999. Assessing the impact of input features in a feed-forward network. Submitted for publication.

Yourdon, E. 1994. *Object-Oriented Systems Design: An Integrated Approach*. Prentice Hall.