Simulation: An Enabling Technology in Software Engineering

Alan M. Christie

Abstract

This paper suggests that the software engineering community could exploit simulation to much greater advantage. There are several reasons for this. First, the Office of the Secretary of Defense has indicated that simulation will play a significant role in the acquisition of defense-related systems to cut costs, improve reliability and bring systems into operation more rapidly. Second, there many areas where simulation can be applied to support software development and acquisition. Such areas include requirements specification, process improvement, architecture trade-off analysis and product line practices. Third, commercial simulation technology, capable of supporting software development needs is now mature, is easy to use, is of low cost and is readily available.

What's so great about simulation?

This paper aims to raise awareness about the usefulness and importance of simulation in support of software engineering. Simulation can be applied in many critical areas and enables one to address issues before they these issues become problems. Simulation is more than just a technology, as it forces one think in global terms, about system behavior, and about the fact that systems are more than the sum of their components. Simulation can provide insights into the designs of, for example, processes, architectures, or product lines before significant time and cost has been invested, and can be of great benefit in support of training. Simulation is being increasingly emphasized in the DoD community, where there is documented evidence that its impact on costs, quality and schedule is non-trivial. I believe that the software engineering community needs to take a stronger role in exploiting the technology [18].

The DoD emphasis on simulation

The Office of the Secretary of Defense has recently initiated an effort focussed on the use of modeling and simulation to support improvement of the acquisition process. Jacques Gansler, Under Secretary of Defense (Acquisition and Technology) states that *a directive which I issued this year* (1998) *requires the integration of modeling and simulation in our acquisition process -- across functional disciplines -- and throughout the life-cycle of systems. We are committed to reforming the acquisition system and recognize that an essential tool for accomplishing that reform will be modeling and simulation* [3]. While he does not explicitly include the software acquisition process, there is no reason to doubt (see the end of this section) that software acquisition can benefit as much as any other DoD acquisition area.

Gansler's remarks are reinforced by those of Dr. Patricia Saunders, Director of Defense Test, System Engineering and Evaluation. In a paper titled *Simulation Based Acquisition* [4] she states that the DoD needs to become a smart buyer, and that in evaluating what to buy, simulation will be a key component. She states that *[w]ithout question the Defense Department is moving toward greater use of simulation-based system acquisition*. She indicates that *[t]he Defense Department envisions an acquisition process supported by the robust, collaborative use of simulation technology that is integrated across acquisition phases and programs. The objectives of Simulation Based Acquisition (SBA) are to:*

1. reduce the time, resources, and risk associated with the acquisition process;

2. increase the quality, military utility, and supportability of systems developed and fielded; and

3. enable integrated product and process development from requirements definition and initial conceptual development through testing, manufacturing and fielding.

Saunders goes on to give evidence from commercial and military programs that the use of simulation has had major positive impacts from the perspectives of cost, schedule and productivity. Citing some of her examples:

Cost: *In the Joint Strike Fighter program it is projected that the virtual manufacturing techniques may save as much as 3 percent of the program's life-cycle cost, which could be $5 billion.*

Schedule: *The use of modeling and simulation tools and processes by the "big three" auto manufacturers has reduced the time from concept approval to production from 5 to 3 years*

Productivity: *It took 38 Sikorski draftsmen approximately six months to develop working drawings of the DH-53E Super Stallion's outside contours. In contrast using modeling and simulation one engineer was able to accomplish the same task for the Commanche helicopter in just one month.*

Clearly the use of simulation in the above examples is different from that in software development. However, there are sufficient parallels that would tend to indicate that similar advantages can be accrued in the software arena. For example, while physical mock-ups are not used in software development, early prototypes are used for the same reasons - i.e., determining system characteristics prior to large investments in implementation.

There are other common problems shared between the physical systems described above and software systems. Examples are:

- the management of changing requirements, and predicting the consequences of such changes,
- the development and optimization of effective processes through which the product is built,
- the estimation and tracking of project costs and schedules.

In addition, this year (1998), the DoD has developed an overall action plan to integrate the various simulation-based acquisition activities on-going at the DoD. This action plan was developed by a Joint SBA Task Force whose aim is *an acquisition process in which the DoD and industry are enabled by robust, collaborative use of simulation technology that is intended to integrate across acquisition phases and programs* [5].

The need for simulation in software engineering

Why can simulation enhance traditional software engineering? An important factor is that it provides insights into complex process behavior. Like many processes, software processes can contain multiple feedback loops, such as associated with correction of defects in design or code. Delays resulting from these effects may range from minutes to years. The complexity resulting from these effects and their interactions makes it almost impossible for human (mental) analysis to predict the consequences. Unfortunately, traditional process analysis does not shed much light on these behavioral issues, and the usual way to resolve them is to run the actual process and observe the consequences. This can be a very costly way to perform process improvement.

Assessing the costs of software development

At an applied level, simulation can support project costing, planning, tracking, and prediction. In a competitive world, accurate prediction provides a significant advantage. If cost estimates are too high, bids are lost, if too low, organizations find themselves in the red. In this context, simulation can provide not only estimates of cost, but also estimates of cost uncertainty. Simulation is a powerful tool to aid activity-based costing, and can incrementally accumulate costs to a very fine degree of resolution. In addition, it can assess the uncertainty of costs based on the interacting uncertainties of independent variables [16, 17].

Supporting metric collection.

Simulation is effective only if both the model, and the data used to drive the model, accurately reflect the real world. There is tight connection between the model and the data in the following sense. A simulation can only be executed if it is supplied with numerical drivers, and this forces the developer to identify points in the model where these drivers are needed. For example, the model may have to know what percentage of design documents pass review and what percentage must be returned for further work. Thus, in the very construction of the model, points where metric data must be collected fall out as a bonus. This approach forces one to collect metric data in a consistent sense from a systems perspective - it is not simply a collection of "nice to have" data. Very often, too much or too little metric data is collected because the analyst does not have clear guidelines on what is essential.

Building consensus and communication

When changes to a process are proposed, experience is likely to be the most important influence. However, experience may not be able to correctly assess behavioral changes resulting from process modifications. One person's experience may not correspond to another's, and subjective judgement comes into play as to whose opinion is correct. Usually the person with greater authority wins. With the ability to quantify the effects through simulation, a much greater degree of insight and understanding can be brought to bear on the decision-making process. Thus simulation can be a significant influence in communication and consensus building. In this context, alternate process designs can be considered in a quantitative manner with respect to such issues as bottlenecking, resource availability, throughput, and costs. These analyses should result in processes that, once installed, will have a considerably higher probability of satisfactory operation.

Leveraging simulation across applications.

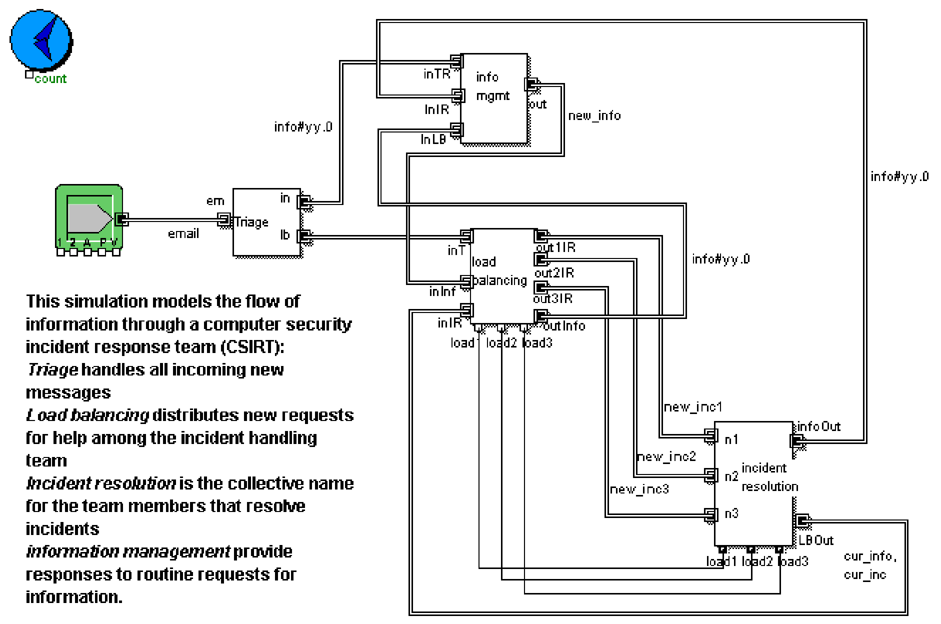
As illustrated in the next section, simulation can support a wide variety of applications. Thus the marginal investment in simulation tools, training and experience-building diminishes as the technology is introduced to successively new applications.

A Discrete Simulation Model

There are many approaches to simulation. Some simulations are based on the need to visualize the airflow across a wing section, while others, such as in combat or flight training, have a need for a virtual reality component. However, the types of simulations we focus on here use symbolic networks of linked elements that model processes or products. For example, we can model entities flowing through an organization consisting of departments, or model information flowing between a set of integrated software tools. Techniques such as discrete event simulation, and systems dynamics are often used here.

To make concrete the type of simulations we are talking about, Figure 2 show components of a discrete simulation model. (It is called "discrete" because the entities that flow through the system are modeled discretely.) The model was developed with the Extend tool [1], and depicts a call-center type of process for a computer security incident response team. CSIRTs, such as CERT® at the Software Engineering Institute, support organizations that have been compromised by unauthorized computer intrusions, or wish to obtain information about guarding against such intrusions. CSIRTs have to communicate with many victimized organizations and one incident may be composed of numerous email dialogs. Hence there is a need for a formal workflow process to manage the large number of interactions, making sure that efficiency and responsivenes are maintained.

Figure 1 (i.e., triage, load balancing, incident resolution, and information management), and these are all modeled in the simulation. Figure 2 illustrates the sub-process for the load-balancing area.



This simulation models the flow of information through a computer security incident response team (CSIRT):
Triage **handles all incoming new messages**
Load balancing **distributes new requests for help among the incident handling team**
Incident resolution **is the collective name for the team members that resolve incidents**
information management **provide responses to routine requests for information.**

Figure 1: Top-level components of the incident handling model

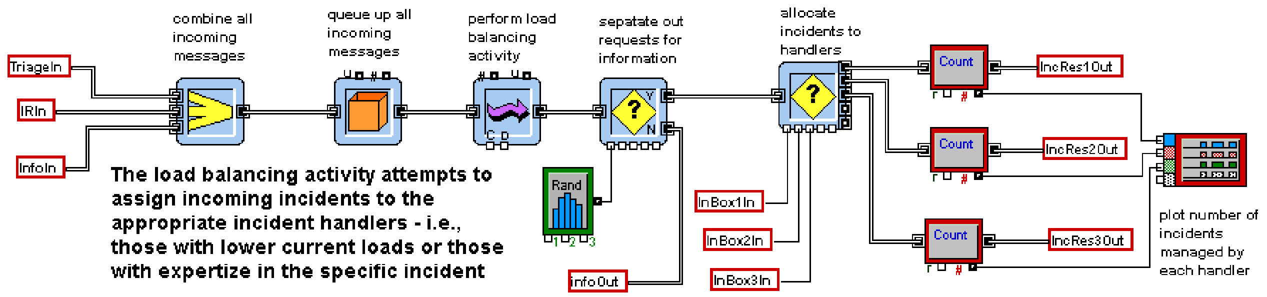

combine all incoming messages
queue up all incoming messages
perform load balancing activity
sepatate out requests for information
allocate incidents to handlers
TriageIn
IRIn
InfoIn
The load balancing activity attempts to assign incoming incidents to the appropriate incident handlers - i.e., those with lower current loads or those with expertize in the specific incident
Rand
InfoOut
InBox1In
InBox2In
InBox3In
Count
Count
Count
IncRes1Out
IncRes2Out
IncRes3Out
plot number of incidents managed by each handler

Figure 2: The Load-Balancing sub-process

On running the model, various plots can be produced. In this example we plot the queue in front of incident handler 3 (see Figure 3) and the number of emails completed for each of the incident handlers (see Figure 4)

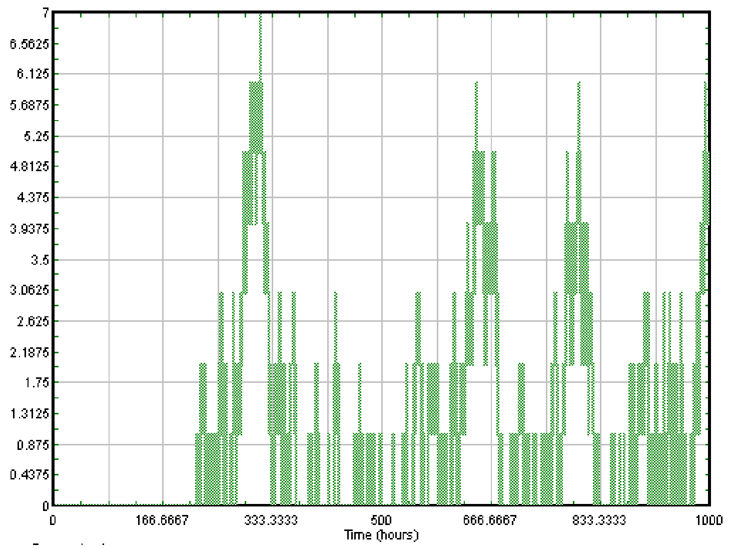


Figure 3: Figure 3: The email queue for incident handler 3

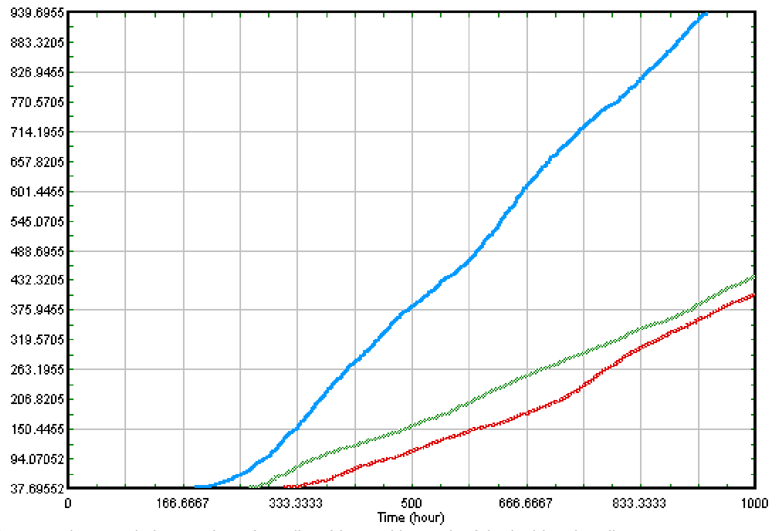


Figure 4: The cumulative number of emails addressed by each of the incident handlers

Simulations such as this can be very useful for designing effective processes and for predicting the resources needed (both human and computer) in order that the anticipated loads can be handled. This model contributed to the task of generating synthetic incident data that supported performance tests on a CSIRT workflow environment. Without such data, we would not have been able to assess performance at such as early state of the workflow system's development.

For more background on technical issues associated with simulation modeling, [2] should be consulted.

Target applications for simulation in software engineering

Simulation has been applied in many fields, such as aerospace or energy production, but to date it has not seen broad practical application in software engineering. This may be because it is more difficult to accurately model human and organizational behavior than to model physical systems, or it may be that the emphasis on software process is a relatively recent phenomenon. Whatever the reason, this is unfortunate because the rewards from it use are many-fold. In this section we briefly review applications of simulation (in no particular order) and some of benefits that can be obtained.

Requirements management

Simulation can be of major help in pinning down software system requirements early in the product lifecycle, particularly for examining temporal behavior. Simulation can mimic the performance characteristics of software components and their interactions, the effects of time delays and feedbacks, and of finite capacities and resource bottlenecks [14, 15]. The CSIRT example provided above illuminates these issues. Alternate architectures and designs can be evaluated in a safe environment, prior to implementation. In addition, requirements are rarely static, but evolve as experience grows with product development. Thus simulation is not only a valuable tool in defining the initial requirements, but can be used to test out alternate modifications, prior to their implementation. Finally, a system simulation can be viewed as a component of the requirements and can provide quantitative measures against which the target software system must comply.

The processes through which the requirements are managed are also critical. However, as far as modeling is concerned, such processes have much in common with other project management processes (e.g., design, development or test). Thus the discussion below under "project management" is relevant to requirements management.

Project management

Simulation can allow managers to make more accurate predictions about both the schedule and the accumulated costs associated with a project [19, 22]. This approach is inherently more accurate than costing models based on fits to historical data, since it accounts for the dynamics of the specific process. With regard to schedule, simulation can account for dependencies between tasks, finite capacity resources, and delays resulting from probable rework loops. Some simulation tools also allow one to compute the accumulation of costs on an activity-dependent basis. These features are useful for generating proposal that are more accurate in cost and schedule are thus more likely to keep a company in business!

Training

Because of the complex dependencies between attributes of organizational systems, these systems can respond in counter-intuitive ways. (The classic example is Brooke's law, which states that hiring personnel late in a project can further delay the project.) Simulation can play an important role in sensitizing managers to the consequences of instabilities resulting from the system feedbacks that can be inherent in badly designed organizational processes. Simulation-based training can provide software development managers with the insights necessary to establish effective processes and to operate these processes in stable manner. Thus the focus here is training management of the design and operation of software processes, not about training them in the technologies that support software development.

Simulation-based training can be performed by individual managers who interact with the simulated software development activities. This individual has control over certain control parameters (such as hiring rate, salaries), and the decisions made alter the course of the subsequent simulation history. Analysis of a training session can be performed after the session to see what went right and what went wrong in the decision making process and to reinforce effective decision making.

To bring groups of managers to a central location for training can be both costly and time consuming In the near future, we may therefore train such groups in a geographically distributed manner, using a simulator whose displays are viewed on the manager's local terminals. Interaction between trainees (allowing for joint decision making) can be provided through the use of collaboration technology. With the increasing interest in this technology, distributed training may soon become practical.

Process Improvement

Simulation can be used to support process improvement at all levels of the Capability Maturity Model® (CMM), but particularly at the higher levels [12, 20,21]. Because simulation forces one to address metrics and process behavior in a methodical way (see the paragraph on *supporting metric collection*), one may argue that simulation can accelerate the introduction of process improvement. Consistent with the philosophy of the CMM, simulation capability at each CMM level incrementally builds upon the simulation capabilities of the preceding levels, and matches the needs of the software engineering practices at that level [6].

Traditionally, revised or new processes are improved through operational experience. This can be expensive and risky. Simulation can provide considerable insights into how a process will work, prior to its implementation. These insights can help the process designer assess alternatives, and show that a specific process design performs in a manner that meets expectations. In this way, processes can be pre-tested, and buy-in is more likely obtained from management. Subjective criticisms are less likely, since quantitative simulation of validated models can produce very specific and credible answers to perhaps hostile questions.

Architecture and COTS integration

Building complex software systems usually begins with addressing the system's architecture. Without a firm notion of how the major components of a software system interact, there is little likelihood that the system will reflect perform effectively. One would like to know early in the development cycle that such attributes as reliability, reusability, maintainability, portability, performance, and modifiability are above some acceptable level. There are complex dependencies between these attributes. For example in improving performance, reusability might be sacrificed, or in improving portability, maintainability might require increased effort. Making tradeoffs in this multi-dimensional space is not easy, but if they are not made at a high level of design abstraction, there is little chance that they can be dealt with once coding begins. Simulation is a tool that can be used to examine some of these architectural tradeoff issues [7]. Simulation can provide early insights, for example with timing, resource usage and bottlenecking, and also into usability. In addition, one can rapidly gain insights into the implications of design changes, by running simulations by varying independent parameters. Finally one can assess sensitivities to parameter changes in a Monte Carlo (i.e., statistical) sense.

Product-line practices

An area where simulation makes considerable sense is in the economic analysis of product lines. In particular, *[b]ecause product line development involves changes in product composition and production, software size measures such as lines of code are not good predictors of productivity improvements. To estimate, track and compare total costs of disparate assets, adaptation of other cost modeling techniques, particularly activity-based costing to asset-based software production is needed* . As mentioned above, some simulation tools incorporate activity-based costing such that, as entities flow through the simulated process, the cost associated with the processing of each entity at each stage can be accumulated. In this way detailed cost predictions can be made with respect to different product line strategies.

Risk management

Projects are often vulnerable to risks resulting from, for example, requirements ratcheting, changing staff levels, funding cuts, and organizational disruptions. Simulation can help identify associated project risks early on. By quantitatively predicting the consequences of alternate decisions, simulation can help design more objective, less risk-prone strategies. There are also risks associated with alternate system architectures or COTS integration strategies. By using simulation to examine the potentially complex interactions of alternate component configurations, the pros and cons of different design decisions can be identified.

Acquisition management

Acquisition management is likely to be dependent on many of the practices described above (for example, requirements management, project management, risk management). Since these practices all can benefit from the use of simulation, so also can acquisition management. More specifically, simulation can help validate a contractor's estimates of costs and schedules and provide insights into the ability of the contractor's design to meet system requirements. Thus through the use of simulation, a project manager can predict potential contractor problems before they become reality. Simulation also has the effect of keeping the contractor honest in its estimates of cost and schedule.

The subject of simulation-supported acquisition has been addressed in some detail by Schacci and Boehm [9, 10]. In these papers, they address the issues of how simulation can support the acquisition lifecycle. They give specific examples of potential applications, and suggest that a research and development effort be established to explore issues such as virtual prototyping, incremental iterative acquisition supported by simulation, and the use of wide-area collaboratories.

Every silver lining has a cloud

As a cautionary note it is well to remember that simulation is not a panacea. The predictive power of simulation is strongly dependent on how well the models are validated. While many scientific and engineering fields can base their models on established physical law, organizational models have to deal with human and other less quantifiable issues. Not only is gathering data difficult when that data must come from human actors, but the reproducibility of scenarios used to validate models cannot as easily be standardized as in experiments based on physical law.

Simulation is a simplification of the real world, and is thus inherently an approximation. As indicated in [13] *it is not possible that a model is absolutely correct. Therefore model [verification and validation] is concerned with creating enough confidence in a model for its results to be accepted. This is done by trying to prove that the model is incorrect. The more tests that are performed in which it cannot be proved that the model in incorrect, the more confidence in the model is increased.*

It is well to remember, however, that the usual alternative to simulation is to rely on human intuition, which is often biased by *mental blindspots' or 'mental tunnels' where we systematically make grave errors and get sidetracked into the wrong answer in certain kinds of problems* [11].

The state of simulation technology

Less than a decade ago, one could only develop a simulation by textual coding of the model. However, since the early 1990s, graphical simulation tools have become available. These tools

- allow rapid model development through, using, for example,
  - drag and drop of iconic building blocks
  - graphical element linking
  - syntactic constraints on how elements are linked
- are less error prone
- require significantly less training
- are easier to understand, reason about and communicate to non-technical staff.

Because of these features, network-based simulation tools allow one to develop large detailed models quite rapidly. The focus thus becomes less on the construction of syntactically correct models and more on the models' semantic validity and the accuracy of their numerical drivers.

The simulation tools in today's market place are robust and reasonably inexpensive. Most tools cost in the range of $500 to $1000. They are therefore accessible by organizations wishing to explore the applications described above.

References

1.http://www.imaginethatinc.com/

2. http://www.pitt.edu/

3. http://www.acq.osd.mil/te/speeches/speeches.html [Note: Because this document or Web site is no longer available online, the link to it was removed from this file.]

4. http://www.acq.osd.mil

5. http://www.msosa.dmso.mil [Note: Because this document or Web site is no longer available online, the link to it was removed from this file.]

6. A. M. Christie, "Simulation in Support of CMM-based process improvement" to be published in Journal of Systems and Software.

7. http://www.sei.cmu.edu/publications/documents/97.reports/97tr029/97tr029chap03.htm

9. http://sunset.usc.edu

10. Schacci, W., Boehm, B., *Virtual Systems Acquisition: Approach and Transitions*, Acquisition Review Quarterly, Vol. 5 No. 2.

(Spring 1998).

11. Massimo Piattelli-Palmarini. Inevitable Illusions: how mistakes of reason rule our minds, John Wiley, 1994. See also http://public.logica.com/~stepneys/bib/nf/piattell.htm.

12. D.M Raffo and M. I. Kellner, "Using Quantitative Process Modeling to Forecast the Impact of Potential Process Improvements", *Proceedings of the 10th International Forum on COCOMO and Software Cost Modeling,"* Pittsburgh, PA, Oct. 1995.

13. S. Robertson, "Simulation Model Verification and Validation: Increase the Users' Confidence", *Proceedings of the 1997 Winter Simulation Conference*, pp53-59.

14. Belscher, R., "Evaluation of Real-Time Requirements by Simulation-Based Analysis," First IEEE International Conference on Engineering of Complex Computer Systems, Los Alamitos, California: IEEE Computer Society Press, November 1995.

15. Lerch, F., et al., "Using Simulation-Based Experiments for Software Requirements Engineering," *Annals of Software Engineering*, 3, N. Mead, ed., 1997.

16. Summary of CAPI Developed Simulations for the Postal Service.

17. L. L. Gardner, M. E. Grant, L J. Rolston "Using simulation to benchmark traditional vs. activity-based costing in product mix decisions", WSC '94. *Proceedings of the 1994 Conference on Winter Simulation*, pp1050-1057.

18. K. Kirby, R. Sawhney, "Simulation: shifting the competitive edge for the next generation", MDC Update (University of Tennessee), Volume 6, 1997.

19. M. I. Kellner, "Software Process Modeling Support for Management Planning and Control", First International Conference on the Software Process", Redondo Beach, CA, pp 8-28, 1991.

20. G. A. Hansen, "Simulating Software Development Processes", *IEEE Computer*, January 1996.

21. J. D. Tvedt, J. S. Collofello, "Evaluating the Effectiveness of Process Improvements on Software Development Cycle Time via System Dynamics Modeling", *Proceedings of the 19th Annual International Computer Software and Applications Conference (COMPSAC'95)*, 1995.

22. T. Abdel-Hamid, S. E. Madnick *Software Project Dynamics*, Prentice Hall,New Jersey, 1991.

Acknowledgements

I would like to thank Jim Withey, Bill Riddle, Anthony Earl, and Caroline Graettinger for their review of this article.