TECHNICAL CORRESPONDENCE

In Defense of Discrete-Event Simulation

In the October 1991 issue of *TOMACS*, Bagrodia, Chandy, and Liao present "A Unifying Framework for Distributed Simulation." The paper is a well-conceived, well-written contribution to the literature. However, I wish to call attention to one paragraph and discuss its implications; my object being to stimulate a discussion within the parallel discrete-event simulation community concerning the relationship between discrete-event simulation and parallel discrete-event simulation. The authors write:

> "We can attempt to define the simulation problem by describing a well-known solution method—say, sequential discrete event simulation—and then by defining the problem as: "Obtain an algorithm that gives the same results as sequential simulation." This approach has the advantage of defining the problem in terms of something very familiar. Indeed, much of the work on distributed simulation takes this approach and thus avoids the need for formal specification. Defining a problem in terms of a well-known solution method has the disadvantage that it is likely to constrain our search for novel methods because we will be much too heavily influenced by the method that we start with. In particular, we will be inhibited in discovering novel distributed simulation algorithms if our point of departure is the standard sequential algorithm. Therefore, we attempt to define the simulation problem, while keeping the problem definition as untainted by solution methods as we can."

The wording and tone of this paragraph shows a subtle disregard for discrete-event simulation (DES) that seems to permeate much of the research in parallel discrete-event simulation (PDES). Similar to the development of the body of theory for operating systems, PDES research has (necessarily) been a bottom-up endeavor. Factors that relate directly to the execution of simulation programs on parallel machines must be addressed early on in order to facilitate initial performance estimates, and thus establish a feasibility for further investigation in the field. The subtlety of the problem arises from a tendency toward myopic concern with the execution of simulation programs and a failure to recognize the "big picture," that is, the existence of a *simulation model life cycle* and the roles and range of influence of the various phases, processes, and artifacts enunciated by the life cycle. So a real danger lies in saying that

> "We can attempt to define the simulation problem by describing a well-known solution method—say, sequential discrete-event simulation—and then by defining the problem as: "Obtain an algorithm that gives the same results as sequential simulation."

because it seems to regard discrete-event simulation as merely an algorithm for the execution of simulation programs on sequential computers. Research in DES dates to the 1950's and involves topics that span the simulation model life cycle, including modeling methodology, simulation programming

languages, model formulation, random number and random variate generation, output analysis, and verification and validation of simulation models.

Clearly, the requirements for the execution of programs on parallel computers may have far-reaching influence in the life cycle. Two approaches to dealing with this problem become evident. The first is to assert that parallel discrete-event simulation is entirely and fundamentally different from sequential discrete-event simulation, and thus PDES must be developed with its own theory and life cycle independent of DES. This philosophy takes the extremist position. The second approach is to identify and resolve problems associated with the *addition* of parallel implementations within the current DES life cycle. This philosophy adopts a more moderate stance. Much of the research on PDES, at least implicitly, advocates the former approach, through a failure to explicitly address potential impacts of the research on the various phases of the simulation model life cycle—impacts that, if addressed, might obviate the research entirely. It may be fairly argued that radical approaches are the best way to discover novel and superior solutions. This position is not without merit. On the other hand, the pragmatist (some may read cynic here) recognizes that acceptance of a new technique hinges primarily on its integratability with existing familiar techniques as well as its bottom line: how much of an investment will be required to adopt the new approach and its cost-effectiveness in both the long- and short-terms.

While good points can be made in favor of the former approach, arguments for the latter seem at least as compelling. If one is to attack the PDES problem within the context of discrete-event simulation (and software engineering), questions related not only to the executable representations of models but also as to how the executable forms are produced and to the nature and influence of higher level representations must be posed and answered. (To be fair, the authors identify this concern when they speak to the need for formal specification uninfluenced by "well-known" methods.) To cite some instances:

(1) Are the existing conceptual frameworks totally inadequate to allow efficient parallel execution of programs derived from conceptual models and from specifications constructed using them? Or have these conceptual frameworks just been applied in a conventional over-constraining fashion?
(2) Are there inherent efficiency-related differences in executables constructed from operational specifications versus those constructed from algebraic specifications?
(3) Do object-oriented specifications naturally lead to inefficient parallel programs?

Finally, some matters that heretofore have gone largely unaddressed. Is PDES to be a viable alternative when execution speed is a desirable but not a preeminent concern? How are verification and validation affected by the addition of parallel implementations? What about software quality factors like reliability, portability, and adaptability? From the standpoint of software

quality, what are the *costs* involved in attaining speed-up of execution? For instance, what is the cost in terms of maintainability of an N-fold speed-up in execution using an extant PDES paradigm, as compared to a sequential implementation in some existing simulation programming language? Since the cost of maintenance dominates the cost of software (some estimate over 70%) even small increases in maintenance costs can be prohibitive.

In closing let me stress that my intention in writing is not to be critical of PDES research or researchers in general, and certainly not to criticize the authors. I am more in agreement with their philosophy than that of many others; the nature of the authors' presentation merely provides a convenient platform on which to raise these issues. My hope is to stimulate within the PDES research community a recognition that model execution cannot exist in a vacuum, and that research should be conducted with at least an *enunciation* of the researcher's view of the bigger picture; so that while cutting away the underbrush of the PDES problem, we do not lose sight of the forest for the trees.

ERNEST H. PAGE
Virginia Tech
Department of Computer Science
Blacksburg, VA 24061
email: *srcpage@popeye.src.vt.edu*