Empirical analysis in software process simulation modeling

David M. Račo a,*, Marc I. Kellner b,1

a School of Business Administration, Portland State University, P.O. Box 751, Portland, OR 97207-0751, USA
b Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Received 15 July 1999; received in revised form 2 November 1999; accepted 3 November 1999

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 paper discusses some of the important empirical issues that arise in software process simulation modeling. We first address issues concerning real-world data used to (1) establish input parameters to a software process simulation model, and (2) establish actual organizational results against which the model’s results (i.e., outputs) will be compared. On the input side, the challenges include small sample sizes, considerable variability and outliers, lack of desired data, loosely defined metrics, and so forth. On the output side, the paper addresses (1) verification and validation of the model, and (2) quantitative approaches to evaluating model outputs in support of managerial decision making including financial performance using Net Present Value (NPV), multi-criteria utility functions, and Data Envelopment Analysis (DEA). The paper focuses on the stochastic modeling using Monte Carlo simulation. The paper is grounded in the authors’ practical application experiences, and major points are illuminated by examples drawn from that field work. © 2000 Elsevier Science Inc. All rights reserved.

Keywords: Process modeling; Return on investments; Empirical software engineering; Data Envelopment Analysis

1. Introduction and background

Significant progress is being made in the area of quantitative simulation modeling of software engineering processes. 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. Recent work has been highlighted in the First International Workshop on Software Process Simulation Modeling (ProSim’98) and in a set of articles from ProSim’98 appearing in a special issue of the Journal of Systems and Software (vol. 46, no. 2/3 (15 April 1999), pp. 89–216). An overview of the field appears in Kellner et al. (1999). An overview of these authors’ research contributions to this field appears in Kellner and Račo (1997) and Račo and Kellner (1999).

Kellner’s work (Kellner, 1991) was the first state-based simulation applied to software engineering processes and has stimulated further work by Račo and others. The work of Abdel-Hamid and Madnick (1991) publicized their use of the System Dynamics (SD) paradigm (continuous simulation) to represent software development processes, particularly at the project level. It has also stimulated considerable interest in the use of SD in general, at a more detailed process level (Madachy, 1994; Tvedt, 1995), at a broader organizational level (Williford and Chang, 1999), and to develop training tools for development managers (Drappa and Ludewig, 1999).

There are a wide variety of reasons for undertaking simulations of software process models, including

• strategic management,
• planning,
• control and operational management,
• process improvement and technology adoption,
• understanding, and
• 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 (Kellner et al., 1999).

This paper discusses some of the important empirical issues that arise in software process simulation modeling. In Section 2, issues related to input parameters for process simulation models are discussed. Section 3 discusses the empirical evaluation of stochastic process simulation model outputs using Monte Carlo simulation. A key concept which is highlighted throughout is that: the required level of precision and formality with which model inputs (and outputs) should be analyzed (and validated) should be driven by the questions the model has been designed to address. Major points are illuminated by examples drawn from the authors’ field work using stochastic modeling and Monte Carlo simulation.

2. Empirical evaluation of input to process simulation models

2.1. Considerations when handling input distributions to process simulations

Development of input parameters for process simulation models mainly requires determining representative distributions for key project parameters such as productivity, task effort, defect injection rates, etc. It also requires building meaningful relationships among key parameters that can be encoded into a model (e.g., the relationship between effort and schedule). Obviously, model results can be only as accurate and as timely as the inputs support. Appropriate and insightful collection and analysis of project data are the basis for developing input parameter values and relationships. These are in turn essential for accurate model representation and output.

Rigorous statistical analyses of project and process data are important and can be used. Rigorous analyses naturally impose certain required characteristics on the quality and quantity of the data. However, we contend that the uses of the model and limitations of the data must be kept in mind throughout in order to guide the analyses to be conducted.

Analysis of project and process data does not need to be overly sophisticated or complex. We propose that depending upon the intended use of the data and the model, the level of formality and the extent of the statistical analyses can be adjusted. For models used to make decisions such as setting fixed contract bid prices, the level of accuracy and formality/rigor is extremely important. However, such a high standard of data accuracy and statistical rigor is not needed for models that predict relative differences – as are needed to support go/no go decisions on process changes.

The variability associated with real-world software development operations often causes process data to contain outliers, which can perturb the results of statistical analyses. The margin of error associated with the data collection process must also be considered as well as the quantity of representative data points that are available. We recommend that the following points be considered in determining how to handle outliers in the data (also see Kitchenham, 1998 regarding methods for handling outliers in the context of conducting ANOVA analyses):

1. Distinguish the purpose of the model. For example, is the goal to predict typical behavior or exceptional behavior? (This directs which data to focus on and which data might be excluded from the analysis.)
2. Attempt to understand the conditions contributing to outlying data. These conditions might include
   • a complex function or set of modules;
   • special technical problems or other rare occurrences that are not considered to be important for planning purposes;
   • junior developers;
   • the project manager is lax about data collection; and so forth.
3. Verify questionable data with internal process experts and/or externally published data sources where appropriate (e.g., defect detection rates for inspections, etc.).

Given an understanding of the above, judgements can be made as to whether including outlying data points would be most important for developing the model, or whether extracting the central tendencies of the distribution is most important. Given the flexibility of simulation models in general, a number of input distributions can be checked for their effect on simulation model results.

2.2. An example

As an example, we present a set of representative data taken from a leading software development firm. Fig. 1 contains a histogram showing the range of inspection efficiencies (i.e., the fraction of existing defects that were detected during code inspections) on a large-scale software development project. The location of each bar along the X-axis represents the inspection efficiency level. The height of each bar represents the number of inspections falling into each efficiency category. As can be seen in Fig. 1, the inspection efficiencies range from 1–55%. The median efficiency is 11%, a mode of 6%, a mean of 14.48% and a standard deviation of 13.34%.

Deciding what can be done with data such as this is difficult. Although there are apparently a large number of observations (91 in this data set), there is a high degree of variability (the ratio of the standard deviation
to the mean or the coefficient of variation is 0.921). As is common, there was no additional information provided with this data set. Clearly, a first step would be to attempt to assign causes to the variation and stratify the sample according to parameters such as code complexity (or other artifact based qualifiers), experience of the personnel involved (or other skill based qualifiers), differences in the development processes used, and so forth.

We can see that if we would be trying to predict the performance of future inspections in order to develop cost estimates for a bid proposal, the amount of variability in the sample would preclude precise predictions. Rigorous analyses would be needed to achieve the added precision (which may or may not be possible to obtain) (see Good, 1994 and Edgington, 1995 for ideas on ways to work with small samples). However, if we are trying to determine the relative impact on project performance if inspections would be improved, the need for understanding the causality of the variation is reduced. We can instead work to verify the data and develop probable scenarios to be used during sensitivity analysis using the simulation model.

For this company, we noted the significant number of poor inspections and investigated their cause. Second, we observed the high inspection efficiencies on eight inspections and investigated their cause. For the poor inspections we found that schedule pressure, lack of rewards, and management discouragement conspired to create a situation where the quality of the inspection was based upon the discretion of individual inspectors. Data collection problems were also uncovered, but the low efficiency scores presented in Fig. 1 were verified as being reasonable approximations of the true value.

In verifying the high inspection efficiencies, we found that one project manager, despite schedule pressures and lack of rewards, chose to maintain a rigorous inspection process for his project team. In fact, this team’s set of modules were above average in complexity and finished testing ahead of schedule.

Using a process simulation model, we provided management with feedback on the overall project performance when code inspection efficiency was poor (as was typical). We also showed the potential project performance that could be achieved, based on the higher efficiency inspections that had been accomplished within the company itself.

2.3. Handling real-world data problems

In the “real world” we often find that the desired measurements do not exist, and that even when they do exist the metrics may be only roughly defined. In such cases, the modeler is forced to make due with what is available. The following strategies can be useful in coping with this reality (Kellner and Raffo, 1997).

When metrics are poorly defined or uncertain, personnel who have worked in or managed the process can help clarify what was actually measured. Unfortunately, in some of these instances exactly what was measured varied from one measurement to another. The coping strategies described in the next few paragraphs may be helpful in such instances.

Situation 1: In some cases, measurements were taken of the desired metric, but the accuracy of the measurements is questionable. For instance, in many firms, effort records can lack accuracy or the desired granularity for modeling purposes.

In such cases:
- **Work with experienced personnel to select representative and accurate data.** If metrics were collected for a certain parameter, often a great deal of data will be available. Different project teams will have had a higher or lower enthusiasm for data collection efforts and the data can reflect this. Experienced personnel can be quite knowledgeable regarding the potential integrity of various data. It is often useful to work with them to “scrub” the data set that will be used for the model.

Situation 2: In some cases, a measurement was taken that is similar to but not quite the same as the desired metric.

In such cases:
- **Make a calculated adjustment.** Sometimes a calculated adjustment can be made to the measured value to approximate the desired metric. As a simple example, suppose cost was desired but effort was collected; the manager or personnel office can often provide an average burdened pay rate that can be multiplied by effort to approximate cost.
- **Work with experienced staff.** The experience and insight of process participants (workers and managers) can sometimes be applied to adjust what was actually measured to become an estimate of the desired mea-
mination. For instance, perhaps project duration was measured, but these duration measurements did not include project inception activities. The staff may know from undocumented experience that project inception typically takes four weeks in their organization, or that it is usually about 5% of the planned schedule.

- **Go to the source documents.** Sometimes summarized project or accounting data do not capture the desired granularity. However, the metrics of interest were collected in the source documents that were completed as part of the process. For example, management was only concerned with the gross number of defects detected by phase during inspections and this was all that was provided in project management reports. However, information describing where these defects were injected was not contained on the raw inspection forms. Reviewing these forms directly enabled us to calculate inspection efficiency and other important parameters which were later used in the model.

- **Adjust decision variables to capitalize on available data.** Management will sometimes be willing to modify their decision variables somewhat in order to cope with the realities of what has actually been measured and recorded in the past. For example, they may prefer to have cost as a decision variable but can accept effort instead.

- **Adjust model scope.** In some cases, the difference between desired and actual metrics is not fundamentally relevant to the comparison between the AS-IS and TO-BE processes. For instance, suppose the full process duration and effort (from project inception through delivery) is desired, information is only available for work following high-level design. If the process change is one that occurs later in the process (say during coding), and has no impact on earlier process activities, the models can simply begin after completion of high-level design. There simply would not be any effect on the difference between the AS-IS and TO-BE models by leaving out the earlier steps.

**Situation 3:** In other cases, neither the desired metric nor a suitable proxy has been measured.

In such a case:

- **Reconstruct the metric.** Sometimes the desired measurements can be (re)constructed from existing information. As a simple example, the actual number of builds may be desired, and can be counted by examining files in the configuration management system.

- **Look for data in other parts of the organization.** The desired measurement (or something close) may be available elsewhere in the organization. For example, in some organizations initial development is separated organizationally from field maintenance and support of a system. While the counts of defects delivered from development might not be available in the development organization, they are very probably available in the support organization.

- **Estimate the desired data using expert opinion.** Process participants may be able to estimate the desired measurements.

- **Adjust scope of model parameters.** Again, in some cases, the desired metric is not fundamentally relevant to the comparison between the AS-IS and TO-BE processes. For instance, suppose the error injection rates during requirements analysis and high-level design are desired, but are totally unavailable. If the process change is one that occurs later in the process (say during coding), and has no impact on earlier process activities, the models can simply begin after completion of high-level design. There simply would not be any effect on the difference between the AS-IS and TO-BE models by leaving out the earlier steps.

- **Get data from the literature.** In some cases, it may be acceptable to utilize typical figures taken from the literature. For example, an organization might not know or be able to (re)construct its defect detection efficiency in unit test, but a typical figure or range can be located in the open literature.

- **Drop the variable from the model.** Sometimes all that can be done is to eliminate the desired variable from the model. If no reasonable basis can be found to establish values for a desired input parameter, there really is no choice but to drop it. If the variable in question is a desired output of the simulation model, it can be retained; but if no reasonable basis can be found to establish values for the desired variable, it will not be possible to quantitatively validate the model for that output variable.

An alternative approach that some modelers have used in order to get more accurate data is to focus on using more aggregated measures of project costs and schedule (as used by cost estimation models). However, this approach is not desirable when predicting aspects related to specific process steps because an artificial allocation of costs and so forth needs to be made.

While these strategies for coping with measurement problems encountered in the real world are not claimed to be exhaustive, they provide a good foundation for dealing with these modeling and metrics issues.

In summary, empirical analysis of project and process data is crucial to the task of developing values and distributions for the inputs that drive a process simulation model. Several techniques are available for coping with the data and measurement issues encountered in practice. Clearly, understanding the intended use of the model can provide the necessary flexibility to derive useful insights from a data set that might otherwise seem problematic.

### 3. Empirical analysis of output from process simulation models

Intelligent interpretation of process simulation model output that is cognizant of management’s needs and the
limitations of the data is key to a successful modeling project. Often software developers are unaware of how metrics data can be used or how to interpret these values. Managers often have a desire to simplify by focusing on one or two output results. Without proper education, the full range of information that simulation models can provide can be misinterpreted.

The issue of understanding model results is exacerbated when the model does not produce simple, deterministic, point estimates. However, because of the high degree of variability in the performance of software work, and the large number of influencing factors (some of which are probably not known or represented in the model), we believe that stochastic modeling with Monte Carlo simulations is the most appropriate approach to apply to process simulation modeling.

Statistical and empirical analyses to test the properties of output distributions are clearly appropriate and useful. At the same time, however, understanding the implications of the simulation results and how they relate to a company’s business situation is equally if not more important. The following sub-sections address the topics of (Section 3.1) model validation, and (Section 3.2) analysis of the model’s outputs. The discussion reflects both the technically appropriate statistical and empirical analyses, as well as a more pragmatic perspective oriented toward real-world value.

3.1. Validation of the model

Several kinds of verification and validation activities are appropriate for software process simulation models:

1. **Face validity of graphical models.** Does the graphical model accurately reflect the real process? Examining face validity is typically the first step in validating a process model. It is qualitative in nature and entails having process experts review and comment upon the structure and details of the graphical model to see if there are any discrepancies between the model and reality. Depending upon the level of detail contained in the graphical model, this step will provide more or less value. The graphical process modeling approach developed by Kellner et al., using Statemate®-based models and three integrated modeling perspectives (Kellner, 1989; Kellner, 1991), is very detailed and reviewing these models with process experts has proven to be a very effective means for facilitating communication and consensus among process experts (Raffo, 1996).

2. **Verification of model inputs.** Were the model inputs calculated correctly and in a manner that is consistent with the purpose of the model? This step entails checking the input parameters used by the model and verifying that they were defined and calculated correctly. Decisions made about the model purpose and the resulting implications on the analysis of the input data should be revisited and confirmed.

3. **Verification of model outputs.** Given a fixed set of inputs, does the model calculate the outputs correctly?

4. **Qualitative assessment of reasonableness of model outputs.** Given that the model calculates the outputs correctly, do the numbers seem appropriate?

5. **Predicting quantitative results of input data sets.** Is the model able to reasonably predict real outputs for a given set of inputs?

6. **Special case validity.** Can the model predict abnormal behavior correctly?

7. **Post-mortem validation.** Were model predictions accurate? Going forward, if the model is given a new set of real data, how well does it predict actually observed results?

In general, it is difficult to thoroughly validate software process simulation models. The reasons for this difficulty include the following:

1. **Length of time and cost.** In order to thoroughly validate a process simulation model, one would have to wait until the end of the project, collect a new set of performance data and compare that to the predicted output values and estimated input values. This additional step is usually time consuming and presents many challenges. Although companies want models they can trust, they are rarely willing to fund these activities. Moreover, stochastic modeling requires multiple observations (i.e., data from multiple projects) to accomplish thorough and rigorous validation.

2. **Available data.** In many situations, the data desired for developing model inputs are not available. Reasonable replacements/surrogates need to be found; these pragmatic measurement issues are addressed in Section 2. Although the target software project may start to collect these desired data, quite often data collection lapses, and therefore values cannot be updated anyway.

Given the limitations mentioned above, the most practical standards for validating a model are the standards of “reasonableness” and “improvement”. Is the model reasonable with respect to model inputs, calculations, and outputs compared to management/expert experience and opinion? This question implies V&V activities 1–5 (above). Does the estimate provided by the model provide additional accuracy, insight, or clarity over the current approach used? For pragmatic managers, the real test of the model is whether it provides value to the decision-making process. For pragmatic researchers, the real test is whether they can show management that their model adds value.

The intended purpose of the model drives the level of validation that is necessary. For a model that will be used to predict exact performance measure values (like COCOMO) a full and thorough validation process is necessary. For a model that is required to predict rela-
tive values in order to make a yes/no or go/no-go decision, the validation process can be less rigorous. For example, when predicting the impact of a process change and comparing the performance of the process before and after the process change is implemented, rigorously validating the model’s ability to predict actual process performance for V&V activities 6 and 7 may not be required. The reason is that the underlying process performance is controlled and held constant so the net impact of the proposed process change can be observed. For a decision comparing relative performances of process alternatives, rigorously validating model inputs pertaining to the process change, as well as conducting the other V&V activities 1, 3, 4 and 5, are probably the most important aspects of the validation.

3.2. Analysis of simulation outputs

Once a model has been suitably validated and output has been generated, the output needs to be analyzed in order to achieve the purpose of the simulation modeling effort (often, to support making a decision).

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, parametric statistical tests can be used to analyze the results. Otherwise non-parametric tests that do not assume the data be normally distributed may be required. The CHI Square tests (Kolomogorov-Smirnov) and probability plots are good tests for normality. For good text book references, see DeGroot (1989) and Pelosi and Sandifer (2000). For a discussion of the application to software process simulation model output, see Raffo (1996).

Also, the correlation among process performance measures should be tested so that no surprises are encountered. Unlike 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 simulation model results from a statistical perspective, other analyses are used to manipulate the data to support the 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. In addition, we have used the following four approaches to evaluate alternatives:

2. Comparison of performance measure deltas using a utility function.
3. Comparison of performance measure deltas using financial measures such as Net Present Value (NPV) or Return on Investment (ROI).
4. Comparison of overall performance measure values using Data Envelopment Analysis (DEA).

These approaches are described below along with a numerical example evaluating the relative merits of five alternative process configurations. Table 1 shows project level performance measure output for five possible inspection configurations for a waterfall process. The

<table>
<thead>
<tr>
<th>CONFIG</th>
<th>REM DEF MEAN</th>
<th>LC EFF MEAN</th>
<th>DUR MEAN</th>
<th>REM DEF STDev</th>
<th>LC EFF STDev</th>
<th>DUR STDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWNN</td>
<td>13.4</td>
<td>8274.7</td>
<td>2849.9</td>
<td>4.92</td>
<td>366.68</td>
<td>546.45</td>
</tr>
<tr>
<td>FFNN</td>
<td>12.6</td>
<td>8452.6</td>
<td>2762.2</td>
<td>4.53</td>
<td>340.88</td>
<td>469.28</td>
</tr>
<tr>
<td>WWNW</td>
<td>9.1</td>
<td>7807.0</td>
<td>2387.7</td>
<td>3.75</td>
<td>277.56</td>
<td>416.30</td>
</tr>
<tr>
<td>WWWW</td>
<td>6.6</td>
<td>7560.3</td>
<td>2055.7</td>
<td>2.76</td>
<td>238.10</td>
<td>326.17</td>
</tr>
<tr>
<td>FFFF</td>
<td>3.3</td>
<td>7615.9</td>
<td>1776.9</td>
<td>1.87</td>
<td>161.01</td>
<td>171.73</td>
</tr>
</tbody>
</table>

| \( \hat{a} \) | Indicates the type of inspections for each process configuration, where F = full Fagan inspection, W = modified walk through, and N = none. The type of inspection is indicated for each pre-test development phase: functional specification, high-level design, low-level design, and code, respectively. |
| \( \hat{b} \) | Indicates the expected number of remaining major defects when the product is released. |
| \( \hat{c} \) | Denotes the expected life cycle effort (from functional specification to release in hours) for the project. |
| \( \hat{d} \) | Denotes the expected life cycle project duration for the given configuration (in work hours, where 40 work hours = 1 week). |
| \( \hat{e} \) | Indicates the standard deviation associated with the distribution for remaining major defects as predicted by the simulation model for the given configuration (integer valued – approximately normally distributed). |
| \( \hat{f} \) | The standard deviation associated with the distribution for life cycle effort as predicted by the simulation model for the given configuration (normally distributed). |
| \( \hat{g} \) | Indicates the standard deviation associated with the distribution for project duration as predicted by the simulation model for the given configuration (normally distributed). |
simulation model used to generate the outputs depicts an actual modified waterfall process used at a leading software development firm. The main development phases are: functional specification, high-level design, low-level design, and coding. Several testing processes follow development. They include unit test, functional test, and system test. The alternative process configurations listed in Table 1 refer to the type of inspection being conducted after each development stage. All configurations use the same modified waterfall development process as a base. The inspection choices are: “F” representing a traditional Fagan inspection, “W” representing a walk through, and “N” representing no inspection. Each inspection type has various parameters associated with it such as defect detection rates, number of inspectors involved, productivity rates for preparing the inspection material and so forth. These were obtained from estimates made by experienced software engineers as well as company records. We assume that the baseline configuration that the company is using is “WWNN” indicating that there is a walkthrough after functional specification development, a walkthrough after high level design, and no inspections after that. The next defect detection step is unit test.

The company is interested in reviewing its inspection policy. They are wondering what would be the value if instead of doing walkthroughs, they did full Fagan inspections after functional specification development and after high-level design. This is indicated by the FFNN configuration. They are also interested in determining what the benefit would be if they did walk throughs after code development or if there would be an increased net benefit of conducting walk throughs after each phase of the development process. These configurations are represented by WWNW and WWWW, respectively. Finally, the company is curious about the potential benefit of implementing a full suite of Fagan inspections along their development process (the FFFF configuration).

To examine these alternatives, a state-based stochastic model using Monte Carlo simulation was developed. The model captured the full development life cycle at the field study firm (from functional specification through release and field service). Predictive models were developed using actual project data from past releases of the product. Linear regression and Task Element Decomposition (Raffo, 1996) were the main techniques used to develop the overall predictive simulation model which integrated models for development cost (effort), product quality (defects) and time to market (task duration). The purpose of this paper is to show possible analyses that can be done given a simulation model and output. As a result, further details regarding the process configurations, the simulation model and the model parameters will not be discussed here. The interested reader is referred to Raffo (1996).

Referring back to Table 1, the second column “REM DEF MEAN” shows the expected number of remaining defects for each inspection configuration. The third column “LC EFF MEAN” shows the expected number of staff hours to complete the life cycle process (the life cycle effort also includes expected implementation effort (costs) associated with changing the process configuration from the WWNN baseline to the proposed alternatives. These implementation costs include developing and delivering new process documentation and training, as well as additional follow-up and mentoring). The fourth column “DUR MEAN” shows the expected duration of the project. This is provided in work hours and we assume that there are 160 work hours per month. Columns 5–7 in Table 1 show the standard deviations of remaining defects, life cycle effort, and project duration, respectively.

In order to determine which configuration is preferred, we need to evaluate the above alternatives. Since there are multiple measures of performance, a tradeoff may need to be made of one performance measure over another. We present several methods of conducting this tradeoff above. Tables 2–5 show the results of these tradeoffs.


When running the simulation model, the results of alternative process configurations with the baseline process are obtained and compared (Table 2). The differences between the performance measures are then calculated and checked for statistical significance using t-tests (Table 3). If all the performance measures of interest (e.g., life cycle cost, product quality, and project schedule) are improved and the differences are statistically significant, then we have confidence that an alternative process will offer an improvement. A base case analysis as well as various sensitivity analyses are also conducted. A discussion of sensitivity analysis is beyond the scope of this paper, but see Raffo (1996) and Raffo and Kellner (1999) for details.

In Table 2, we can see the expected values for the performance measures of quality (REM DEF MEAN), cost (LC EFF MEAN), and schedule (DUR MEAN) as well as the respective standard deviations. These values show the expected reduction (savings) associated with each performance measure for each process alternative compared to the WWNN baseline. Hence, for the FFFF, WWWW, and WNNW alternatives, we see a noticeable improvement in all the performance measures compared to the baseline WWNN process. For the FFNN configuration, however, we see that although remaining defects (quality) and duration (schedule) are expected to modestly improve, we expect that the life cycle effort (cost) is expected to get worse (increase). As a result, the FFNN configuration presents a mixed result which would need to be traded off. Table 3 shows the t-statistics associated with the differences presented.
Table 2
Mean deltas with standard deviation results

<table>
<thead>
<tr>
<th>Rank</th>
<th>CONFIG</th>
<th>RD Delta</th>
<th>LC EFF Delta</th>
<th>DUR Delta</th>
<th>RD Delta</th>
<th>LC EFF Delta</th>
<th>DUR Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FFFF</td>
<td>10.18</td>
<td>658.87</td>
<td>1073.05</td>
<td>3.81</td>
<td>310.08</td>
<td>475.65</td>
</tr>
<tr>
<td>2</td>
<td>WWWWW</td>
<td>6.82</td>
<td>714.48</td>
<td>794.24</td>
<td>2.48</td>
<td>177.03</td>
<td>273.34</td>
</tr>
<tr>
<td>3</td>
<td>WNNW</td>
<td>4.34</td>
<td>467.84</td>
<td>462.21</td>
<td>1.48</td>
<td>135.26</td>
<td>209.30</td>
</tr>
<tr>
<td>4</td>
<td>FFNN</td>
<td>0.80</td>
<td>-177.80</td>
<td>87.69</td>
<td>0.76</td>
<td>87.40</td>
<td>159.14</td>
</tr>
<tr>
<td>5</td>
<td>WNNN</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

*Expected difference between the given configuration and the baseline WWNN configuration for the remaining defects performance measure. A positive value indicates an improvement (i.e. a reduction for this variable).

Table 3
Expected differences in performance measures with t-statistic results

<table>
<thead>
<tr>
<th>Rank</th>
<th>CONFIG</th>
<th>RD Delta</th>
<th>LC EFF Delta</th>
<th>DUR Delta</th>
<th>RD Delta</th>
<th>LC EFF Delta</th>
<th>DUR Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FFFF</td>
<td>10.18</td>
<td>658.87</td>
<td>1073.05</td>
<td>3.81</td>
<td>310.08</td>
<td>475.65</td>
</tr>
<tr>
<td>2</td>
<td>WWWWW</td>
<td>6.82</td>
<td>714.48</td>
<td>794.24</td>
<td>2.48</td>
<td>177.03</td>
<td>273.34</td>
</tr>
<tr>
<td>3</td>
<td>WNNW</td>
<td>4.34</td>
<td>467.84</td>
<td>462.21</td>
<td>1.48</td>
<td>135.26</td>
<td>209.30</td>
</tr>
<tr>
<td>4</td>
<td>FFNN</td>
<td>0.80</td>
<td>-177.80</td>
<td>87.69</td>
<td>0.76</td>
<td>87.40</td>
<td>159.14</td>
</tr>
<tr>
<td>5</td>
<td>WNNN</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

*See Table 2 for definitions.

Table 4
Utility function results

<table>
<thead>
<tr>
<th>Rank</th>
<th>CONFIG</th>
<th>REM DEF Mean</th>
<th>LC EFF Mean</th>
<th>DUR Mean</th>
<th>Util Fcn Mean</th>
<th>Util Fcn STDev</th>
<th>PR (U &lt; 0) %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FFFF</td>
<td>3.3</td>
<td>7615.9</td>
<td>1776.9</td>
<td>3293.60</td>
<td>1342.58</td>
<td>0.71</td>
</tr>
<tr>
<td>2</td>
<td>WWWWW</td>
<td>6.6</td>
<td>7560.3</td>
<td>2055.7</td>
<td>2643.87</td>
<td>723.25</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>WNNW</td>
<td>9.1</td>
<td>7807.0</td>
<td>2387.7</td>
<td>1622.28</td>
<td>500.07</td>
<td>0.09</td>
</tr>
<tr>
<td>4</td>
<td>FFNN</td>
<td>12.6</td>
<td>8452.6</td>
<td>2762.2</td>
<td>242.23</td>
<td>609.44</td>
<td>34.46</td>
</tr>
<tr>
<td>5</td>
<td>WNNN</td>
<td>13.4</td>
<td>8274.7</td>
<td>2849.9</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

*See Table 2 for definitions.

Table 5
NPV results

<table>
<thead>
<tr>
<th>Rank</th>
<th>CONFIG</th>
<th>REM DEF Mean</th>
<th>LC EFF Mean</th>
<th>DUR Mean</th>
<th>NPV (15%) Mean</th>
<th>NPV (15%) STDev</th>
<th>PR (NPV &lt; 0) %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FFFF</td>
<td>3.3</td>
<td>7615.9</td>
<td>1776.9</td>
<td>$362,291.35</td>
<td>$118,344.45</td>
<td>0.11</td>
</tr>
<tr>
<td>2</td>
<td>WWWWW</td>
<td>6.6</td>
<td>7560.3</td>
<td>2055.7</td>
<td>$253,041.92</td>
<td>$68,513.12</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>WNNW</td>
<td>9.1</td>
<td>7807.0</td>
<td>2387.7</td>
<td>$157,874.18</td>
<td>$44,518.84</td>
<td>0.09</td>
</tr>
<tr>
<td>4</td>
<td>FFNN</td>
<td>12.6</td>
<td>8452.6</td>
<td>2762.2</td>
<td>$271,836.80</td>
<td>$26,910.00</td>
<td>15.15</td>
</tr>
<tr>
<td>5</td>
<td>WNNN</td>
<td>13.4</td>
<td>8274.7</td>
<td>2849.9</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

*See Table 2 for definitions.
in Table 2. All t-statistics values have an absolute value considerably greater than 2.0 (which is required at the \( P = 0.05 \) level given the number of data points) and are statistically significant.

2. Comparison of performance measure deltas using a utility function. Again, the simulation model is run, obtaining the results for the AS-IS and TO-BE processes (or for various process alternatives as in this example), and the differences between the performance measures are obtained. If these differences show conflicting results where some performance measures worsen and others improve (as with the FFNN alternative), 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. Again this overall measure can be checked for statistical significance and a decision can be taken. The reader is referred to Raffo (1996) and Raffo and Kellner (1999) for details and examples.

Table 4 shows the utility function values for the four alternative process configurations being considered. The utility function captured management’s preferences when trading off among the three performance measures of cost, quality, and schedule. The weighting scheme used for this utility function is provided in Raffo (1996). To compute the values in Table 4, a linear utility function was developed based upon interviews with project managers at the field study site. As mentioned above, the FFFF, WWWW, and WNNW configurations were clearly preferred to the baseline WNNN configuration. Moreover, given the way management traded off between the three performance measures, even the FFNN configuration would be preferred for this project since the utility function value is greater than 0 and is statistically significant (t-statistic = 2.81, needed to be greater than 2.0).

3. Comparison of performance measure deltas using financial measures such as NPV. This approach is very similar to the approach using a utility function. NPV 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. Table 5 shows the expected NPV for each configuration. In this example, staff hours of effort were valued at $125 per hour. Remaining defects were estimated by the company to cost 150 hours per defect to repair. Life cycle effort differences were assumed to be incurred at the end of year 1 and field repair costs were assumed to be incurred at the end of year 2. Again, all the configurations show an improvement over the baseline WNNN configuration which is indicated by their positive expected NPVs.

4. Comparison of overall performance measure values using DEA (Charnes et al., 1994). DEA may be viewed as an optimization technique which finds an “efficient 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 the given configuration has one of the best (shortest) schedules, but more defects and higher costs than most of the other configurations. Hence, the given configuration would want 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 the given configuration is the best using this objective function, it is held as a candidate for the optimal set. If another configuration beats the given configuration using the given configuration’s objective function, the DEA program knows that the given configuration is sub-optimal and discards it. The DEA program evaluates all the configurations in this manner and determines an efficient frontier from which final selections can be made.

Some examples of applying DEA to software projects include Banker and Kemerer (1989) as well as Banker et al. (1987, 1991, 1994). These DEA models evaluate factors impacting the productivity of software maintenance projects as well as new development but are not used to evaluate simulation output data.

Table 6 shows the DEA outputs associated with the expected performance measures of each of the configurations. As can be seen the WWWW and FFFF alternatives are both on the productive frontier as is indicated by their overall efficiency being equal to 1 or 100%. The efficiency score indicates the relative efficiency of each configuration. An efficiency score of 1 (or 100%) indicates that the configuration is potentially one of the most efficient configurations and a member of the efficiency frontier. An efficiency score of less than 100% indicates a configuration’s relative distance from the efficiency frontier or convex linear hull in a linear programming sense. We can see that using DEA, the FFNN configuration performed worse than the baseline WNNN configuration by having a lower overall efficiency. The poor effort score combined with the modest gains on the other performance measures proves to be a less efficient combination.

Both the magnitude of the difference in the performance measures between the two comparisons as well as their relative weights, provide the possibility of having a rank reversal problem (Schoner and Wedley, 1989; Satay and Vargas, 1987). In the examples shown above, clearly the FFFF, WWWW and WNNW alternatives appear to be superior to the baseline WNNN process. Moreover, given the magnitude of the improvements in
the performance measures (as indicated by the \(t\)-statistics), there is very little chance of the rank reversal. However, the choice as to whether the FFNN or the WWNN process is preferred depends heavily upon how the three performance measures would be traded off as well as the modest magnitude of the improvements. Each of the methods helps provide indications as to when the ranking will be close. At these times the rank reversal probabilities can be determined.

It is interesting to note that the different evaluation techniques above show slightly different rankings. In general, each method has its own approach to ranking process alternatives and either implicitly or explicitly makes value tradeoffs among the performance measures. The best approach depends upon which method most closely matches the decision maker’s true preferences. At the same time, some decisions makers may choose to use multiple ranking methods since (1) having multiple methods return similar rankings can provide additional confidence in the rankings and (2) differences in rankings can quickly identify areas which may merit further exploration.

4. Summary

Software process simulation models can be very effective in providing a framework and focus for metrics collection programs. The results and information provided by these models can be highly 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, and the real-world data they reflect, are important. Sophisticated (but often complex) techniques may help. More important, however, is to understand the purpose and intended use of the model, and to then filter input data and interpret output data accordingly. By making appropriate filtering decisions, greater opportunities to utilize existing data become possible. By making informed and appropriate interpretations of process simulation model results, which are cognizant of the limitations of the data, useful insights can be gained to support project decisions.

In this paper, we explored a variety of statistical and analytical techniques that could be employed to evaluate various process alternatives. We have also shown examples of how utility functions, financial measures (such as NPV) and Data Envelopment Analysis (DEA) 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.

References


David M. Raffo received his PhD degree from Carnegie Mellon University. His current research is in the area of strategic software process management and design. His dissertation work developed a theoretical framework and quantitative techniques for predicting the performance of software development projects. The concepts and theories have been field-tested at leading software development firms. He has more than 35 refereed papers in the software engineering and management. Dr. Raffo has received research grants from the National Science Foundation, NASA, Northrop Grumman Corporation, IBM Corporation, Tektronix, and the Software Engineering Research Center (SERC). 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 teaches courses in Software Process Management, Software Process Modeling, Total Quality Management (TQM) and Operations Management.

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 approximately 1,800 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.