Paper appeared in: Kellner, Madachy, and Raffo, "Software Process Modeling and Simulation: Why, What, How," Journal of Systems and Software, Vol. 46, No. 2/3 (15 April 1999).

Software Process Simulation Modeling: Why? What? How?

Marc I. Kellner[*], Raymond J. Madachy[†], and David M. Raffo[‡]

[*] Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
mik@sei.cmu.edu

[†] USC Center for Software Engineering
University of Southern California
Los Angeles, CA 90089-0781
madachy@usc.edu

[‡] School of Business Administration
Portland State University
Portland, OR 97207-0751
davidr@sba.pdx.edu

Abstract

Software process simulation modeling is increasingly being used to address a variety of issues from the strategic management of software development, to supporting process improvements, to software project management training. The scope of software process simulation applications ranges from narrow focused portions of the life cycle to longer-term product evolutionary models with broad organizational impacts. This article provides an overview of work being conducted in this field. It identifies the questions and issues that simulation can be used to address ("why"), the scope and variables that can be usefully simulated ("what"), and the modeling approaches and techniques that can be most productively employed ("how"). It includes a summary of the papers in this special issue of the *Journal of Systems and Software*, which were presented at the First International Silver Falls Workshop on Software Process Simulation Modeling (ProSim'98). It also provides a framework that helps characterize work in this field, and applies this new characterization scheme to many of the articles in this special issue. This paper concludes by offering some guidance in selecting a simulation modeling approach for practical application, and recommending some issues warranting additional research.

1 Introduction

Over the past few decades the software industry has been assailed by numerous accounts of schedule and cost overruns as well as poor product quality delivered by software development organizations in both the commercial and government sectors. At the same time, increasing customer demands for "better, faster, cheaper" and the increasing complexity of software have significantly "raised the bar" for software developers to improve performance.

The industry has received help in the form of a plethora of case tools, new computer languages, and more advanced and sophisticated machines. A key question is, "How can the tools, technologies, and people work together in order to achieve these increasingly challenging goals?" Potential answers to this question imply changes to the software development process or software organization. Possible changes will require a significant amount of resources to implement and have significant implications on the firm – good or bad. How can organizations gain insights into potential solutions to these problems and their likely impacts on the organization? One area of research that has attempted to address these questions and has had some success in predicting the impact of some of these proposed solutions is software process simulation modeling.

Software process simulation modeling is gaining increasing interest among academic researchers and practitioners alike as an approach for analyzing complex business and policy questions. Although simulation modeling has been applied in a variety of disciplines for a number of years, it has only recently been applied to the area of software development and evolution processes.

Currently, software process simulation modeling is beginning to be used to address a variety of issues from the strategic management of software development, to supporting process improvements, to software project management training. The scope of software process simulation applications ranges from narrow focused portions of the life cycle to longer-term product evolutionary models with broad organizational impacts. This article provides an overview of work in the field of software process simulation modeling. Moreover, it identifies the questions and issues that simulation can be used to address (i.e., why simulate), the scope and variables that can be usefully simulated (i.e., what to simulate), and the modeling approaches and techniques that can be most productively employed (i.e., how to simulate). This article focuses on the "big picture" and offers the reader a broad perspective. In addition, it introduces this special issue of the *Journal of Systems and Software* containing papers from ProSim'98 and it provides a framework to help characterize work in this field. Table 1 lists the papers included in this issue.

2 Background

This section briefly discusses the key words from the title: "process", "model", and "simulation", and what they mean when they are put together. An organizational (e.g., business) ***process*** is a logical structure of people, technology, and practices that are organized into work activities designed to transform information, materials, and energy into specified end result(s) (adapted from (Pall, 1987)). Examples of organizational / business processes include software development and evolution, systems engineering, engineering design, travel expense reimbursement, work selection and funding, and project management, to name but a few. This article focuses on software processes (i.e., those used for software development and evolution), although much of what is said applies equally well to many other business processes – particularly those involving design and engineering. A ***software process*** has been specifically defined as "A set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals)." (Paulk et al., 1993).

A ***model*** is an abstraction (i.e., a simplified representation) of a real or conceptual complex system. A model is designed to display significant features and characteristics of the system, which one wishes to study, predict, modify, or control. Thus a model includes some, but not all, aspects of the system being modeled. A model is valuable to the

Author	Title
Kellner, Madachy, and Raffo	"Software Process Simulation Modeling: Why? What? How?"
Christie	"Simulation in Support of CMM-based Process Improvement"
Drappa and Ludewig	"Quantitative Modeling for the Interactive Simulation of Software Projects"
Lehman and Ramil	"The Impact of Feedback in the Global Software Process"
Pfahl and Lebsanft	"Integration of System Dynamics Modelling with Descriptive Process Modelling and Goal-Oriented Measurement"
Powell, Mander, and Brown	"Strategies for Lifecycle Concurrency and Iteration: A System Dynamics Approach"
Raffo, Vandeville, and Martin	"Software Process Simulation to Achieve Higher CMM Levels"
Rus, Collofello, and Lakey	"Software Process Simulation for Reliability Management"
Scacchi	"Experience with Software Process Simulation and Modeling"
Wernick and Lehman	"Software Process White Box Modelling for FEAST/1"
Williford and Chang	"Modeling Fed Ex's IT Division: A System Dynamics Approach to Strategic IT Planning"

Table 1: List of Papers in this Special Issue

extent that it provides useful insights, predictions, and answers to the questions it is used to address.

A ***simulation model*** is a computerized model that possesses the characteristics described above and that represents some dynamic system or phenomenon. One of the main motivations for developing a simulation model or using any other modeling method is that it is an inexpensive way to gain important insights when the costs, risks, or logistics of manipulating the real system of interest are prohibitive. Simulations 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 any of the following forms:

- System uncertainty and stochasticity – The risk or uncertainty of the system is an essential feature to consider. Bounds on this uncertainty and the implications of potential outcomes must be understood and evaluated. Analytical models have constraints on the number and kind of random variables that can be included in various types of models. Simulation provides a flexible and useful mechanism for capturing uncertainty related to complex systems without these restrictive constraints.
- Dynamic behavior – System behavior can change over time. For example, key variables such as productivity and defect detection rates can change over time. A dynamic model is necessary when accounting for or controlling these changes is important. However, analytic techniques such as dynamic programming can become intractable when the system being modeled becomes overly complex. Dynamic simulation models are very flexible and support modeling of a wide variety of system structures and dynamic interactions.
- Feedback mechanisms – Behavior and decisions made at one point in the process impact others in complex or indirect ways and must be accounted for. For example, in a software development process the decision to hire or not to hire a new employee has multiple impacts and implications over the entire course of the project. The decision to inspect requirements or not also has many implications. For complex feedback or feedforward systems, analytic models may not be useful or even possible. Simulation enables a more usable alternative.

Common purposes of simulation models are to provide a basis for experimentation, predict behavior, answer "what if" questions, teach about the system being modeled, etc. Such models are usually quantitative, although this is not always the case.

A ***software process simulation model*** focuses on some particular software development / maintenance / evolution process. It 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 developer to be especially relevant to the issues and questions the model is used to address.

The papers in this special issue provide a good sampling of recent work in the area of software process simulation modeling. The bibliographies of these articles provide an extensive set of additional references to related work. Other significant work in this field includes (Abdel-Hamid and Madnick, 1991; Akhavi and Wilson, 1993; Burke, 1997; Gruhn, 1992; Gruhn 1993; Kellner, 1991; Kusumoto, 1997; Madachy, 1994; Raffo, 1996; Raffo and Kellner, 1999; Tvedt, 1996) and others.

3 Why Simulate?

There is a wide variety of reasons for undertaking simulations of software process models. In many cases, simulation is an aid to decision making. It also helps in risk reduction, and helps management at the strategic, tactical, and operational levels. We have clustered the many reasons for using simulations of software processes into six categories of purpose:

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

When developing software process simulation models, identifying the purpose and the questions / issues management would like to address is central to defining the model scope and data that need to be collected. In the following paragraphs, examples of practical questions and issues that can be addressed with simulation are presented for each of these six purpose categories.

3.1 Strategic Management

Simulation can help address a broad range of strategic management questions, such as the following. Should work be distributed across sites or should it be centralized at one location? Would it be better to perform work in-house or to out-source (subcontract) it? To what extent should commercial-off-the-shelf (COTS) components be utilized and integrated, as opposed to developing custom components and systems? Would it be more beneficial to employ a product-line approach to developing similar systems, or would the more traditional, individual product development approach be better suited? What is the likely long-term impact of current or prospective policies and initiatives (e.g., hiring practices, training policies, software process improvement initiatives)? In each of these cases, simulation models would contain local organizational parameters and be developed to investigate specific questions. Managers would compare the results from simulation models of the alternative scenarios to assist in their decision making. One article focusing on strategic questions in this special issue is "Modeling Fed Ex's IT Division: A System Dynamics Approach to Strategic IT Planning" by Williford and Chang.

3.2 Planning

Simulation can support management planning in a number of straightforward ways, including

- forecast effort / cost, schedule, and product quality
- forecast staffing levels needed across time
- cope with resource constraints and resource allocation
- forecast service-level provided (e.g., for product support)
- analyze risks

All of these can be applied to both initial planning and subsequent re-planning. Papers in this special issue by Christie, Pfahl and Lebsanft, Powell et al., Rus et al., and Williford and Chang address various issues falling into this category.

In the preceding cases, the process to be used is taken as fixed. However, simulation can also be used to help select, customize, and tailor the best process for a specific project context. These are process planning issues. Some representative questions in this category are

- Should process A or process B be used on our new project?
- Will process C enable us to reach our productivity goals for this new project?
- What portions of process D can be minimized or skipped to save cost and schedule, without seriously sacrificing quality?

In these cases, simulations would be run of the process (or alternatives) under consideration, and their results evaluated along the dimensions of interest (often cost, cycle-time, and quality) in order to answer the questions posed.

3.3 Control and Operational Management

Simulation can also provide effective support for managerial control and operational management. Simulation can facilitate project tracking and oversight because key project parameters (e.g., actual status and progress on the work products, resource consumption to-date, and so forth) can be monitored and compared against planned values computed by the simulation. This helps project managers determine when possible corrective action may be needed. By using Monte Carlo simulation techniques (see section 5.2), probability distributions can be developed for important planned variables (e.g., milestone dates for key events, such as design approval, start of integration testing, etc.; effort consumption through to key events or planned dates; defect levels detected at key points). Statistical techniques such as control charts can then be used to determine whether corrective action is needed.

Project managers can also use simulation to support operational decisions, such as whether to commence major activities (e.g., coding, integration testing). To do this, managers would evaluate current project status using current actual project data and employ simulation to predict the possible outcome if a proposed action (e.g., commence integration testing) was taken then or delayed. Simulation can also be employed in operational management decisions to modify planned work priorities, detailed allocation of staff, scheduling, hiring needs, etc., based on current circumstances and projected outcomes. For examples of papers in this domain, see Pfahl and Lebsanft, and Powell et al. in this issue.

It should be noted that models used for operational monitoring and control require up-to-date detailed project data to be meaningful (see sections 4.4 and 4.2 for common input and result parameters, section 5.3 for issues related to metrics and data collection for simulation models, also see (Raffo, Harrison, and Keast, 1998) for further information on developing operational planning and control models vs. planning models).

3.4 Process Improvement and Technology Adoption

Simulation can support process improvement and technology adoption in a variety of ways. In process improvement settings, organizations are often faced with many suggested improvements. Simulation can aid specific process improvement decisions (such as go / no-go on any specific proposal, or prioritization of multiple proposals) by forecasting the impact of a potential process change before putting it into actual practice in the organization. Thus, simulation can assist in the engineering of standard, recommended processes. These applications use simulation *a priori* to compare process alternatives, by comparing the projected outcomes of importance to decision makers (often cost, cycle-time, and quality) resulting from simulation models of alternative processes. These same simulation techniques were discussed in section 3.2 in the context of process planning, to help select, customize and tailor the best process for a specific context. Although the techniques are much the same, the purpose here is process improvement, in contrast to planning. Simulation can also be used *ex post* to evaluate the results of process changes or selections already implemented. Here, the actual results observed would be compared against simulations of the processes not selected, after updating those simulations to reflect actual project characteristics seen (e.g., size, resource constraints). The actual results would also be used to calibrate the model of the process that was used, in order to improve future applications of that model. A number of articles in this special issue fit into this area including Christie, Pfahl and Lebsanft, Powell et al., Raffo et al., and Scacchi.

Just as organizations face many process improvement questions and decisions, the same is true for technology adoption. The analysis of inserting new technologies into a software development process (or business process) would follow the same approach as for process change and employ the same basic model. This is largely because adoption of a new technology is generally expected to affect things that are usually reflected as input parameters to a simulation (e.g., defect injection rate, coding productivity rate) and / or to change the associated process in other more fundamental ways.

3.5 Understanding

Simulation can promote enhanced understanding of many process issues. For example, simulation models can help people – such as project managers, software developers, and quality assurance personnel – better understand process flow, i.e., sequencing, parallelism, flows of work products, etc. Animated simulations, Gantt charts, and the like are useful in presenting simulation results to help people visualize these process flow issues. Simulations can also help people to understand the effects of the complex feedback loops and delays inherent in software processes; even experienced software professionals have difficulty projecting these effects by themselves due to the complicated interactions over time. (For example, how many project managers resist inspections or thorough testing because of their seeming effect of extending schedule, failing to appreciate that later savings will usually more than compensate for the short-term loss?) In addition, simulation models can help researchers to identify and understand consistent, pervasive properties of software development and maintenance processes (a.k.a. software laws). Moreover, simulations (especially Monte Carlo techniques) can help people understand the inherent uncertainty in forecasting software process outcomes, and the likely variability in actual results seen. Finally, simulations help facilitate communication, common understanding, and consensus building within a team or larger organization. All simulation models help with process or organizational understanding to some degree. In particular the papers of Drappa and Ludewig, Lehman and Ramil, Powell et al., Raffo et al., Scacchi, and Wernick and Lehman fall into this category.

3.6 Training and Learning

Simulation can help with training and learning about software processes in several ways. Although this purpose cluster is closely related to that of "understanding", the particular setting envisioned here is an explicitly instructional one. Simulations provide a way for personnel to practice / learn project management; this is analogous to pilots practicing on flight simulators. A simulated environment can help management trainees learn the likely impacts of common decisions (often mistakes), e.g., rushing into coding, skipping inspections, or reducing testing time. Finally, training through participation in simulations can help people to accept the unreliability of their initial expectations about the results of given actions;

most people do not possess good skills or inherent abilities to predict the behavior of systems with complex feedback loops and / or uncertainties (as are present in software processes). Overall, active participation in a good mix of simulations can provide learning opportunities that could otherwise only be gained through years of real-world experience. The articles by Christie and by Drappa and Ludewig in this special issue discuss these applications further.

4 What to Simulate?

Having reviewed the major reasons for undertaking simulations of software processes, we now turn to a discussion of what to simulate. In general, the basic purpose of the simulation model (why simulate), coupled with the specific questions or issues to be addressed, will largely determine what to simulate (See Figure 1). As can be seen in Figure 1, many aspects of what to simulate are inter-related and driven based on the model purpose and key questions described in section 3. The following paragraphs discuss aspects of what to simulate in terms of (1) model scope, (2) result variables, (3) process abstraction, and (4) input parameters, respectively.

4.1 Model Scope

Determining model scope is an important issue and an iterative process. The scope of the model needs to be large enough to fully address the key questions posed. This requires understanding the implications of the questions being addressed. For example, a company may be considering a process change that is localized to the code and unit test steps. They would like to develop a model to predict the impact of this change on overall project performance. Though the change being made to the process is limited to the code and unit test steps, the change will likely have implications for the quality and effort performance of later phases of testing. Therefore, the scope of the model needs to reflect these expanded impacts of the process change. Moreover, in order to address the key questions related to “overall project performance”, result variables will need to be identified to predict specific performance dimensions at the required levels. Consideration of these result variables may in turn lead to reconsideration of the model scope needed, and so on in an iterative fashion.

The scope of a software process simulation is generally one of the following:

- a portion of the life cycle (e.g., design phase, code inspection, some or all of testing, requirements management)
- a development project (i.e., single product

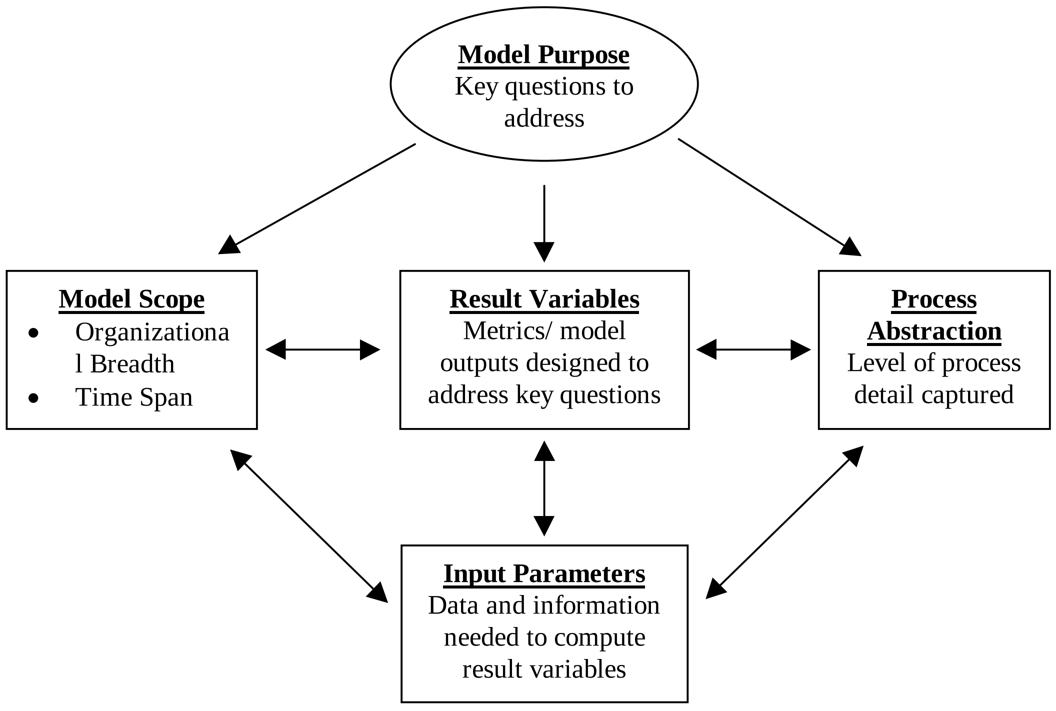
Model Purpose
Key questions to address
Model Scope
• Organizational Breadth
• Time Span
Result Variables
Metrics/ model outputs designed to address key questions
Process Abstraction
Level of process detail captured
Input Parameters
Data and information needed to compute result variables

Figure 1: Relationships Among “Why” and “What” Aspects

development life cycle)
- multiple, concurrent projects (e.g., across a department or division)
- long-term product evolution (i.e., multiple, successive releases of a single product)
- long-term organization (e.g., strategic organizational considerations spanning successive releases of multiple products over a substantial time period)

Underlying these five scope categories are two dimensions: time span and organizational breadth. We can think of time span with some approximate guidelines as

- short (less than 12 months),
- medium (between 12 - 24 months; about the duration of one development project or maintenance cycle), or
- long (more than 24 months)

We think of organizational breadth as pertaining to

- less than one product / project team
- one product / project team
- multiple product / project teams

Table 2 shows how the five scope categories can be mapped onto the dimensions of time span and organizational breadth. Two cells are unnamed because people do not generally develop simulations with those scopes. One scope category (portion of life cycle) fits three cells in the table. For example: design is (short, =1); inspection is (short, <1); requirements management is (medium, <1) because it goes on through most of the project but only involves a portion of the team.

Some of the purpose categories have significant implications for the selection of scope. For example, a purpose of "strategic management" likely implies the scope "long-term organization"; certainly this would be the case for the product-line approach question posed above (section 3.1). Nevertheless, some other "strategic management" issues and questions may be asked in relation to as small a scope as a "development project"; an example is provided by the out-sourcing question posed above (section 3.1). In essence, the question and its context must be well understood by the model developer in order to determine the appropriate scope for the simulation.

4.2 Result Variables

The result variables are the information elements needed to answer the key questions that were specified along with the purpose of the model. Depending on the key questions being asked, a great many different variables could be devised as the results of a process simulation. However, the most typical result variables for software process simulation models include the following:

- effort / cost
- cycle-time (a.k.a. duration, schedule, time to market, interval)
- defect level
- staffing requirements over time
- staff utilization rate
- cost / benefit, return on investment (ROI), or other economic measures (usually of interest for changes)
- throughput / productivity
- queue lengths (backlogs)

Once again, the questions and issues being addressed largely determine the choice of result variables. For example, technology adoption questions often suggest an economic measure such as ROI; operational management issues often focus on the traditional project management concerns of effort / cost, cycle-time, and defect levels. It should also be noted that it is quite common to produce multiple result variables from a single simulation model.

Organizational Breadth	**Time Span**		
	Short	**Medium**	**Long**
< 1 product / project	portion of life cycle	portion of life cycle	
= 1 product / project	portion of life cycle	development project	long-term product evolution
> 1 product / project		multiple, concurrent projects	long-term organization

Table 2: Model Scope by Organizational Breadth and Time Span

When defining the result variables for a model, it is important to specify their scope. For example, are managers interested in predictions of overall end-of-project effort? If so, the scope of the model may need to include the full process life cycle. At a minimum, the model needs to capture all process steps after the point that changes are made. This may cause the scope of the model to be modified if it has already been defined. In most cases, as the purpose and key questions of the model are refined, model scope and result variables get refined in an iterative process. In addition, some result variables are best looked at as continuous functions (e.g., staffing requirements), while others only make sense at the end of the process being simulated (e.g., ROI).

Definition of the result variables has implications for the level of abstraction of the model and model input parameters as well. For example, in addition to predicting end-of-project effort, suppose management was also interested in effort predictions for each of the major life-cycle steps. Perhaps management is interested in detailed effort estimates for process steps within the major life-cycle steps. Requesting these result variables has significant implications for the level of abstraction of the model as well as the kinds of data and input parameters that need to be developed.

4.3 Process Abstraction

When planning and developing a simulation model, the model builder needs to identify the key elements of the process, their inter-relationships, and behavior, for inclusion in the model. The focus should be on those aspects of the process that are especially relevant to the purpose of the model, and believed to affect the result variables. For example, it is important to identify

- key activities and tasks
- primary objects (e.g., code units, designs, problem reports)
- vital resources (e.g., staff, hardware)
- activity dependencies, flows of objects among activities, and sequencing
- iteration loops, feedback loops, and decision points
- other structural interdependencies (e.g., the effort to revise a code unit after an inspection, as a function of the number of defects found during the inspection)

These should then be represented in the simulation model when it is developed.

4.4 Input Parameters

The input parameters (key factors, drivers, independent variables) to include in the model largely depend upon the result variables desired and the process abstractions identified. Typically, many parameters are needed to drive software process simulation models; some require a few hundred parameters (e.g., Abdel-Hamid and Madnick, 1991). Examples of several typical input parameters are provided below:

- amount of incoming work (often measured in terms of software size (LOC) or function points, etc.)
- effort for design as a function of size
- defect detection efficiency during testing or inspections
- effort for code rework as a function of size and number of defects identified for correction
- defect removal and injection rates during code rework
- decision point outcomes; number of rework cycles
- hiring rate; staff turnover rate
- personnel capability and motivation, over time
- amount and effect of training provided
- resource constraints
- frequency of product version releases

Some of these are usually treated as constant over the course of a simulation (e.g., the functions for effort), while others are often treated as varying over time (e.g., personnel capability and motivation). It is also noteworthy that inter-dependencies among some of these parameters may be represented in the model, rather than treating them as independent variables. For example, personnel capability may be influenced by staff turnover and hiring rates.

5 How to Simulate?

After determining why and what to simulate, many technical options, considerations, and issues remain for the model developer. These are outlined below under the headings of (1) simulation approaches and languages, (2) simulation techniques, and (3) data / measurement issues.

5.1 Simulation Approaches / Languages

Various modeling approaches have been used to investigate different aspects of the software process. Despite best intentions, the implementation approach often influences what is being modeled. Therefore, the most appropriate approach to use will be the one best suited to the particular case at hand, i.e., the purpose, questions, scope, result variables desired, etc.

A variety of simulation approaches and languages have been applied to software processes, including

- state-based process models
- general discrete event simulation
- system dynamics (or continuous simulation)
- rule-based languages
- Petri-net models
- queueing models
- project management (e.g., CPM and PERT)
- scheduling approaches (from management science, manufacturing, etc.)

Each of the first four categories are represented and discussed further in other articles in this special journal issue.

5.2 Simulation Techniques

The purpose of this section is to highlight some of the important capabilities and techniques that should be considered when selecting a particular simulation tool. This set of considerations is based upon the authors' collective experience in employing a variety of tools in the course of developing software process simulations in industrial settings.

The model may be depicted in a form that is primarily visual or textual. Visual models (i.e., graphical, diagrammatic, or iconic) have become the norm for software process simulations because they promote understandability and ease of development. These tools often include an ability to animate the model during simulation to show the flows of objects (e.g., code units, designs, problem reports) through the process, the activities currently being performed, and so forth. This can be very helpful as a further aid to understanding, and as a tool to use during validation of the model with process experts.

Even when a model is primarily visual, it almost invariably entails supplemental textual information specifying interrelationships among components, equations, distributions for random variables, etc. As a result, suitable repository capabilities that can store, present, and report this information easily is highly desirable. Another useful capability offered by many simulation tools is the simultaneous reporting of results (usually plots of result variables over time) while the model is executing.

Many tools support interactive simulation, where the user specifies or modifies some of the input parameters during model execution rather than only beforehand, and / or can step the model through its execution. Some tools also allow batches of simulations to be executed automatically from a single set-up, and results accumulated across the individual runs in the batch.

A simulation can be deterministic, stochastic, or mixed. In the deterministic case, input parameters are specified as single values (e.g., coding for this unit will require 5 work-days of effort, or 4 hours per hundred lines of code; there will be two rework cycles on code and unit test; etc.). Stochastic modeling recognizes the inherent uncertainty in many parameters and relationships. Rather than using (deterministic) point estimates, stochastic variables are random numbers drawn from a specified probability distribution. Mixed modeling employs both deterministic and stochastic parameters.

In a purely deterministic model, only one simulation run is needed for a given set of parameters. However, with stochastic or mixed modeling the result variables differ from one run to another because the random numbers actually drawn differ from run to run. In this case the result variables are best analyzed statistically (e.g., mean, standard deviation, distribution form) across a batch of simulation runs; this is termed Monte Carlo simulation. Although many process simulation tools support stochastic models, only some of them conveniently support Monte Carlo simulation by handling batches well.

Finally, sensitivity analysis is a very useful technique involving simulation models. Sensitivity analyses explore the effects on the key result variables, of varying selected parameters over a plausible range of values. This allows the modeler to determine the likely range of results due to uncertainties in key parameters. It also allows the modeler to identify which parameters have the most significant effects on results, suggesting that those be measured and / or controlled more carefully. As a simple example, if a 10% increase in parameter A leads to a 30% change in a key result variable, while a 10% increase in parameter B leads to only a 5% change in that result variable, one should be somewhat more careful in specifying or controlling parameter A. Sensitivity analysis is applicable to all types of simulation modeling. However, it is usually performed by varying the parameters manually, since there are few simulation tools that automate sensitivity analyses.

5.3 Data / Measurement Issues

Simulation models should undergo calibration and validation to the extent possible. Validation can be performed, in part, by inspections and walkthroughs of the model (providing what is often referred to as face validity). In addition, actual data should be used to validate the model empirically and to calibrate it against real-world results. Considerations of the questions to be addressed, desired result variables, input parameters, validation, and calibration, often suggest metric data (measurements) that would be valuable to collect, and show how they are useful. Unfortunately a lack of relevant, desired data in practical settings is all too common.

Accurate simulation results depend on accurate values of the parameters. Similarly, calibration also depends on accurate measurement of the results seen in practice. In many "real world" settings these variables have not been measured, or at least not carefully measured. Strategies that can be useful in coping with this situation include the following (Kellner and Raffo, 1997):

- Adjust existing values to approximate desired variables, e.g., if cost is desired but only effort is available, one can generally approximate cost using an average hourly rate.
- Construct values from other detailed records, e.g., data on when defects were injected (by phase) may not be available as part of summary reports from inspections, but might be found on the original, detailed, inspection forms.
- Obtain estimates from personnel involved, based on their past experience or hypothetical expectations when necessary.
- Utilize typical values taken from the literature, e.g., defect detection efficiencies for source code inspections and unit tests, measured in other organizations, are widely available.

6 Characterizing Software Process Simulations

Specific simulation models of software processes will differ widely along any of the dimensions discussed in sections 3 through 5 (each of the why, what, and how sub-categories). It is our opinion that the most important dimensions for broadly characterizing a simulation model are

- purpose ("why")
- scope ("what")
- key result variables ("what")
- simulation approach / language employed ("how")

Characterizing different modeling work reported in the literature is a first step toward understanding the major differences in the various work that has been conducted. To support such a characterization, we use the taxonomies presented in the preceding sections of this article along the four dimensions just proposed. To that end, we have developed a characterization grid, which is introduced below.

As shown in Figure 2, we have used purpose and scope as the two dimensions of the basic grid. We include the key result variables as appropriate for each cell (a cell being a specific purpose and scope combination). We also note the simulation approach / language employed in each model.

By way of example, the authors have used this grid to characterize some of our own past work. Figure 2 shows the work reported in (Kellner, 1991; Madachy, 1994; and Raffo, 1996) (the latter two represent Ph.D. dissertations). As we have applied this scheme, we have striven to indicate only those cells and results actually reported. However, the grid could also be used to characterize what a particular simulation approach / language is well suited for.

Kellner's 1991 article focused on showing how simulation of process models could be used to support management planning and control. The Statemate®[1]-based process modeling approach used (Kellner, 1989; Curtis et al., 1992) also supports thorough understanding of the process, which is facilitated by the multi-perspective diagrammatic model representation, animation, interactive as well as batch simulation, and (more recently) simultaneous reporting of results. The paper (Kellner, 1991) demonstrated deterministic as well as stochastic modeling, the former with single simulation runs and the later with the Monte Carlo batch approach.

Madachy's 1994 dissertation used system dynamics to model the effect of performing formal inspections. In particular, the cost, schedule and quality impacts were compared for varying degrees of conducting inspections throughout the life cycle. It modeled the interrelated flows of tasks, errors and personnel throughout different development phases, detailed inspection activities, and was calibrated to industrial data.

Raffo's 1996 dissertation focused on showing how software process simulation modeling could be used to support process improvement, by forecasting the impact of potential process changes before they are actually implemented in the organization. This work contributed many substantial extensions and improvements to the techniques originally reported

[1] Statemate is a registered trademark of i-Logix Inc., Andover, Massachusetts. (http://www.ilogix.com)

Scope / **Purpose**	Portion of life cycle	Development project	Multiple, concurrent projects	Long-term product evolution	Long-term organization
Strategic management					
Planning	**Kellner, 1991: (SB)** effort, cycle-time, staffing profile				
Control and operational management	**Kellner, 1991: (SB)** effort, cycle-time, staffing profile				
Process improvement and technology adoption	**Madachy, 1994: (SD)** effort, cycle-time, defect levels, staffing profile, ROI **Raffo, 1996: (SB)** effort, cycle-time, defect levels, utility	**Madachy, 1994: (SD)** effort, cycle-time, defect levels, staffing profile, ROI **Raffo, 1996: (SB)** effort, cycle-time, defect levels; overall project cost, quality, and schedule performance, uncertainty/ risk, and ROI			
Understanding	**Kellner, 1991: (SB)** effort, cycle-time, staffing profile	**Raffo, 1996: (SB)** effort, cycle-time, defect levels; overall project cost, quality, and schedule performance, uncertainty/ risk, and ROI			
Training and learning					

Legend for modeling approaches used: (SB) = state-based, (SD) = system dynamics

Note: In all of these cases, effort, cycle-time, and defect levels were computed for the full scope indicated, as well as for its constituent phases and / or activities

Figure 2: Characterization of Authors' Past Work

in (Kellner, 1991), as well as applying them to a different purpose, broader scope, and with additional result variables – which can be clearly seen in Figure 2.

Scope / **Purpose**	Portion of life cycle	Development project	Multiple, concurrent projects	Long-term product evolution	Long-term organization
Strategic management					**Williford and Chang: (SD)** staffing (permanent and contract); cost factors for IT infrastructure, tools, training, recruiting, and compensation
Planning		**Rus et al.: (SD)** defects, effort, reliability factors **Pfahl and Lebsanft: (SD)** defects, cycle-time, effort	**Powell et al.: (SD)** achievable project timetables and quality; degree of project, delivery, and phase concurrency		**Williford and Chang: (SD)** see above for result variables
Control and operational management		**Pfahl and Lebsanft: (SD)** defects, cycle-time, effort	**Powell et al.: (SD)** see above for result variables		
Process improvement and technology adoption	**Raffo et al.: (SB)** cost, quality, cycle-time	**Pfahl and Lebsanft: (SD)** defects, cycle-time, effort	**Powell et al.: (SD)** see above for result variables		
Understanding	**Raffo et al.: (SB)** cost, quality, cycle-time	**Drappa and Ludewig: (RB)** percent of function points implemented, defects, effort, and duration	**Powell et al.: (SD)** see above for result variables	**Wernick and Lehman: (SD)** system size and effort trends	
Training and learning		**Drappa and Ludewig: (RB)** see above for result variables			

Legend for modeling approaches used: (RB) = rule-based, (SB) = state-based, (SD) = system dynamics

Figure 3: Characterization of Seven Articles in this Special Issue

Finally, Figure 3 shows our placement of many of the articles included in this special issue on our characterization grid. (Once again, the articles are listed in full in Table 1.) It is readily apparent that these articles cover a wide variety of applications, thereby demonstrating the value of software process simulation modeling in a broad range of problem settings. Although the grid is appropriate for characterization of specific models, it is not suited to description of more abstract papers discussing various applications. Accordingly, the articles in this issue by Christie, by Lehman and Ramil, and by Scacchi are not amenable to inclusion in this grid. Note that only the major purposes and scopes of the simulations are shown for each article, since most of them can also be expanded to other cells in terms of secondary results and / or future extensions.

7 Summary of Special Issue Articles

This section briefly summarizes the articles in this issue that were presented at ProSim'98. The paper summaries appear alphabetically by author name. As illustrated in Figure 3, these papers address a variety of purposes (why), scopes and result variables (what), and modeling approaches (how). Figure 3 expounds on the summaries by showing the various dimensions upon which the simulation work can be classified as well as the primary result variables.

In "Simulation in Support of CMM-based Process Improvement" Alan Christie from the Software Engineering Institute (SEI) reviews how software process simulation can be applied at all levels of the CMM®[2]. At the lowest level, simulation can help improve awareness of dynamic process behavior on process performance. As process maturity improves, simulation is increasingly tied to operational metrics, to validate one's understanding of process behavior and improve the simulations' predictive power. This predictive power allows new processes to be tested off-line and introduced with much greater confidence.

Drappa and Ludewig's article, "Quantitative Modeling for the Interactive Simulation of Software Projects", presents current research on the SESAM project (Software Engineering Simulation by Animated Models). The goal of the project is to address the problems facing many software development projects, such as schedule and cost overruns as well as product quality issues, through improved training. Students using the simulator can control the simulated project interactively, making decisions about various process and product attributes, and ultimately making the project more or less successful. The model is expressed in a newly defined, rule-based modeling language. The paper discusses the simulation system, the model-description language, and a new quality assurance model.

In their paper, "The Impact of Feedback in the Global Software Process", Lehman and Ramil address general and fundamental issues relating to modeling software processes. One key focus is identifying essential activities that often underlie all software processes. To that end, they discuss how the presence of feedback loops impact or drive the evolution of most computer systems being used in the real world today. As a result, when modeling and simulating a software process to assess its performance and improve it, feedback loops as well as mechanisms and controls that impact the process characteristics significantly impact the outcomes and should be included in the model. This includes feedback loops as well as mechanisms and controls that originate outside the process. Through the FEAST/1 project, a number of industrial software processes using several different modeling and simulation techniques are being modeled and studied. See the companion paper by Wernick and Lehman for a system dynamics approach.

Over the last ten years, system dynamics modeling has been applied in software organizations to help compare process alternatives and to support project planning. In the paper "Integration of System Dynamics Modelling with Descriptive Process Modelling and Goal-Oriented Measurement", Pfahl and Lebsanft present methods for combining system dynamics modeling with the Descriptive Process Modeling (DPM) and measurement-based Quantitative Modeling (QM) approaches. This new approach, called IMMS (Integrated Measurement, Modeling and Simulation), is based on lessons learned that were derived from an industrial system dynamics modeling activity. The approach helps to address the need for a well defined and repeatable procedure for generating and using information based on experience and empirical evidence.

Improving time to market and development efficiency are topics of interest to all software development firms. In "Strategies for Lifecycle

[2] CMM is a registered (in the U.S. Patent and Trademark Office) mark of Carnegie Mellon University. (http://www.sei.cmu.edu)

Concurrency and Iteration: A System Dynamics Approach", Powell, Mander, and Brown present the use of a system dynamics simulation model to evaluate the use of advanced software life-cycle techniques for accelerating software development schedules. The paper describes work conducted by Rolls-Royce York University Technology Centre to investigate strategies for optimizing the degree and manner in which concurrent software engineering and staged-delivery life-cycle techniques should be applied on a collection of partially interdependent software development projects. This is done with the goal of meeting competitive demands on both product time-to-market and business performance.

In "Software Process Simulation to Achieve Higher CMM Levels", Raffo, Vandeville, and Martin present research utilizing state-based stochastic simulation models of a large-scale, real-world, software development process. The goal of the work was to provide a quantitative analysis of proposed process change alternatives in terms of key performance measures of cost, quality, and schedule. The model also supports a quantitative assessment of risk or uncertainty associated with each alternative. The approach is viewed as being key to successfully achieving higher level CMM practices related to Quantitative Process Management and Software Quality Management (CMM - Level 4) as well as Process and Technology Change Management (CMM - Level 5).

Rus, Collofello, and Lakey's paper, "Software Process Simulation for Reliability Management" discusses a variety of issues related to software reliability engineering and current efforts at Boeing's St. Louis site to model complex development processes using a variety of tools including expert systems, as well as system dynamics and discrete event simulation models. The paper describes a process simulator for evaluating how different software reliability strategies could be used. The process simulator is a part of a decision support system that can assist project managers in planning or tailoring the software development process, in a quality driven manner. The expert system is a rule-based fuzzy logic system, and the process simulator is developed using the system dynamics modeling paradigm.

In his paper, "Experience with Software Process Simulation and Modeling", Walt Scacchi presents an overview of the research that has been conducted through the USC System Factory project and the USC ATRIUM Laboratory over the past 10 years. This research has investigated various kinds of modeling and simulation issues related to understanding the organizational processes involved in software system development using a variety of techniques including both knowledge-based systems and discrete event simulation. This paper highlights some of the experiences and lessons learned through these research efforts.

In a companion paper to Lehman and Ramil (see aforementioned) titled "Software Process White Box Modelling for FEAST/1", Wernick and Lehman discuss the impact of feedback and feedback control on the evolution of software systems. System dynamics models were constructed to test the hypothesis that the dynamics of real-world software evolution processes lead to a degree of process autonomy. One software process being examined reflects the development of a defense software system. The process examined involves the preparation of successive releases, each subjected to a field trial whose time-scales and objectives are beyond the developers' control. The model has been successfully calibrated and has provided useful results in predicting software system evolutionary trends.

Federal Express' use of simulation to help address the challenges of Year 2000 conversion and other strategic initiatives faced by IT management is presented in "Modeling Fed Ex's IT Division: A System Dynamics Approach to Strategic IT Planning" by Williford and Chang. This paper describes how Fed Ex developed a macro-scale system dynamics model to predict staffing, training, and infrastructure funding over a 5-year period. The paper presents highlights of Fed Ex's models of development and support processes that balance hiring of permanent employees with contract workers. In addition, an infrastructure rollout model is presented that estimates server hardware purchases to support computing workloads as they are migrated from a mainframe environment.

8 Conclusion

8.1 Choosing a Simulation Approach

In some instances, die-hard proponents of a given simulation modeling approach have seemed to argue that theirs is *the* best approach to use and is entirely appropriate for every situation. It is probably true that a model developer who is very skillful with a particular approach and tool can succeed in modeling almost any process situation with that approach and tool – no matter how awkward and unnatural the representation may ultimately be. However, we are convinced that no single modeling approach or tool is the most natural and convenient one to use in all software process situations. In fact, this is one of the principles that we discussed at ProSim'98.

Therefore the best advice we can offer practitioners in selecting a simulation modeling approach is to use one that is well-suited to the particular case at hand, i.e., the purpose, questions, scope, result variables desired, etc. In fact, an important open research direction is to compare alternative simulation modeling approaches for software processes, with the aim of understanding the range of situations where each is particularly suitable, natural, and convenient to apply. Nevertheless, some more detailed guidance (based on current understanding[3]) follows:

- Continuous-time simulations (e.g., system dynamics) tend to be convenient for strategic analyses, initial approximations, long term trends, high-level (global) perspectives, etc. – essentially analyses above the detailed process level.
- Discrete event and state-based simulations tend to be convenient for detailed process analyses and perspectives, resource utilization, queueing, relatively shorter-term analyses, etc.

For example, analysis of unit development costs of a modified inspection process can be modeled in full detail with a discrete event or state-based modeling tool, as in (Raffo, 1996). Alternatively, system dynamics can be used to model the aggregate effect of a modified inspection process, as in (Madachy, 1994) (and should be somehow calibrated to the low-level activities contained in a discrete model).

We now turn to a more specific comparison of the discrete-event and system dynamics simulation approaches. Discrete event models contain distinct (identifiable and potentially differing) entities that move through the process and can have attached attributes. Changes happen in discrete steps. This supports sophisticated, detailed analyses of the process and project performance. System dynamics, on the other hand, models the levels and flows of entities involved in the process, although those entities are not individually traced through the process. Changes happen in a continuous fashion. Both methods can handle stochastic effects. Table 3 compares the two methods, and is derived from (Kocaoglu et al., 1998). As alluded to above, virtually all of the "disadvantages" in Table 3 can be handled by knowledgeable model developers – sometimes elegantly but sometimes with ugly artificial kludges.

	System Dynamics	**Discrete Event**
Advantages	accurately captures the effects of feedback clear representation of the relationships between dynamic variables	CPU efficient since time advances at events attributes allow entities to vary queues and interdependence capture resource constraints
Disadvantages	sequential activities are more difficult to represent no ability to represent entities or attributes	continuously changing variables not modeled accurately no mechanism for states

Table 3: Comparison of System Dynamics and Discrete Event Modeling Techniques

[Table 3 here]

Ideally, a process simulation modeling approach would also support process representation, guidance, and execution capabilities. Unfortunately, many simulation approaches are not very useful for these additional purposes. As an example of a partial solution, however, the Statemate-based modeling approach is well-suited to simulation, representation, and guidance (Kellner, 1991; Kellner, 1989) and it has been shown that such a model can be "compiled" into an executable form suitable for a process-centered environment (Heineman, 1993).

8.2 Hybrid Simulation

Given the preceding arguments, it is apparent that there are cases with sufficient complexity and breadth that no single modeling approach is well suited to all aspects of that situation. A hybrid simulation approach may then be in order[4]. For

[3] The authors gratefully acknowledge discussions with Gregory A. Hansen (CAPI) that contributed to the development of this guidance.

[4] The term "hybrid simulation" was originally used for simulation that involved both analog computing (for continuous aspects) and digital computing (for discrete aspects). However, analog computing for simulation has been largely replaced by continuous

instance, a process may involve important issues that are naturally discrete but others that are inherently continuous. As an example, individual developer's productivity can be nicely modeled as varying over time as a function of experience, morale level, motivation, etc. – i.e., as a continuous variable. For the same process and modeling situation, however, the start of major development projects can most naturally be viewed as discrete events. Thus, an important question that should be answered during modeling is "What aspects of this particular software process are best represented as being continuous and what aspects are best represented as being discrete?" There are some tools that allow both continuous and discrete aspects within a single model, e.g., Extend®[5], but the integration is not as smooth as is desirable. Some work is currently being done to overcome these limitations and formulate solutions (Kocaoglu et al., 1998), however more work is necessary.

In other cases, one may wish to consider both low-level details about aspects of the process, as well as a broad scope such as "long-term product evolution". In such an instance, multi-stage modeling (micro then macro) seems promising. This might involve discrete event or state-based simulations at the detailed level, with their outputs feeding in as parameters to a higher level continuous model covering the desired scope. Just as in software design or estimation, a combined top-down and bottom-up approach is often superior to either approach used alone.

8.3 Recommended Research Issues

Finally, we conclude with a brief list of interesting open research issues in the area of quantitative process model simulation. (Some of these have already been noted earlier in this article, but are repeated here for the sake of collecting them into one place for easy reference.) It is hoped that this list will stimulate future research.

modeling tools for digital computers (e.g., the variety of system dynamics modeling tools available for PCs). Therefore, we extend the meaning of hybrid simulation to cover models that include both continuous and discrete aspects (following (Hansen, 1997)) such as the cases described in this section (8.2). In general, we would use the term hybrid simulation to apply to any combination of fundamentally different modeling approaches.

[5] Extend is a trademark of Imagine That, Inc., San Jose, California. (http://www.imaginethatinc.com)

- Investigate ways to support the integration of representation, guidance, simulation, and execution capabilities for models of software processes. This may include making simulation models more user-friendly to open them up to a wider audience.
- Improve approaches for the validation and calibration of simulation models against limited real-world data.
- Compare alternative simulation approaches with the aim of understanding the range of situations where each is particularly suitable, natural, and convenient to apply.
- Explore and develop techniques for hybrid simulation modeling, and understand when to apply them.
- Develop extensions to better support (re)planning issues under stochastic modeling, e.g., critical path determination, probability of meeting targets, resource allocation.
- Develop generalized process simulation models that may be easily adapted by others. Investigate the feasibility of validating standard "plug and play" model components.

References

Abdel-Hamid, T., and Madnick, S., *Software Project Dynamics*, Prentice-Hall. Englewood Cliffs, New Jersey, 1991.

Akhavi, M., and Wilson, W., Dynamic Simulation of Software Process, in *Proceedings of the 5th Software Engineering Process Group National Meeting* (Held at Costa Mesa, California, April 26-29, 1993), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, (1993). (Presentation in Session Three, Advanced Track.)

Burke, S., Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization, Tech. Rept. CMU/SEI-96-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 1997.

Curtis, B., Kellner, M., and Over, J., Process Modeling, *Communications of the ACM* 35, 9, pp. 75-90 (September 1992).

Gruhn, V., Software Process Simulation on Arbitrary Levels of Abstraction, *Computational Systems Analysis* 1992, pp. 439-444 (1992).

Gruhn, V., Software Process Simulation in MELMAC, *SAMS* 13, pp. 37-57 (1993).

Hansen, G., *Automating Business Process Reengineering: Using the Power of Visual Simulation Strategies to Improve Performance and Profit*, 2nd Ed., Prentice-Hall. Englewood Cliffs, New Jersey, 1997.

Heineman, G., Automatic Translation of Process Modeling Formalisms, Tech. Rept. CUCS-036-93, Department of Computer Science, Columbia University, New York City, New York, November 1993.

Kellner, M., Software Process Modeling: Value and Experience, in *SEI Technical Review*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, pp. 23-54, (1989).

Kellner, M., Software Process Modeling Support for Management Planning and Control, in *Proceedings of the First International Conference on the Software Process* (Held at Redondo Beach, California, October 21-22, 1991), IEEE Computer Society Press, Los Alamitos, California, pp. 8-28 (1991).

Kellner, M., and Raffo, D., Measurement Issues in Quantitative Simulations of Process Models, in *Proceedings of the Workshop on Process Modelling and Empirical Studies of Software Evolution* (Held at Boston, Massachusetts, May 18, 1997), (in conjunction with the 19th International Conference on Software Engineering), pp. 33-37 (1997).

Kocaoglu, D., Martin, R., and Raffo, D., Moving Toward a Unified Continuous and Discrete Event Simulation Model for Software Development, in *Proceedings of the Silver Falls Workshop on Software Process Simulation Modeling (ProSim'98)* (Held at Silver Falls, Oregon, June 22-24, 1998), (1998).

Kusumoto, S., et al., A New Software Project Simulator Based on Generalized Stochastic Petri-net, in *Proceedings of the 19th International Conference on Software Engineering (ICSE-19)* (Held at Boston, Massachusetts, May 17-23, 1997), IEEE Computer Society Press, Los Alamitos, California, pp. 293-302 (1997).

Madachy, R., A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment, Ph.D. Dissertation, Dept. of Industrial and Systems Engineering, University of Southern California, Los Angeles, California, December 1994.

Pall, G., *Quality Process Management*, Prentice-Hall. Englewood Cliffs, New Jersey, 1987.

Paulk, M., et al., Key Practices of the Capability Maturity Model, Version 1.1, Tech. Rept. CMU/SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, February 1993.

Raffo, D., Modeling Software Processes Quantitatively and Assessing the Impact of Potential Process Changes on Process Performance, Ph.D. Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

Raffo, D., Harrison, W., and Keast, L., Coordinating Models and Metrics to Manage Software Projects, Working Paper, School of Business Administration, Portland State University, Portland, Oregon, 1998.

Raffo, D., and Kellner, M., Modeling Software Processes Quantitatively and Evaluating the Performance of Process Alternatives, in *Elements of Software Process Assessment and Improvement*, (K. El Emam and N. Madhavji, eds.), IEEE Computer Society Press, Los Alamitos, California, 1999. (forthcoming)

Tvedt, J., A Modular Model for Predicting the Impact of Process Improvements on Software Development Time, Ph.D. Dissertation, Computer Science and Engineering Dept., Arizona State University, Tempe, Arizona, 1996.

Author Biographies

Dr. Marc I. Kellner is a senior scientist at the Software Engineering Institute (SEI) of Carnegie Mellon University, and has pioneered much of the work on software process modeling and definition conducted at the SEI. He has published more than 30 refereed papers on software process issues, and has delivered approximately 100 technical presentations at numerous conferences world-wide. He has also taught tutorials on process modeling, definition, and related topics to more than 1,100 software professionals. Currently, Kellner leads a team developing exemplary process guides for paper and for the Web, as well as continuing his research and development work in other areas, including quantitative process model simulation.

Prior to joining the SEI in 1986, Kellner was a professor at Carnegie Mellon University, where he established and directed a B.S. degree program in Information Systems. He has also served on the faculty of The University of Texas (Austin), and consulted for several organizations. Kellner received his Ph.D. in Industrial Administration - Systems Sciences (specializing in MIS) from Carnegie Mellon University. He also holds a B.S. in Physics, a B.S. in Mathematics, both with University Honors, an M.S. in Computational Physics, and an M.S. in Systems Sciences, all from Carnegie Mellon University.

Dr. Raymond J. Madachy is an Adjunct Assistant Professor in the Computer Science department at the University of Southern California and the Manager of the Software Engineering Process Group at Litton Guidance and Control Systems. He has published over 35 articles and is currently writing the book “Software Process Modeling with System Dynamics” with Dr. Barry Boehm. He received his Ph.D. in Industrial and Systems Engineering at USC, has an M.S. in Systems Science from the University of California, San Diego, and a B.S. in Mechanical Engineering from the University of Dayton. He is a member of IEEE, ACM, INCOSE, Tau Beta Pi, Pi Tau Sigma, and serves as co-chairman of the LA Software Process Improvement Network (SPIN) steering committee and program chair for the International Forum on COCOMO and Software Cost Modeling.

Dr. David M. Raffo received his Ph.D. in Operations Management and Masters degrees in Manufacturing Engineering and Industrial Administration from Carnegie Mellon University. His current research is in the area of strategic software process management and software process simulation modeling. Raffo has twenty-one refereed publications in the software engineering and management fields. He has received research grants from IBM, Tektronix, the National Science Foundation, the Software Engineering Research Center (SERC), and Northrop-Grumman. Prior professional experience includes managing software development and consulting projects at Arthur D. Little, Inc., where he received the company's Presidential Award for outstanding performance. Currently, Dr. Raffo is an Assistant Professor of Operations Management and Information Systems in the School of Business Administration at Portland State University. He is Co-Director of Portland State University’s Center for Software Process Improvement and Modeling.