Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

PARALLEL DISCRETE EVENT SIMULATION: THE MAKING OF A FIELD

Richard M. Fujimoto

School of Computational Science and Engineering
Georgia Institute of Technology
266 Ferst Drive
Atlanta, GA 30332, USA

Rajive Bagrodia

Scalable Network Technologies Inc.
6059 Bristol Parkway #200, Culver City, CA 90230
Computer Science Dept., University of California
Los Angeles, CA 90095, USA

Randal E. Bryant

Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA

K. Mani Chandy

Computing and Mathematical Sciences
Department
California Institute of Technology
Pasadena, CA 91125, USA

David Jefferson

Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550, USA

Jayadev Misra

Computer Science Department
The University of Texas at Austin
Austin, TX 78712-1757, USA

David Nicol

Department of Electrical and
Computer Engineering
University of Illinois, Urbana Champaign
1308 W. Main St., Urbana IL 61820, USA

Brian Unger

Computer Science Department
University of Calgary
2500 University Drive
Calgary, Alberta T2N 1N4 Canada

ABSTRACT

Originating in the 1970's, the parallel discrete event simulation (PDES) field grew from a group of researchers focused on determining how to execute a discrete event simulation program on a parallel computer while still obtaining the same results as a sequential execution. Over the decades that followed the field expanded, grew, and flourishes to this day. This paper describes the origins and development of the field in the words of many who were deeply involved. Unlike other published work focusing on technical issues, the emphasis here is on historical aspects that are not recorded elsewhere, providing a unique characterization of how the field was created and developed.

1 INTRODUCTION

Parallel discrete event simulation (PDES) is a field concerned with the execution of discrete event simulation programs on a parallel computer. The field began with work in the 1970's and 1980's in first defining the synchronization problem along with associated terminology (e.g., logical processes) and the

development of algorithmic solutions. Seminal work resulted in two approaches. The first, now called conservative synchronization, grew from pioneering work by two groups working independently and without knowledge of each other in the late 1970's. K. Mani Chandy and Jay Misra at the University of Texas in Austin (Chandy and Misra 1979), and a master degree student at MIT, Randy Bryant (Bryant 1977a) developed what is now referred to as the Chandy/Misra/Bryant (CMB) algorithm. A few years later, David Jefferson and Henry Sowizral at the Rand Corporation came up with an entirely different approach known as Time Warp (Jefferson 1985), resulting in a class of methods termed optimistic synchronization. These remain the major classes of algorithms used in PDES today.The late 1980's and 1990's saw the solidification of the PDES field, and most importantly, the establishment of researchers who would continue to work in this area. On a technical level, work in the field during this period gave rise to two camps – those promoting conservative methods, and those promoting optimistic approaches. Researchers in each camp relished the opportunity to promote the advantages of their favorite methods, often at the expense of the other camp. This period saw the formation of the annual PADS conference where researchers met to debate the finer points of these approaches, as well as other related technical issues.

The 1990's brought two important developments to the field. One was the creation of the High Level Architecture (HLA) standard led by the U.S. Department of Defense, and subsequently adopted as IEEE Standard 1516 (IEEE Std 1516.3-2000 2000). Its broad applicability encompassed many applications, not just defense, and enabled much broader deployment of the technology. The second development was the commercialization of PDES technology in industry leading to commercial products and services.

In the following each author gives their own perspective on the origin and evolution of the PDES field in their own words. Mani Chandy, Jay Misra, Randy Bryant, and David Jefferson provide their perspectives dating back to their work in developing seminal algorithms. David Nicol and Richard Fujimoto, who have remained active researchers in the PDES field from its beginnings, provide their own personal perspectives on the evolution of the field. Finally, longtime PDES researchers and entrepreneurs Brian Unger and Rajive Bagrodia discuss the field as well as past and on-going commercialization efforts. As chance would have it, the authors all recently attended Jay Misra's retirement celebration and/or the 2016 PADS 2016 conference and are depicted in Figure 1.

Figure 1: Left: PDES researchers at Jay Misra's retirement celebration (April 29, 2016, Austin, Texas). Front row (left to right): David Jefferson, Richard Fujimoto, Jay Misra, Mani Chandy and Randy Bryant. Back row: Rajive Bagrodia (third from left). Also pictured are Manfred Broy and Leslie Lamport. Right: PDES researchers from the 2016 PADS conference (May 16, 2016, Banff, Alberta Canada). Left to right: Brian Unger, David Jefferson, George Riley, David Nicol, and Richard Fujimoto.

2 CONSERVATIVE SIMULATION (K. MANI CHANDY AND JAYADEV MISRA)

In mid 1970s we were doing research on the performance analysis of computing systems using discrete-event simulation and probability theory. We were also working on concurrent algorithms and methods of proving their correctness. Later, we worked on concepts such as knowledge and learning in distributed systems. This combination of interests led naturally to the question: Can discrete-event systems be efficiently simulated on a distributed set of machines?

The fundamental problem in parallelizing the traditional sequential algorithm is management of the event list, which is inherently sequential. We developed algorithms (Chandy and Misra 1979; Chandy et al. 1979; Chandy and Misra 1981; Chandy and Misra 1986; Bagrodia et al. 1987) based on our work in distributed computing, to overcome this problem. Independently, Randy Bryant also developed this algorithm which is now known as the Chandy-Misra-Bryant algorithm. This algorithm falls within the class of conservative, as opposed to optimistic, algorithms.

2.1 Knowledge

A principle relating simulation to computing by a distributed set of agents is an agent's *knowledge* (Misra 1986; Halpern and Moses 1990). A set of agents *knows fact f* at a point in a computation if *f* can be deduced given only the states of the agents in the set. An agent *learns f* between two points in a computation if the agent did not know *f* at the earlier point and knows *f* at the later point. At each point in a computation each agent knows some facts about the history of the computation and may learn more as the computation progresses. There may not be an efficient algorithm (or even any algorithm) to compute *f* even when an agent knows *f*. Agent knowledge allows us to separate two concerns: (1) we first design a simulation based on how agents learn, during the course of a simulation, what they finally need to know, and (2) later design algorithms by which agents compute the relevant knowledge.

2.2 Queuing Networks

Queuing networks were our motivating examples in developing the simulation algorithm. We partition a network into subnetworks each of which is simulated by an agent. In a time-stepped simulation each agent knows the state of a subnetwork at some moment t (all times are henceforth integers); consequently, the agent knows which jobs depart at time $t+1$ from that subnetwork. In a sequential simulation each agent knows the time of the next departure from its subnetwork if no job arrives earlier. Our algorithm extends earlier algorithms as follows: each agent communicates *all* its knowledge about future departures to other agents. Consider a first-come-first-served queue that has a certain number of jobs at moment t. The agent simulating the queue knows their departure times, and hence the departure time, T, of the last job. That is, the agent knows the job transitions from this queue in the interval $[t, T]$ and communicates this information to others, as given below.

A message in the simulation is a triple (e, t, v) where e is an edge of the queuing network, t is a time and v is a set of jobs indicating that the set of jobs v traveled along edge e at time t. Further, any job that travels along e before t has already appeared in such a message.

2.3 Deadlock

When designing a distributed simulation algorithm we ran into the problem of deadlock. To see this consider an agent simulating a queue q that has two incoming edges e and e', and which receives a message (e, t, v) initially. The agent *cannot* conclude that no job arrives at q before t because a message (e', t', v'), where $t' < t$ may be received by the agent *later* in the simulation. So, this agent has to wait until it receives messages along all its incoming edges, and then choose the one with the lowest time for processing. Such waiting by all agents in a cycle can cause deadlock. Note that the agents simulating the queues in the cycle *collectively* do know the next job that arrives at some queue in the cycle, but

individual agents in the cycle do not have that knowledge. Deadlock is broken by algorithms that enable an individual agent to learn what the collective knows.

2.4 Deadlock Resolution

To overcome the problem of deadlock we needed two innovations: the introduction of *null* messages and algorithms for breaking deadlock. A null message is of the form *(e, t, v)* where *v* is the empty set, denoting that no job travels along *e* up to time *t* from the given moment in the simulation. This allows the receiving agent to extend its knowledge over a longer time horizon. The introduction of null messages avoids deadlock; however, some simulations may have a large number of null messages, each of which only moves the time horizon forward by a small amount. In these cases, a more efficient solution is to run the simulation to deadlock, have a superposed algorithm detect the deadlock and then resolve the deadlock by using the collective knowledge of the agents.

2.5 Final Remarks

Our initial effort was directed towards simulation of queuing networks. Subsequently, conservative algorithms have been refined and improved by many researchers and applied to several areas including VLSI, communication networks and real-time control systems. We believe that the research in the area will continue to make simulation of large systems in different domains more efficient.

3 FROM DATAFLOW COMPUTING TO PDES (RANDAL E. BRYANT)

My involvement in discrete-event simulation came about as a graduate student in MIT's Dataflow Computing group, headed by Prof. Jack B. Dennis. Dennis and his associates had proposed creating a new class of *dataflow* computers, where a number of independent processing elements cooperatively execute a program by repeatedly performing computations and sending the results to other processing elements according to data dependencies in the expressions to be evaluated (Dennis and Misunas 1974). Rather than having a program counter determine which steps the machine should carry out, control is integrated into flow of data among processing elements. This model was inspired, in part, by the IBM implementation of high performance arithmetic units (Tomasulo 1967), variants of which are found in almost all modern processors.

Dennis generalized his ideas on dataflow computers to a complete methodology for designing computer systems, which he referred to as *packet communication architectures* (PCA) (Dennis 1975). Although that name was not widely adopted, and Dennis' original paper has been lost in time, its abstract is informative:

> "Packet Communication Architecture is the structuring of data processing systems as collections of physical units that communicate only by sending information packets of fixed size, using an asynchronous protocol. Each unit is designed so it never has to wait for a response to a packet it has transmitted to another unit while other packets are waiting for its attention. Packets are routed between sections of a system by networks of units arranged to sort many packets concurrently according to their destination. …"

Similar system design principles were developed around the same time by Gilles Kahn (Kahn 1974), and such systems are now referred to as *Kahn Process Networks*. Both Dennis' model and Kahn's assumed the processing elements are connected by unbounded FIFO channels. Hoare's Communicating Sequential Processes (CSP) model, devised in the same general time period (Hoare 1978), also conceives of a system as a set of independent, message-passing processes, but these provide no buffering—the sending of a message by one process is synchronized with its receipt by another.

Dennis had plans to build a general simulation facility for PCA systems (Leung et al. 1976). In retrospect, it would have been much simpler to simulate these systems using discrete-event simulation software running on a single machine, but, inspired by this new vision of system design, they conceived

the simulation facility to itself be structured according to PCA principles. That is, the facility would consist of a number of independent microprocessors connected by a routing network implementing FIFO channels.

Even though the general plans had been laid, and hardware design had commenced, the designers for this facility had no scheme for accurately modeling the temporal behavior of the system being simulated. As a young graduate student looking for a master's thesis topic, they suggested I look into this.

I had no experience in simulation, but it was clear that a conventional approach based on a centralized event list would be neither practical, nor in the spirit of the design principles being explored. Instead, I thought about virtualizing time, incorporating time stamps into the messages being sent between processing elements and relying only on message passing to control the simulation. With this starting point, the principles of conservative distributed simulation became the most natural choice. Each processing element waits for messages on all of its incoming links, and then processes the one(s) with the earliest arrival times. Such an approach can lead to deadlocks due to the formation of cycles of waiting processing elements. By having elements send out null messages (which I referred to as "time packets"), forward progress can be guaranteed, although it may require passing these messages around a cycle many times before useful work can proceed. The null message approach fit within the design goals of relying only on existing communication channels and with no global synchronization.

In my master's thesis (Bryant 1977a), I also devised a way to terminate the simulation by sending out probe messages that would traverse the network and detect the absence of any data messages, again without any centralized control. I later generalized this to allow the system to periodically jump ahead in simulation time, a method I referred to as *time acceleration* (Bryant 1979). Both of these relied on messages sent along the existing communication channels, rather than by some sort of out-of-band communication among the processing elements.

The simulation facility was never completed, and I never implemented any of my algorithms. The 1977 MIT technical report would have been the only record of my work, and this work would have faded into obscurity had it not been for a visit by Jack Dennis, sometime in 1979, to the University of Texas. There, Mani Chandy described his and Jayadev Misra's ideas on mapping simulation onto the CSP framework. Jack recognized the similarity of their approach to mine and later sent them a copy of my technical report. By this point, Chandy and Misra's paper (Chandy and Misra 1979) was already well along in the publication pipeline, but they arranged for a footnote to be inserted at the bottom of the section on related work. Their paper also demonstrated that this approach would work with the synchronized messages of CSP, a more challenging case than my assumption of unbounded channels.

In retrospect, I can see that my then-novel approach to simulation arose out of the constraints of the problem posed to me, rather than any deep intellectual insight. I had to devise a way to run a simulation within an environment where concurrent processes operated asynchronously, coordinating their efforts only by passing messages among each other. These principles have become well established with the widespread deployment of network-based systems, but they were still novel at the time. Indeed, IEEE held its first conference on distributed systems only in 1979. Having received positive feedback from Chandy and Misra, I published a paper there (Bryant 1979). It is gratifying to see a continued level of effort in distributed simulation research, and its many deployments in actual systems.

4 ORIGIN OF TIME WARP AND OPTIMISTIC METHODS OF PARALLEL DISCRETE EVENT SIMULATION (DAVID JEFFERSON)

The origin of Time Warp and optimistic methods dates to about 1981 when I was an Assistant Professor at the University of Southern California. The Rand Corporation had a research project funded by the Air Force to dramatically improve simulation technology in two ways: (1) to create some kind of modeling language or tools that would allow nonprogrammers (i.e. Air Force generals) to set up and run military simulation models, and (2) to find a way to execute those models in parallel so they would run much faster. Rand at the time had a good deal of expertise bearing on the first goal, but they had little expertise

in parallel computation. They had heard, however, that USC had a new professor (me) who had a background in parallel computation (which was relatively rare back then) and so they called me to ask if I wanted to consult for them on the project.

I knew how sequential discrete event simulation worked from graduate school, but I had not seen any literature on parallel discrete event simulation (PDES). I did not know of Chandy and Misra's (Chandy and Misra 1979) or Bryant's work (Bryant 1977b), so I started working on the problem from first principles. In hindsight, I think that was actually lucky, because if I had read their work my thinking might have been channeled into the paradigm they pioneered.

Like Chandy, Misra, and Bryant (CMB), I imagined a parallel simulation as decomposed into logical processes (LPs) that execute on a distributed platform, and that the LPs communicated and synchronized via timestamped event messages. But whereas they were influenced by simulation of dataflow, networks, and queueing models, I was aiming at military combat models. Hence, while they assumed that there was a static communication graph among the LPs, and assumed both FIFO message transmission down each channel between LPs and also monotonically nondecreasing timestamps on the events in each channel, I could make none of those assumptions. In a combat model there is no static interaction graph, and no reason to suppose that the sequence of events that one LP schedules for another have to be in strictly nondecreasing simulation time order. So I assumed a global, random access communication pattern in a flat global space of LPs.

Initially I thought the classic sequential event list algorithm could probably be parallelized in a fairly straightforward manner. But it quickly became clear to me that the fundamental problem of synchronization was much deeper than expected. I am not sure exactly when the idea of using rollback as a synchronization primitive occurred to me, but I remember feeling forced into it because I could construct very small artificial models with two or three LPs that worked fine when executed sequentially, but would always deadlock when executed in parallel without the ability to roll back! One could of course allow a deadlock and then break it with a global analysis, as Chandy and Misra had unknown to me already proposed (Chandy and Misra 1981), but in my view such a scheme was unscalable in the general case and not attractive.

I presented these initial ideas, including I think the apparent need for rollback in at least some models, to my manager, Phil Klahr, and I asked him if it was OK to consider such bizarre methods. To his great credit his answer was basically that I could propose any method that works correctly and produces parallel speedup. And Phil also made another key decision at that point: to team me with another young Rand researcher, Henry Sowizral.

Henry and I began an intensely productive collaboration that lasted less than two years, but in that time the basic foundations of Time Warp were invented. Together we took the leap and adopted speculative execution with unrestricted, asynchronous, distributed rollback as our fundamental synchronization paradigm, without using any of the classical synchronization primitives that were based on interrupt handling, locking, and process block-and-resume. In so doing we had to reconsider every other aspect of distributed computation and how it was affected by the introduction of rollback. Besides creating algorithms for distributed rollback itself, about which I will say more below, we had to create new approaches to synchronization, scheduling, message queueing, storage management, I/O, runtime error handling, event tie handling, and termination detection, all of which were different from their counterpart algorithms in conventional distributed computation. The majority of that conceptual framing took place over the first nine or ten weeks of our work together. The rest of our collaboration was spent fleshing out the ideas, implementing and refining a prototype implementation, becoming acquainted with the prior literature, writing a major tech report about Time Warp (Jefferson and Sowizral 1982), and exploring other potential applications besides simulation. In all of this work Henry and I were coequal partners.

At first glance the idea of unrestricted, asynchronous, distributed rollback as a synchronization primitive seemed at best a ludicrously expensive, unscalable technique, and at worst perhaps literally

impossible to implement correctly, let alone efficiently. The key problem was that as an LP in the simulation is executing forward speculatively it sends asynchronous event messages to other LPs, and if the LP later rolls back, all of those messages are potentially incorrect and have to somehow be recalled or cancelled. Without stopping the simulation globally, how exactly do you effectively "unsend" a message? The message in question may be in flight, or be enqueued at its destination, or it may already have been processed, giving rise to secondary incorrect messages sent to other LPs, which themselves may be in transit or enqueued or have been processed, prompting more tertiary incorrect messages, etc. The tree of incorrect consequences of the original incorrect speculative events can be growing exponentially and may also develop cycles, even while the effort to cancel them all is still in progress. And, there may be many such trees growing from separate speculative events that end up interacting with one another.

The most important innovation in the Time Warp algorithm, which elegantly resolves all the problems with asynchronous distributed rollback, is the notion of antimessages. An antimessage is in all respects exactly like an ordinary event message, except that is flagged with a negative "sign". In order to cancel an event message `E` sent to receiving logical process P, and also cancel all of its exponentially growing tree of incorrect consequences, all that is required is to send the antimessage `-E` to the same destination P. When `-E` arrives, it is enqueued, and may cause P to roll back just as any positive message does when it arrives in the "past" of the receiving LP. That rollback causes a secondary round of antimessages to chase down any secondary positive event messages that the P had sent. The queueing discipline for event messages and antimessages is also unusual: whether or not an antimessage causes a rollback, it "annihilates" with the corresponding positive message when both are in the same queue, and both of them simply disappear, like particles and antiparticles in physics.

The success of the symmetry between message and antimessages led me to make symmetry a key design principle in Time Warp and its extensions over the years. After our collaboration ended I wrote a widely-read paper, "Virtual Time" (Jefferson 1985), in which I included other applications of Time Warp besides simulation, and described a kind of space-time symmetry in the form of an extended analogy between virtual memory with page faults, and Time Warp with "time faults", i.e., rollbacks. When some years later I was concerned with the problem of flow control and the larger issue of storage management in Time Warp, the symmetry principle led me to introduce the notion of message sendback, i.e. sending a message backward in space and time, from receiver back to the original sender, causing the original sender to roll back to a time before he sent it (Jefferson 1990). Sendback of a message was designed to be symmetric to the forward transmission of a message, and also symmetric to rollback of an event. The introduction of message sendback made flow control and storage management work, and eventually allowed us to prove that Time Warp could be space optimal (Gafni 1985; Jefferson 1990).

Henry and I built the first prototype implementation of Time Warp in InterLisp on a 3 Mb/s network of four Xerox Dolphin workstations. Our first benchmark model was an event-driven version of the cellular automaton known as the Game of Life, chosen because it was easy to program, easy to check that it was working correctly, easily scalable, and with easily adjustable "granularity", since each LP could represent more or less of the cellular area, allowing us to set the communication-to-computation ratio in the simulation to whatever we pleased. After considerable futzing to eliminate background computation that spoiled our initial performance measurements (demand paging, Lisp garbage collection, I/O, and other users' jobs and network traffic) I believe we were eventually able to measure a speedup factor of about 2.5 on our four-node system under ideal, tuned conditions. Unfortunately no record of either the code or measurements survives, since we never published them.

Eventually the collaboration with Henry faded, and starting in about 1984 I had the opportunity to start a Time Warp project at the Jet Propulsion Laboratory. A new parallel machine was being built at Caltech, the Caltech Hypercube, which had 32 nodes connected by communication channels in the topology of a 5-D hypercube (Jefferson et al. 1987). Each node contained an Intel 80286/87 processor pair and 256KB of RAM. While it was intended for physics computations they were also interested in making the case that it was useful for a broad array of other applications. The U.S. Army was our sponsor

this time, and they too were interested in parallelizing combat models. We assembled an outstanding development team headed by Brian Beckman, and later by Peter Reiher, and over the course of seven years (!) built and experimented with the Time Warp Operating System (Jefferson et al. 1985; Jefferson et al. 1987). TWOS was later ported to a successor machine, the 64-node JPL Mark III Hypercube, which had Motorola 68020/68881 processors with a comparatively whopping (for those days) 4MB of RAM per node, and still later to a 112-node BBN Butterfly machine. These machines were big enough to do real scaling performance studies.

Probably the best known study we did was captured in the speedup curve shown in Figure 2 done in July 1988 on the Mark III Hypercube. It shows the speedup relative to a sequential execution of TWOS running a combat model with 380 LPs in configurations from 4 to 32 nodes. In the 32 node run the execution time was reduced from 1.4 hours to down to 5.3 minutes, a speedup of over a factor of 16. We were so pleased we put this graph on a T-shirt.

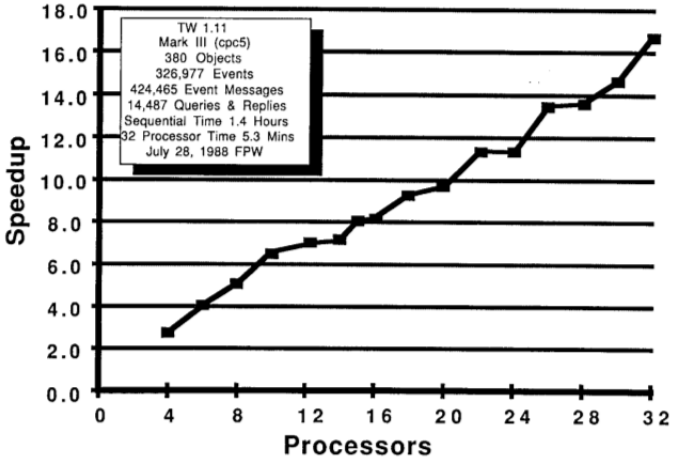

Figure 2. TWOS speedup on the Mark III Hypercube, July 1988.

The TWOS project was in some respects way ahead of its time. It included both lazy and aggressive cancellation, permitted dynamic creation and destruction of LPs, implemented the cancelback protocol extension to Time Warp for global flow control and storage management, allowed dynamic migration of LPs from one node to another to support load balancing, and had a capability for parallel-in-time execution. These features are still not present in most implementations of Time Warp today 25 years later.

By the mid 1980s a rivalry of sorts developed between advocates of *conservative* synchronization methods pioneered by Chandy and Misra and their students, and *optimistic* synchronization based on Time Warp. Many, if not most, researchers were aligned with one side or the other, and many papers were published comparing the strengths and weaknesses of each and conducting comparative performance studies. I, of course was an advocate for optimistic methods.

Some time in the late 1980s Richard Fujimoto spent a summer with the TWOS project and then went on to build his own simulator, Georgia Tech Time Warp. He did the first carefully controlled studies comparing the performance of conservative and optimistic methods on the same (artificial) models and the same platform, sharing as much simulator code as possible. He was able to show that on some models optimistic methods outperform conservative methods by a wide margin, and in other cases the opposite is true by similarly wide margins. Today we understand that a crucial factor is the presence or absence of good *lookahead* information at runtime.

While the rivalry between conservative and optimistic methods was never personal, it did get fairly intense as academic debates in computer science go. It lasted for somewhere between 10 and 20 years, and was more or less transmitted to a couple of generations of graduate students. My own participation in

the rivalry was primarily driven by what I perceived was an uphill battle to get people to seriously consider a radical idea like speculative computation and rollback as a new approach to distributed simulation. Although the rivalry is mostly extinguished now as the issue has come to be fairly well understood, it is still not uncommon to publish performance comparisons between conservative and optimistic methods in the wake of changes in technology, models, and scale and as new generations of researchers re-examine the issue. My current opinion is that both conservative and optimistic synchronization have their place, and neither dominates the other. I think the best approach is to think of a simulator as capable of both optimistic and conservative synchronization, with conservative synchronization used whenever and wherever there is good lookahead information, and optimistic synchronization otherwise.

5 KEY IDEAS IN CONSERVATIVE SYNCHRONIZATION (DAVID NICOL)

The two decades following the discovery of Time Warp were characterized by a competitive dialog between the conservative and optimistic synchronization camps. It is fair to say that the holy grail of many optimistic thrusts was synchronization that is entirely transparent to the modeler. It is fair to say that a mantra of mine, as an experience programmer, was to avoid the overhead of developing the entire infrastructure needed to perform simulations optimistically: state-saving, event-cancellation, anti-messages, and rollback.

To the credit of the optimistic camp, they were correct in the observation that optimistic synchronization is the best hope for a general purpose parallel simulation platform. To the credit of the conservative synchronization camp, we were correct in the observation that conservative techniques are comparatively easy to implement, and that the goal of a general purpose parallel simulation platform that automatically delivers excellent performance is simply a bridge too far. To the credit of both camps, there is now a recognition that good performance generally requires the model developer to have an understanding of the underlying synchronization overheads and model accordingly, that some models can achieve good performance without optimistic techniques, that some models can achieve good performance only with optimistic techniques, and that models which integrate simulation and emulation of compiled protocol stacks can only be performed using conservative methods. The question of "which style of technique is better" is acknowledged now to have the answer "It depends... ."

I see two bedrock observations governing the synchronization issue. The first was exposed in the seminal Chandy-Misra-Bryant work: to avoid having a cycle of logical processes blocking on each other's forward progress, one needs to have what became known as "lookahead", the ability of one logical process to look a little ways ahead into its future behavior and discern that for some positive epsilon of virtual time it will not do anything to affect the state of another logical process. Success at conservative synchronization hinges on an ability to find and exploit such lookahead. The second bedrock issue is that in optimistic synchronization, assurance of pristinely correct replay after a rollback requires support for recovery that goes all the way through the application code, through any middleware it runs on, through all system library calls, and potentially to the lowest layer of the computer stack that contains state. If you cannot reset *all* forms of memory to the exact values it had at the point to which it is returned, it is possible for the replay to deviate from what it should be, and potentially cause logical errors, even application failure. The developers of the Time Warp Operating system understood this, and recognized they needed to develop an entire operating system on optimistic principles. Jason Liu and I eventually caught on to the issue, and described it in "The Dark Side of Risk: Everything Your Mother Never Told You About Time Warp" (Nicol and Liu 1997). I'll leave it to the optimistic folk to grapple with the state issue, I'll now talk more about lookahead.

While still a graduate student I saw that synchronization could and should be tied to the simulation model under study ("Problem Oriented Protocol Design") (Nicol and Reynolds 1984). This was a precursor to later advances in conservative synchronization made by finding classes of problem characteristics that enabled identification of lookahead. A hugely important one is that under certain

circumstances one LP enters a state where for an epoch, further state changes at that LP cannot impact another LP which is sometimes influenced by the first. This was discovered independently by Lubachevsky (Lubachevsky 1988) in the context of physics-based Ising-spin models, and myself (Nicol 1988) in the context of queueing networks. In both cases there is a scheduled point in the future where the influence an LP has on another may change. Lubachevsky called the interim period "opaque". My observation was that in a queueing system with non-preemptive scheduling, when a job went into service at time 90 with a service requirement of 10 it is possible to compute a lower bound 100 on the time at which the next job will leave that server and arrive at another. The impact of the server downstream is opaque to further arrivals at the server, because allocation of service to the active job is insensitive to these changes.

I took the observation a step further and noticed that one can sometimes sample the values of random variables before they are actually needed, and use that to identify opaque periods. For example, in a first-come-first-serve queue one can pre-sample the service times and routed destinations of as many future arriving jobs as one likes. Even if the queue is empty at time 90, if the service demand of the next job to arrive---whenever it does---is known in advance to be 10, then there will be no departures from this server before time 100. The possibilities of using this observation with different queue disciplines and network topologies was thoroughly explored by myself and others. However, while ample lookahead can be found this way, these explorations highlighted the challenge of generalizing a framework that uses it transparently (even within the context of queueing networks.)

With respect to synchronization the battlelines were firmly in place by the late 1980s. Someone proposed a bake-off session at WSC, where a prototypical simulation problem would be proposed, and solved using space-time synchronization (Chandy and Sherman 1989, Bagrodia and Liao 1990), Time Warp, and a conservative solution, with a performance comparison at the end. The problem proposed was "Sharks World", in which sharks cruise in an absolutely straight line through a toroidal world, and gobble up any (moving) fish that lay directly in their path as they move. I believe the problem was chosen to be quite challenging for conservative techniques. To everyone's surprise I brought in a solution that achieved extremely high performance. The Sharks' World problem as stated had a kind of lookahead in spades. The behavior of the sharks and fish were completely independent of each other. I exploited this invariance to set up equations whose solutions identified when and where these shark-fish interactions could occur, and use their solution to compute the state that one would reach had a PDES approach been taken (Nicol and Riffe 1990). The example illustrates what I think is an important principle of conservative synchronization---use domain knowledge to your advantage. However, it can also be used to point out the fragility of that approach. If the sharks dynamically chased fish the solution I crafted would have been, well, shark bait. There is a balance to strike between the depth of exploitation of problem characteristics, and the resilience of the solution to changes in the problem statement.

I meet Phil Heidelberger at WSC following the publication of some of the pre-sampling work, and he had the instinct that the technique of *uniformization* used in the solution of steady-state equations for continuous time Markov Chains could be used in presampling. His instincts were dead on correct. Considering general classes of CTMCs moves us away from depending on funky particulars of a model, the approach treats a truly general class of problems. To be sure there are strong mathematical assumptions about these models (the classical "memoryless" property), but these same assumptions are the key to finding and exploiting lookahead. The details are technical but the key idea is this: For any given window of simulation time, it is possible for LPs (which necessarily are simulating interacting CTMCs) to generate schedules of simulation times at which they will synchronize with other LPs. The LPs exchange these schedules. Then an LP does its CTMC simulation up to one of these synchronization points. If it is an incoming synchronization it just waits for a message that either changes its state, or reports that the sender has reached this point in time and will not change the state (called a pseudo-event). In the former case the LP may adjust its state, cancel some future events and/or schedule new ones and then simulate up to the next synchronization point. In the latter case the LP is free to continue. In the

case of an outbound appointment, the LP flips a coin weighted by a function of the LP's state, and depending on the outcome either executes an event which changes its state and may affect another's (e.g., a job departure from an queue with infinite servers), or not. In the former case it reports the state change to the receiving LP, in the latter case it reports a pseudo-event. Here the underlying mathematical structure of CTMCs allows one to pre-sample crucial random variables and using these create opaque periods. Phil and I did a series of papers exploring this idea (Nicol and Heidelberger 1993, 1995) and achieved what for a brief shining moment must have been the largest real speedups observed (in the hundreds), certainly by a conservative technique.

In the 90's an idea of approaching synchronization through global windows of simulation time popped up in a number of contexts. The Wisconsin Wind Tunnel used it for multiprocessor emulation (Reinhardt et al. 1993), Jeff Steinman used it in his Breathing Time Buckets algorithm (Steinman 1993), and I used it in what has become known as the YAWNS (Yet Another Windowing Network Simulator) approach. There is a very powerful principle at work. If you can arrange to synchronize all LPs at a common point in time and the model semantics imply that they don't have to synchronize before that point, then you can escape entirely the overheads of polling and appointments. Efficient parallel algorithms for barrier synchronization exist. If you synchronize globally every ε units of time, and (critically) there is enough simulation workload to do in parallel every ε units of time, then the time required to perform the barrier is significantly less than the time needed to execute that workload, and global synchronization is a clear win. It was this intuition that enabled me to prove (Nicol 1993) that under certain technical conditions about a stochastic simulation, with a limited ability to pre-sample random variables, a window-based conservative synchronization algorithm gives performance that is optimal in an asymptotic sense, for under these conditions the amortized cost of computing the barrier goes to zero.

The relationship of workload and synchronization overhead is a crucial one, often overlooked in empirical research studies. In a conservative simulation, synchronization is something that happens at the *border* of an aggregated submodel, where submodels synchronize at the border. Intuitively, all other things being equal, if the model exhibits spatial locality and the model-to-processors assignment takes advantage of that locality, then the larger the area-to-perimeter ratio is, the less the synchronization overhead contributes to overall performance, and the more the load balance does. Speedup is not the only motivation for using parallel computers, indeed, in scientific computing the ability to increase the size of complexity of a model is as important. It makes sense then to evaluate conservative techniques in their most likely context of use.

In its most direct application, if the smallest minimum delay between send time and receive time between any two LPs is ε, one can synchronize every ε units of simulation time and be assured that no message sent within one synchronization window will be received in the same window, hence no other LP-LP synchronization is possible. However, this single parameter in what might be a huge model has tremendous impact on overall performance. On the other hand, eschewing barriers and sticking with point-to-point synchronization can run afoul of situations where an LP is connected to many other LPs, so that the overhead maintaining point-to-point synchronization is itself overwhelming. In response to these problems I constructed a hybrid solution called *composite synchronization* (Nicol and Liu 2002). The idea here is to choose a threshold ξ and classify all LP-to-LP linkages whose minimum delay is less than the threshold as *fast*, and all others as *slow*. We can use a global window synchronizing every ξ units of simulation time. Any interactions on synchronous channels can be managed at the barrier, while point-to-point appointments manage interactions across asynchronous channels. In practice we can dynamically and automatically change ξ in search of the performance sweet-spot. Just this year I wrote a thought piece on the relationship of load balance and synchronization overhead when using composite synchronization (Nicol 2017).

In the last ten years I have continued to work in parallel simulation, but as a unifying technology for model evaluations on testbeds comprised of real devices, simulations, and emulations. The observation is

for such a model to exhibit behavior similar to that in the field, one needs to embed all of these in virtual time. The gratifying thing to me is that the work we all did in synchronization is foundational, and now has application in domains we weren't thinking about all at first.

6 FROM PHOLD AND THE PADS CONFERENCE TO HLA (RICHARD FUJIMOTO)

I first stumbled upon the PDES synchronization problem when I was a doctoral student at the University of California in Berkeley (1978-1983), but didn't begin any serious work in the area until after I had graduated. Here, I'll recount some of my memories on three aspects of the field that are still visible today: the PHOLD benchmark and conservative/optimistic debates, the PADS conference, and the High Level Architecture. These developments largely occurred from 1987 to 1997, arguably, the "golden years" when the PDES research field formed and flourished.

6.1 PHOLD and the Conservative vs. Optimistic Debates

Stemming from my undergraduate interests in computer architecture, my doctoral dissertation at Berkeley focused on hardware for parallel computers, specifically switches (Reed and Fujimoto 1987). To evaluate our ideas about switches we needed parallel applications. It became clear to me then that the simulator I had developed to evaluate our switches would be a perfect benchmark program. I immediately bumped into the synchronization problem. Ideas akin to those developed by Chandy, Misra, and Bryant (published a few years earlier, but unknown to me) came to mind, but my main interest was in hardware design, so I didn't embark on any serious studies of the problem.

A few years later I renewed my interest in this area. I then came across Chandy and Misra's and other papers on conservative synchronization, as well as Jefferson's work on Time Warp. OK, so there are solutions to the synchronization problem, but no one knew which approach was better. So my attention focused on doing a serious comparison of these algorithms. Foremost in my thinking was the need to have a very efficient sequential simulation to determine the speedup obtained using parallel computing, which led to an exploration of the literature in event list implementations. A paper by Doug Jones comparing different priority queues for discrete event simulations caught my attention (Jones 1986). This comparison, and others that had appeared at the time, were all based on something called the HOLD model, which seemed to be the standard benchmark for evaluating sequential event list performance. So for better or worse, PHOLD (Parallel HOLD) was born, and became the application workload model I used in my comparisons of the Chandy/Misra/Bryant and Time Warp algorithms (Fujimoto 1988, 1990), and continues to be used to this day.

In retrospect, PHOLD has been both a blessing and a curse. On the one hand it addressed the same needs that I had at the time, namely, it was easy to implement and allowed parameterized control of a wide range of experiments. In addition to enabling comparison of PDES approaches, it also provided a way to examine PDES performance over the years, e.g., see (Barnes et al. 2013). However, PHOLD is a highly regular, well-structured benchmark that does not capture the irregularities found in many, perhaps most, parallel simulation applications. As such, probably the most that can be said for PHOLD performance results is PDES performance is not likely to be any better for real world applications.

The late 1980's and early 1990's were exciting times for the PDES community. The community of researchers was growing, and battle lines were being drawn between the "conservative" and "optimistic" camps. The conservative side was informally led by David Nicol and his former PhD advisor Paul Reynolds Jr., who in turn had been a PhD student working with Mani Chandy and Jay Misra. David Jefferson and Brian Unger were perhaps the most outspoken proponents of the optimistic approach. The conservative/optimistic debates were fervent with each side enthusiastically promoting their point of view. Such arguments are not uncommon in academia. In some cases competing researchers grow to become bitter lifelong enemies. I'm happy to say such was not the case within the PDES community as researchers grew to respect each other's work and hostilities have long since died out. There was no clear winner in the conservative vs. optimist debates; either approach demonstrated more favorable properties

depending on particulars of the application. Nevertheless the debates did serve the purpose of helping drive the technologies forward.

6.2 Time Warp Research

In my own work it was well recognized that two key challenges in Time Warp needed to be addressed. These were the time and memory need to perform state saving, and the potential overheads and wasted computation associated with rollback. In the state saving problem I saw an opportunity to return to my early interests in computer architecture. I proposed a type of memory management unit called the rollback chip to perform state saving in hardware (Fujimoto et al. 1992); this led to a proof-of-concept prototype (Buzzell et al. 1990), and subsequently, a parallel computer based on Time Warp called the Virtual Time Machine (Fujimoto 1989b). Subsequently, our research focused on a software-based approach that avoided state saving altogether, or at least as much as possible, using reverse computation. PhD students Chris Carothers and Kalyan Perumalla (Carothers et al. 1999) did much of this work, with Kalyan's doctoral research focusing on automating the generation of reverse computing code. Both continued to purse this research separately after leaving Georgia Tech.

Separate from my interest in hardware solutions, much of my early work focused on developing an efficient implementation of Time Warp. My early work focused on shared memory multiprocessors. Here, I developed a technique called direct cancellation that implemented anti-messages with a simple pointer (Fujimoto 1989a), leading to a very efficient implementation that yielded good speedup. We developed several other innovations including an efficient shared memory algorithm for computing Global Virtual Time (GVT) with PhD student Maria Hybinette (Fujimoto and Hybinette 1997) as well as an efficient, "lazy" approach to implement fossil collection that we called on-the-fly fossil collection. The initial Time Warp software that I had developed evolved into a system called Georgia Tech Time Warp (GTW) described in (Das et al. 1994; Fujimoto 2000). We realized that Time Warp anti-messages could be used to allow application programs to unscheduled events, a standard operation needed by discrete event simulation and PhD student Samir Das developed and evaluated this approach (Lomow et al. 1991); we later found out that Greg Lomow, a student working with Brian Unger at the University of Calgary developed a similar approach, but had not published his work as they were incorporating it into a commercial implementation of Time Warp, so we agreed to jointly published the work with Greg.

Driven by a desire to gain a better fundamental understanding of Time Warp, I worked with Ian Akyildiz and Dick Serfozo, colleagues at Georgia Tech who had expertise in performance modeling, as well as PhD students Anurag Gupta and Liang Chen to develop analytical models to predict Time Warp performance (Akyildiz et al. 1993; Gupta et al. 1991). Liang's theoretical work and experimental work by Samir Das (Das and Fujimoto 1997b) enabled us to gain insight into Time Warp performance when one limited the amount of memory allocated to the Time Warp execution using Jefferson's cancelback algorithm (Jefferson 1990). This led to Samir's doctoral work where he developed an algorithm to adaptively control the amount of memory allocated to Time Warp to maximize performance and reduce the amount of rolled back computation (Das and Fujimoto 1997a). Chris Carothers continued this work by developing adaptive Time Warp mechanisms to create load balancing methods on shared platforms (Carothers and Fujimoto 2000). Later work focused on grids (Park and Fujimoto 2008) and cloud computing environments (Malik et al. 2010).

GTW was used for a variety of applications, but perhaps the most memorable was its use to create fast air traffic control simulations. Fred Wieland developing an air traffic simulation called the Detailed Policy Assessment Tool (DPAT) on GTW that was deployed for air traffic analyses, making it at the time one of the few real-world deployments of Time Warp for a real-world application (Wieland 2001).

Finally, another branch of research focused on using Time Warp principles to accelerate *multiple* simulation runs. Maria Hybinette developed a parallel simulation cloning mechanism and showed its benefits on air traffic control simulations using DPAT (Hybinette and Fujimoto 2001), and Steve Ferenci

developed an approach called updateable simulations to use traces of prior runs to create new ones (Ferenci et al. 2002).

6.3 The PADS Conference

While the role of conferences in disseminating new developments is obvious, they also play a critical role in building research communities. This was particularly true for the PDES field, and PADS serving as its central meeting place. PADS was unique in that PDES was its primary focus, at least in the early years. It arguably retains the reputation as the principle outlet for PDES research to this day.

The first PADS conference was, in fact, not called "PADS". The conference was born in 1985 under the sponsorship of the *Society for Computer Simulation* (SCS), now known as the *Society for Modeling and Simulation Intl*. It began meeting annually in 1988 as a track in the SCS Western Multiconference. David Jefferson and Brian Unger organized the 1988 conference and at the end of the meeting, asked me if I would be interested in organizing the 1989 conference. I immediately accepted, and so began a long extended engagement with the PADS conference that lasts to this day.

Early on, many attendees came from a computer science background and did not have interest in the other tracks of the multiconference which were much more application focused. Many felt stronger affiliations toward IEEE and ACM rather than SCS, and the multiconferences were limited in terms of site selection, with many meetings held in the same hotel in San Diego. In response to these concerns, co-sponsorship with ACM and IEEE was obtained, and a management structure with a steering committee that included elected members as well as representation from the three societies was put in place. In 1993 ACM started the "Federated Computing Research Conference" (FCRC) format, a multi-conference including computer science focused conferences. This suited the PADS folks very well, so PADS left the SCS multi-conference for this venue. Ironically the 1993 FCRC meeting was held in San Diego, at exactly the same location where prior PADS meetings has been held! So much for changing locations. But the next ACM FCRC conference would not occur until 1996, raising questions about 1994 and 1995. A proposal was developed to hold PADS in conjunction with WSC, as a separate track, but the proposal hit a snag; PADS insisted on a separate proceedings and the WSC board wasn't willing to do that. That was a show-stopper. A bolder proposal was to run the 1994 meeting as a separate, stand-alone conference in Edinburgh Scotland. Back then, computer science conferences were not held as frequently in Europe as they are today, and with a new format for PADS, there was a certain amount of anxiety within the steering committee, or perhaps just me, but we decided to go that route.

The 1994 conference was a huge success. Our fears of sparse attendance evaporated in the long summer days of Scotland and we were able to get Internet pioneer Len Kleinrock to deliver the keynote talk. In hindsight, we made exactly the right decision for 1994, for several reasons, all unrecognized by us at the time. Making it a stand-alone conference gave PADS a clear identity. Establishing the conference in the UK brought in greater participation by researchers in Europe, where PDES interest was growing. The conference was now free to select venues that would provide the most benefit to the community, and the conference subsequently rotated among locations in Europe, Asia, and North America, and truly became an international meeting. Had the conference joined WSC, no doubt it would have fallen into the shadow of a much bigger conference, and might have disappeared.

Working on the PADS conference was a labor of love in many ways. But admittedly, it's a certain amount of work and worry, and after a while, one just gets tired. I remember one year my term on the steering committee was finally expiring, and I was happy to pass on the baton to another PADS enthusiast. I prepared to close what I thought would be my last PADS business meeting, held at the conference itself, with a big sigh of relief. Then there was an unexpected motion from the floor that I be named a lifetime member of the PADS steering committee! After a unanimous positive vote, I was back on the committee! How could I say no? As they say, no good deed ever goes unpunished!

Where did the name "PADS" come from? When the conference was being reorganized the name question arose. The first few conferences went under the name "distributed simulation." Wanting to

include "parallel" in the name to emphasize high performance computing, I began lobbying for the name "Parallel and Distributed Simulation." I remember arguing for this change with David Jefferson, who had other suggestions, but I stuck to my guns. Finally, David asked me why I was so adamant. I confessed I liked the acronym, PADS! That was the clincher, and we began using PADS in 1991 and it stuck.

6.4 The High Level Architecture and Pivoting to Research in Federated Simulations

In 1992 the ORSA Journal on Computing invited me to write a paper outlining the state-of-the-art and future directions for PDES research. In a brazen attempt to draw attention to the article, I picked what I thought would be a provocative title: "Parallel Discrete Event Simulation: Will the Field Survive?" (Fujimoto 1993). The issue also included commentaries by other leading researchers and my rejoinder. Based on the keynote I gave at PADS 1991 I largely lamented that PDES was marching forward, but was not being extensively adopted by practitioners, a common concern in the PDES community at the time.

Out of the blue, I received a phone call from Richard Weatherly of the MITRE Corporation. Richard had been working on a U.S. Department of Defense (DoD) project called the Aggregate Level Simulation Protocol that was tasked with interconnecting wargame simulations (Wilson and Weatherly 1994). The rage in DoD those days was on federating simulators and getting them to interoperate. Richard explained that his current project was something called the High Level Architecture, an effort led by Judith Dahmann of the Defense Modeling and Simulation Organization whose goal was to create a single, common architecture for *all* M&S in the DoD. In a heartbeat I was on a plane to Washington D.C.

The technical challenge was to define an approach to integrate simulations using different approaches to manage simulation time (time stepped, event driven, etc.). In the midst of the "conservative vs. optimistic" debate we needed an approach to allow conservative and optimistic parallel simulators to interoperate. Another challenge was to build consensus among the various stakeholders who had their own ideas of how things should be done. Many in the DoD M&S community were not familiar with PDES and had never heard of CMB or Time Warp. Fortunately I had some help. Part of Jade Simulations, a company founded by Brian Unger, was bought by Science Applications International Corporation (SAIC), a major defense contractor with a large stake in defense M&S and HLA. Brian's former students Darrin West and Larry Mellon were leading much of the HLA effort in SAIC and were well versed in PDES. With a small team representing various constituencies we managed to get consensus on the specifications. I recall working out many technical details on planes between Atlanta and DC where I was free from phone calls, emails (this was well before inflight wifi), visitors and students coming to my door.

To make a long story short, HLA was approved as the standard architecture for all M&S in the U.S. DoD in 1996, and was subsequently standardized by IEEE (IEEE Std 1516.2-2000 2000) and later updated (IEEE Std 1516.1-2010 2010). Richard Weatherly and Judy Dahmann deserve a tremendous amount of credit seeing the HLA effort through. Richard's team at MITRE were very capable developers and Judith quite adept at maneuvering though political difficulties to gain consensus. Judith also had a subtle sense of humor. The HLA effort had periodic meetings of the Architecture Management Group (AMG) at the DMSO headquarters in Alexandria Virginia, that included perhaps a hundred or so key stakeholders across the DoD M&S community. At one meeting, just before Christmas, Judith delivered a present to each attendee – a screw with an attached bolt. Her only comment at the meeting was that the interpretation was left up to us!

Subsequent to my HLA work my research program pivoted to focus on federated simulations. Federated simulations represented a way to quickly create parallel simulations from sequential code. I needed a new code base to accomplish this and spent the 1997-98 academic year on leave at the Defense Evaluation Research Agency in Malvern, England, developing high performance runtime infrastructure (RTI) software. This resulted in RTI-Kit that later evolved into the Federated Distributed Simulation Kit (FDK) (Fujimoto and Hoare 1998). Using this software then PhD student George Riley developed a parallel version of the NS2 network simulator, dubbed PDNS (parallel/distributed NS) (Riley et al. 2004). Through a DARPA-funded project we demonstrated the largest available discrete event network

simulations of the day using over a thousand processors of a supercomputer at the Pittsburgh Supercomputing Center (Fujimoto et al. 2003).

I continue to work on PDES to this day, both in developing the core technology and use of the underlying principles to create new simulation methods for various applications. With new platforms and new constraints such as the emergence of power consumption as a critical concern, many important research problems remain (Fujimoto 2016).

7 JADE INITIATIVES - RESEARCH & COMMERCIALIZATION (BRIAN UNGER)

John Cleary, David Jefferson, and I, along with several of our graduate students, created a spinout company from the University of Calgary called "JADE Simulations" in 1988. JADE was aimed at providing high performance simulation software and services to clients in the telecommunications and transportation industries. Figure 3 is suggestive of the enthusiasm surrounding this commercialization initiative and its core products based on Time Warp.

The company was the direct result of a five year research project within the Computer Science Department at the University of Calgary, and a subsequent synergistic collaboration with David Jefferson. Project JADE is described below followed by a brief history of JADE, the company, and some of its progeny.

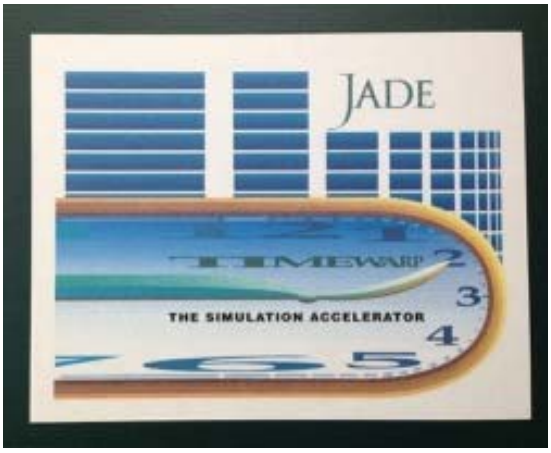

Figure 3. Jade Simulations 'Core Product'.

7.1 Project JADE (Just Another Distributed Environment) - 1982-1986

The JADE multilingual distributed software prototyping environment was developed at the University of Calgary through a National Science and Engineering Research Council of Canada "Strategic Grant" in the 1982 through 1986 period (Unger et al. 1984). The JADE Project contributed much of the software and hardware infrastructure, and technical expertise, which facilitated a complete implementation of Jefferson's Time Warp in less than one year, 1986.

Eight Computer Science faculty, 6 full time programming staff, and ~24 graduate students, all at the University of Calgary were initially involved in Project JADE. The core components of JADE were completed by 1985 when the environment was released to a number of participating research groups. This distributed software prototyping environment was then used to explore a number of application areas ranging from Distributed Prolog, VLSI simulation, ATM network simulations, monitoring and prototyping distributed systems, and the design, implementation and testing of PDES mechanisms.

7.1.1 Project JADE Major Components

The JADE Inter-Process Communication (JIPC, pronounced gypsy) 'synchronous' kernel (Cleary et al. 1985; Neal et al. 1984; Xiao et al. 1987) supported the execution of a 'distributed program' whose components were implemented in ADA, C, LISP PROLOG or SIMULA. The execution of these components gave rise to corresponding Logical Processes (LPs) that interacted solely through JIPC messages. JIPC supported such execution on networks of workstations, on a Mesh machine, and on a few shared memory Multiprocessors. Supporting 'interpreted' components (i.e. Lisp, Prolog) enabled pausing execution, modifying a component, and continuing execution.

The JADE interactive graphical user interface consisted of three major components: the Jaggies graphics system, the Jagged graphics editor, and a bit-mapped workstation Window System. The objectives of Jaggies and Jagged were to support illustrations and displays of process execution with real-time animation. Jagged was an interactive program, built on top of Jaggies, that assists users in building static pictures. The Jaggies pictures thus created can then be animated by distributed programs that send JIPC messages to Jaggies. Jagged is described in (Dewar and Unger 1984; Unger et al. 1986a, b). The Jade Window System predated commercial window systems enabling a user to interact with a number of simultaneously executing LPs. Multiple views were thus provided into an executing distributed system.

Prototyping is accomplished by supporting the implementation, testing, and performance evaluation of an embedded distributed program that interacts with simulation models of devices which exist in the target system (Lomow and Unger 1985). Mechanisms for monitoring distributed system execution that support this prototyping include dealing with non-determinism and the impact on system behavior caused by monitoring itself (Joyce et al. 1987; Joyce and Unger 1985). The JADE extensible monitoring system collects LP interaction information via unmonitored JIPC messages.

7.1.2 Project JADE Achievements

The significance of Project JADE achievements is reflected in our refereed publications. The Jade research group has published over 45 journal papers, 76 full conference papers, 34 technical reports, and 10 books and films. Graduate students involved in the project completed 35 M.Sc. and 8 Ph.D. degrees.

7.1.3 Project JADE Exploration of Time Warp & PDES

In 1985 several core researchers involved in Project JADE became aware of David Jefferson's work at UCLA and JPL on Time Warp. Given our interests in both simulation and distributed systems we enthusiastically met with David and his JPL team and studied Time Warp and Virtual Time. At that time there still was no full implementation of Time Warp though the JPL version was well underway. It seemed a natural fit to build Time Warp on extensions of JIPC, initially called TIPC. Basically, TIPC (pronounced typsy) added time advance primitives and state saving and restoration functions to JIPC.

David Jefferson, John Cleary, Brian Unger and Xiao Zhonge, along with several graduate students, collaborated on a JADE implementation of Time Warp. This design dealt with several Time Warp performance issues such as efficient state saving, aggressive versus lazy rollback, model level design issues (e.g., process view versus event view, model decomposition and load balancing), avoiding rollback storms, flow control and global storage management. Xiao completed an implementation of our resulting design in 1986. In 1986 and 1987 two graduate students completed theses that embodied several further optimizations of Time Warp. This PDES technology and expertise motivated the creation of a company.

7.2 JADE Simulations International Ltd 1988 - 1993

The company JADE Simulations International (or just Jade Simulations or Jade) was created in 1988 by Brian Unger (CEO), John Cleary (CTO), Greg Kletke (CFO) and David Jefferson (Board) with graduate students Greg Lomow, Larry Mellon and Darrin West. We had two angel investors, Robert Olson and Hammie Hill, who along with personal funds from the founders, launched the company with a total startup capital of ~$200,000.

Over the next 3-4 years Jade received several venture capital investments, each about $1 million. We were fortunate in being able to recruit Thomas Csathy as our Board Chair, formerly President of several companies including Burroughs (Unysis) and Cognos, and past Vice President of IBM Canada. The CEO of venture capital company Alta-Can Telecom, Mike Raymont, was also a key Board member.

7.2.1 Jade Products

Darrin West's partial implementation of Time Warp formed the kernel of Jade's time synchronization mechanism. The re-implemented TIPC supported deterministic execution and several performance optimizations including lazy rollback, lazy re-evaluation (Baezner et al. 1994) and cancelback. Metrics were defined which using the Project Jade distributed monitoring system enabled making many runtime problems visible such as load balancing, rollback storms, model design and decomposition weaknesses. Explicit, deterministic ordering to message timestamps was a later important addition (Cleary and West, 1998). The visualization tools supported validation and performance tuning, global virtual time dynamics, rollback, memory usage and recovery (Dewar and Unger 1984, Joyce, Lomow et al. 1987).

A programming language, Sim++ (Baezner et al. 1990; Gomes et al. 1995), was developed that provided either an event view or process view for simulation model design. Sim++ also provided both an efficient sequential execution kernel and a Time Warp optimistic PDES kernel for several platforms. Except for limitations placed on the use of shared and global memory, Sim++ programmers had access to all of the language facilities of C++, including user-defined data types, dynamic memory allocation and deallocation, multiple inheritance, dynamic binding, and polymorphism. Sim++ and the two kernels were supported on several hardware platforms, the BBN Butterfly, the Meiko Transputer based Mesh Machine, and networks of Unix workstations.

7.2.2 Applications, Target Markets, Successes and Winding Down

Our initial target market was telecom network simulation. We had some initial success with ATM network and telecom signaling simulations (Unger et al. 2000). We spent some time on VLSI modeling and simulation, in particular, a version of Verilog built on Sim++ and Time Warp. Although this continued to be an area where performance was at a premium, and thus parallel execution attractive, we were not successful in completing a viable product. Another fascinating area in which we spent time and energy was parallel and distributed Prolog (Cleary et al. 1988). One new recruit, Jim Inkster, led our Ottawa office and had some experience in air traffic control applications where we had significant success.

By 1992-93 Jade had about 30 employees with offices in Calgary and Ottawa with a wholly owned USA subsidiary company (Jade Simulations International Inc.) in Washington D. C. Headquarters and R&D continued to be based in Calgary. Jade had closed significant contracts including: a joint Jade and CAE (www.cae.com) $37 million air traffic control simulation project with the Canadian Department of Transport; a continuing contract with Stentor and Bell Canada for telecom trunk network and signaling simulations; a high fidelity network simulator for the Naval Research Laboratory in Washington DC; a Motorola contract for simulating parts of their Iridium satellite telecom system; and a telecom signaling network model for Ameritech Services of Chicago.

Nevertheless, achieving cost-effective simulation speedups via Time Warp on the hardware platforms available in the late 1980s and early 1990s was rarely possible. Achievable speedups on tens of nodes could not justify the increased complexity of model design and performance tuning needed, particularly where state saving overheads with fine grained applications required much larger platforms to reach a substantial cost advantage. Further, fine grained applications such as VLSI simulations with static communication graphs could more cost effectively be served by conservative mechanisms such as CMB. Only large military applications, with fatter, coarse grained, events and dynamic communication graphs were deemed highly economically attractive in the early 1990s. Although Jade had acquired Canadian secret clearances, and the Jade USA subsidiary had been created to support these applications, our Canadian origins and continued Canadian corporate control were a substantial barrier to securing USA military contracts.

After five years of trying to build a growing profitable business in PDES our investors concluded Jade was ahead of its time and that it was in their interest to wind down the company recovering funds

through selling off parts of the company where possible. Thus the Jade USA subsidiary and technology were sold to Science Applications International Corporation (SAIC) (www.saic.com). At that time, several Jade employees were also hired by SAIC. The Canadian parts of Jade continued to operate, mainly satisfying its current contracts, while winding down. Brian Unger went back to the University of Calgary and John Cleary went back to his New Zealand home and Waikato University.

7.3 Jade Progeny – A Selection: SAIC, Games, Finance 1993 - 2017

When SAIC bought Jade's USA subsidiary and its PDES technology in 1993, several (~4) Jade employees also went to SAIC including two of Jade's founders: Darrin West and Larry Mellon. They both worked at SAIC until 2000 and most of the information in this section comes from them, and from Greg Lomow, who also left Jade in 1993 to head for Wall Street and financial applications.

A new C++ Time Warp implementation was built at SAIC using a shared memory backplane on the SGI Origin, and a Cray hypercube machine, as well as, on a distributed shared memory backplane. This was called Thema, and was used in a number of DARPA contracts for military training and systems simulation. During the development of the DoD HLA, the team at SAIC helped surface issues related to Time Warp, and its unique time management and fossil collections issues so that the specification could make use of PDES techniques. These efforts led to a number of new discoveries, related to cancelback, memory optimization, and memory hardware caching side effects. SAIC clients that benefitted from this PDES technology included: NRL, HPSS and DARPA.

From SAIC both Darrin and Larry moved into the computer games world in 2000, particularly massive multiplayer distributed games. They worked at a number of companies including: Orcus, Maxis / Electronic Arts (The Sims Online), The Amazing Society (Superhero Squad Online) and Area 52 Games. During 2000-2017 they also started their own companies: Maggot Ranch and Emergent Game Technologies. All of these products tapped into distributed simulation in some form. Darrin and Larry were instrumental in developing Advanced Distributed Simulations (ADS), initially in work with NRL, which introduced "Interest Management". IM (a generalization of sectorization) added publish/subscribe mechanisms by category, e.g. geographical grid cells. This technology was useful in multiplayer games to provide scalability. First Person Shooters tended to use broadcast techniques, but massively multiplayer games like WOW needed interest management to handle thousands of players and Entities.

Being real time applications, Time Warp was rarely applicable. But they did use a state save mechanism to ensure remote clients started from the same conditions. The fundamental notions of the Virtual Time paradigm with time stamped messages and communicating logical processes was used repeatedly as a means to address or correct or hide message ordering issues and deal with load balancing as entities and computation needed to move between servers. Game Entities are very much LPs. They interact with nearby Entities, and need to be mapped to processors that contain others they tightly communicate with. Migration is a big deal and message routing is a challenge. There can be thousands of Entities so there is always a desire for parallel execution. One unique challenge is that game developers think in time steps, because they need to render a graphics frame with up to date positions and states of Entities. But it is more efficient to model things using discrete events. The tension between the two paradigms, time stepped or event based, is tough in the games world.

Subsequent to Jade, Greg Lomow worked on Wall Street building trading and asset management systems, and later in Oregon on billing and payment systems, and generally on large scale business systems. He reports in 2017: "Pretty much everything I've been working on since 1993 is related to the distributed, near-real time systems work we did in PDES, yet nothing directly related to PDES".

8 PDES TOOLS & COMMERCIALIZATION (RAJIVE BAGRODIA)

I was fortunate to work with Professors Mani Chandy and Jayadev Misra as my PhD Thesis advisors at the University of Texas at Austin. Immediately after graduation, I joined UCLA as an Assistant Professor

of Computer Science. So, no surprise that my early PDES research was aimed at developing *hybrid* simulation algorithms and tools that spanned both conservative and optimistic domains.

8.1 PDES Tools Research

The 80s and particularly 90s were a period of rapid innovation in parallel computing with the advent of many new parallel machines and parallel programming systems. For PDES researchers, this provided a unique opportunity to harness the advances in parallel computing to make PDES technology applicable beyond the queuing network models that were widely used in the early days of PDES research. My own research focused on developing general-purpose parallel simulation languages and systems that could be used to develop (parallel) simulation models across diverse areas, where much of the esoteric aspects of PDES could be kept 'under the covers'.

One of the early attempts by my research group at UCLA to achieve this goal was the design of Maisie (Liao and Bagrodia 1994); message-passing had emerged as a useful paradigm for parallel computing and Maisie demonstrated an easy extensibility of that model to provide PDES implementations that could be developed in a transparent manner (Bagrodia et al. 1987). The early concepts of Maisie were refined to develop PARSEC (Bagrodia et al. 1998), which was used for parallel simulation of a number of diverse applications ranging from gate-level circuit simulations, parallel programs, database systems, and wireless networks. However, PARSEC cast too wide a net – design of a language that could be used for discrete-event simulation of any application, using multiple underlying time synchronization protocols that included pure conservative, pure optimistic, and the afore-mentioned hybrid (Jha and Bagrodia 1994) ones, on both shared-memory and message-passing families of parallel architectures. Although conceptually feasible, this attempt at universality presented a major challenge for keeping the critical PDES technology 'under the hood' – PDES implementations were finicky and extracting performance gains on the supercomputers of that era, depended on a myriad of factors that required specific application characteristics to be appropriately leveraged by the underlying time synchronization algorithm and perhaps even the parallel programing architecture.

The next step in the evolution of my group's research was to focus on specific application areas and implementations of time synchronization algorithms that could be optimized transparently to exploit application characteristics. Around this time, DARPA initiated the GloMo or Global Mobile Information Systems program, to develop technology to leverage wireless and mobile computing for application in the digitized battlefield. The major objective of UCLA's DOMAINS project that was funded under this program was: "*The GloMo environment consists of a rapidly changing infrastructure of interconnected heterogeneous networks in a (possibly) hostile setting that must support a large number of mobile users whose multimedia traffic requirements are severe, critical, and real-time. [..] This proposal describes an integrated approach to the solution of this problem through the development of Intelligent Agent technology in conjunction with a scaleable simulation capability called GloMoSim*" (Bagrodia et al. 1997). It was expected that GloMoSim would scale to simulate networks with tens of thousands of mobile devices, the scale of the operational military networks envisaged by then GloMo Program Manager – Mr. Rob Ruth; the state of the art then was about 50-100 such devices. Over the next few years, we designed GloMoSim (Zeng et al. 1998) by leveraging and further advancing the PDES technology used in PARSEC. GloMoSim successfully used PDES to not only scale to high-fidelity simulations of networks with over a thousand mobile radios, where the entire protocol stack was modeled for each radio, but also achieved the ability to run large scale models faster than 'real-time', i.e., the execution took less wall clock time than the simulation horizon. Although initially envisaged as a system that would leverage both conservative and optimistic algorithms, GloMoSim eventually used only the conservative family of PDES algorithms (Misra 1986). A primary aim in our design was to 'hide' the lookahead capability within the lower layers of the network models, where it was most effective, thus allowing networking researchers with no knowledge of PDES to develop models. The viability of this approach was demonstrated by the

variety of models, specifically for mobile ad hoc networking (MANET) protocols that were developed by wireless networking researchers who had no familiarity with PDES.

Another significant aspect of this research was the use of PDES to improve model execution time such that high-fidelity models did not 'just run faster', but could meet the strict deadline imposed by 'real-time'; thus PDES systems could be applied to significantly expand the reach of live, virtual, constructive (LVC) simulations. One of our early use of PDES technology in LVC systems was presented in TWINE (Zhou et al. 2006), a novel framework that combined simulation, emulation, and live networks in an integrated testbed for adaptive wireless systems. The integration of PDES technology in the TWINE framework enabled the evaluation of real applications (e.g. streaming video and Voice Over IP) and protocol implementations over a wide range of wireless networking scenarios *while achieving scalability* of the overall testbed. These two research thrusts – network simulation and LVC models –set the stage for the next chapter in my personal PDES journey.

8.2 Commercialization

In the late 90s and early 2000, the start-up bug consuming the technology sector had begun to spread to many in the research community, and I was certainly not immune to its siren song! The relative success of GloMoSim within the research community and the spreading mania on the promise of wireless communications for the Internet economy was too much to resist. I founded Scalable Network Technologies around the turn of the century with the hope that users were not only interested in 'free' scalable simulators but would actually be willing to spend money to use that technology to solve problems in the 'real world'. At about that time, DARPA also launched its big initiative towards digitizing battlefield communications -- the Future Combat System (FCS) was the next big thing for the US Army. Although Scalable started as a typical garage start up, in our very early stages, we were fortunate to obtain funding from a variety of DoD sources. These included DoD SBIR projects as well as the DARPA FCS Communications program, managed by Dr. Jim Freebersyser. The external funding helped us to continue our efforts to develop a commercial product while simultaneously using the initial prototype internally to demonstrate feasibility of MANET technology for FCS systems. Our first Commercial-Off-The-Shelf (COTS) product offering was QualNet (Scalable Network Technologies 2001), launched as a commercial derivative of GloMoSim. The network simulation market was then dominated by OPNET, a commercial network simulator provided by an established public company -- OPNET Technologies.

Scalable's early customers were primarily interested in the ability to run high-fidelity models of large heterogeneous networks leveraging MANET protocols. As MANET protocols were themselves an innovative concept, it was unclear that they could scale effectively to large networks operating potentially in harsh environments while supporting both real-time and data traffic. This was an ideal use case for PDES as such simulations generated a large number of events. Sequential implementations of these models ran substantially slower than real time for even small networks with well under 100 radios, while the DoD was interested in using this technology to equip brigades and larger deployments where the radio count was expected to exceed many thousands.

The primary value proposition for Scalable, from the very beginning, was to execute high-fidelity, large scale network simulations at or faster than real-time using PDES technology and the rapidly evolving parallel architectures of the day. It was this aspect of QualNet and the difficulty for OPNET to effectively exploit PDES model execution (Wu et al. 2001) that allowed us to carve out our niche. While our use of PDES was clearly what powered QualNet's speed and scalability, the development team went to great lengths to 'hide' the PDES-related attributes within the kernel such that an average user would not need to learn about 'logical processes', 'model partitioning', or 'lookahead and null messages'!

Within a few years of the company being launched, Scalable had sold commercial licenses to large defense primes (e.g., Boeing), leading Fortune 500 companies (e.g., Microsoft and NTT DoCoMo), DoD agencies (e.g., Space and Warfare System Centre - Pacific) and established a niche market for scalable

and high-fidelity network simulations. At this point, it would be appropriate to confess that the Scalable team battled through a variety of technical, financial, and organizational obstacles to stay afloat, beyond just the challenge of maintaining efficient and transparent implementations of cutting edge PDES and real-time synchronization algorithms within the QualNet kernel. Perhaps, the most difficult challenge was that OPNET was then firmly embedded within the DoD as the de facto standard for network simulations. It provided a robust network simulator with an advanced user interface and the user community had made significant investments into their software, training and model libraries; switching costs from OPNET were substantial and only the most adventurous users were prepared to look elsewhere. Thankfully, there were enough such users, including Dr. Albert Legaspi from the US Navy, one of the earliest adopters of PDES-based network simulations within the DoD.

A significant early user of QualNet, was The Boeing Co. in its role as the prime contractor on the FCS program. The Army used force-on-force simulators widely, but they had a limitation in that the simulators assumed 'perfect communications' among the participating entities, i.e., all messages from any source to any destination were received instantaneously. The FCS program was expected to use 'on-the-move' communications for data and multi-media communications; as communication latency and jitter could significantly impact mission success, it became important for the force-on-force simulators to incorporate the realistic impact of 'imperfect' communications on mission success. This required the development of a Communication Effects Server (CES). Scalable in partnership with the US Army Communications and Electronics Command won its bid to develop the CES using QualNet as the underlying simulator (Bagrodia et al. 2006; Doshi et al. 2006). This proved to be a 'big break' for our fledgling company as we developed the FCS CES over the next few years, under a program managed by Mr. Steve Goldman and Dr. Dilip Kumar at Boeing and Mr. Kent Pickett for the FCS Program.

Simultaneously, Scalable had invested significant IR&D resources in the development of its next product, EXata (Scalable Network Technologies 2008), which provided a PDES kernel and set of external interfaces to synchronize simulations running in 'real time' with live network hardware (e.g. routers and radios), network software (e.g. third party network managers and network monitoring software), and applications (e.g., Apache web server hosted on mobile platforms). EXata allowed us to substantially broaden the market for PDES-based network simulators from its initial use for 'off line' simulations to novel uses for test and training of net-centric systems. By exploiting EXata's ability to embed high-fidelity models within other training simulators or interfaced with test technologies, we were able to use PDES powered network models in multiple real-time contexts. The CES subsequently morphed into the Joint Network Emulator (JNE), which used EXata as the underlying network simulator and widened its applicability into diverse areas of network test, training, and cyber resilience (Williamson et al. 2012). JNE is widely used today across the US DoD services and US and European defense prime contractor community for the simulation of large scale tactical communication systems and more importantly, to embed such real-time simulations in LVC networks used to assess mission performance with actual mission command applications. More recently, JNE also incorporated an extensive library of cyber threat and defense models that can be used to assess the cyber resilience of battlefield networks and operational mission threads (Varshney et al. 2011). This library, as all the other models within JNE, maintained their scalability by exploiting the conservative PDES algorithms.

Among many uses to solve real military networking issues, we mention the use of JNE by the US Army Operational Test Command, where JNE was used to model the Joint Tactical Radio System (JTRS), and the model was interfaced with the live JTRS Enterprise Network Manager (JENM). Prior to the availability of live JTRS radio units, JNE was often used to both demonstrate JENM functionality as well as to assess its scalability under diverse operational conditions. For many of these applications, JNE used PDES implementations on multi-core parallel architectures to provide the needed execution speed and scalability. JNE was also incorporated within a larger US Army test suite called BCNIS (DiGennaro et al. 2009) which was used to simulate large JTRS networks, but more importantly to interface and stimulate *live networks* for developmental (DT) and operational testing (OT) by the US

Army. In particular, BCNIS successfully supported Joint Tactical Radio System – Ground Mobile Radio (JTRS-GMR) DT and OT events as well as the Network Integration Evaluation (NIE) 11.2, among the Army's largest Operational Test event. In one such event, a network of tens of GMR radios emulated in BCNIS, were interfaced with a handful of live GMR units, allowing a test to scale up to a configuration that would otherwise have been impossible and/or cost-prohibitive, given that GMR units were then being produced in extremely small numbers.

A second use case that provided even more dramatic savings for a test and analysis application used JNE together with other simulators to run a distributed LVC exercise with over 550 tactical radio emulations with realistic traffic loading to represent a brigade operations and fires exercise (Bucher and Bouwens 2013). This distributed emulation included a tactical network architecture with satellite and Link-16 networks, terrestrial networks of JTRS radios augmented with aerial tiers and unique radio/router architectures at platoon and below level, superimposed over a Warfighter Information Network-Tactical (WIN-T) Increment 1 satellite backbone, and tightly coupled with operational maneuver of forces simulated in OneSAF. If live systems were used, the US Army led team estimated that this test would have cost $800 Million; use of the LVC models allowed this test to be completed for a total cost of $3.6Million!

The most recent example of the continuous broadening of the applicability of PDES is the use of EXata and JNE for cyber resilience assessment of communication networks that include DoD weapon systems. The Cyber TASE (Test Analysis and Simulation Environment) project, led by the US Navy, is developing an LVC capability to assess the cyber vulnerabilities and resilience of DoD enterprise systems (Petrie 2015). The constructive component of Cyber TASE was developed around StealthNet, a distributed cyber M&S capability that uses PDES as a core technology to ensure that large scale models of networks under attack from a diverse set of cyber threats can be executed in near real-time. I am comfortable that what we are witnessing is but the dawn of the cyber M&S area and commercial applications of PDES for assessing cyber resilience of a diverse set of network-centric systems provides a significant opportunity for future growth.

9 CONCLUDING REMARKS

With this paper we have attempted to provide some insight into the events and activities leading to the creation, evolution, and growth of the parallel discrete event simulation field, especially in its formative years in the last quarter of the 20th century. Driven by advances in underlying hardware and software technologies on the one hand, and the needs of increasingly sophisticated simulations on the other, the field has flourished from early problem formulations and innovative solutions, to development and maturation of the technology, to commercialization and application to real world problems. The field remains an active and vibrant area of research and development and we expect will continue to enjoy great interest and innovation in the years ahead.

ACKNOWLEDGMENTS

We thank Robert Sargent, a PDES contributor himself, who organized the history of simulation track for the Winter Simulation Conference, and invited and supported the writing of this paper.

Any presentation such as this necessarily ignores the contributions of many men and women who helped create, evolve, and grow the field. Many of these individuals continue to develop PDES technology and applications to this day. We acknowledge and thank these individuals for their numerous and varied efforts.

REFERENCES

Akyildiz, I. F., L. Chen, S. R. Das, R. M. Fujimoto, and R. Serfozo. 1993. "The Effect of Memory Capacity on Time Warp Performance." *Journal of Parallel and Distributed Computing* 18 (4):411-422.

Baezner, D., G. Lomow, and B. W. Unger. 1990. "Sim++ : The Transition to Distributed Simulation." In *SCS Conference on Distributed Simulation*, 211-218. SCS Simulation Series.

Baezner, D., G. Lomow, and B. W. Unger. 1994. "A Parallel Simulation Environment Based on Time Warp." *International Journal in Computer Simulation* 4 (2):183-207.

Bagrodia, R., K. M. Chandy, and J. Misra. 1987. "A Message Based Approach to Discrete Event Simulation." *IEEE Transactions on Software Engineering* SE-13 (6).

Bagrodia, R., L. Kleinrock, G. Popek, M. Gerla, L, Zhang, and P. Reiher. 1997. "DOMAINS: Design of Mobile Adaptive Networks Using Simulation and Agent Technology**."** DARPA Contract No. DAAB07-97-C-D321, Computer Science Department, UCLA.

Bagrodia, R., R. Meyer, B. Park, H. Song, Y. Chen, X. Zeng, M.Takai, and J. Martin. 1998. "Parsec: A Parallel Simulation Environment for Complex Systems." *IEEE Computer Magazine*.

Bagrodia, R., K. Tang, S. Goldman, and D. Kumar. 2006. "An Accurate Scalable Communication Effects Server for the Fcs System of Systems Simulation Environment." In *Proceedings of the Winter Simulation Conference*, edited by D. Nicol, R. Fujimoto, B. Lawson, J. Liu, F. Perrone, F. Wieland, 1226-1233, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barnes, P. D., C. D. Carothers, D. R. Jefferson, and J. M. LaPre. 2013. "Warp Speed: Executing Time Warp on 1,966,080 Cores." In *Principles of Advanced Discrete Simulation*, 327-336.

Bryant, R. E. 1977a. "Simulation of Packet Communication Architecture Computer Systems." M.S. thesis, MIT-LCS-TR-188, Computer Science Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Bryant, R. E. 1977b. "Simulation of Packet Communications Architecture Computer Systems." In *Mit-Lcs-Tr-188*.

Bryant, R. E. 1979. "Simulation on a Distributed System." In *First International Conference on Distributed Computing Systems*, 544-552. IEEE Computer Society.

Bucher, N., and C. Bouwens. 2013. "Always on-on Demand: Supporting the Development, Test, and Training of Operational Networks & Net-Centric Systems." In *NDIA 16th Annual Systems Engineering Conference.*

Buzzell, C. A., R. M. Fujimoto, and M. J. Robb. 1990. "Modular VME Rollback Hardware for Time Warp." In *SCS Distributed Simulation Conference*, 153-156.

Carothers, C., K. S. Perumalla, and R. M. Fujimoto. 1999. "Efficient Optimistic Parallel Simulations Using Reverse Computation." *ACM Transactions on Modeling and Computer Simulation* 9 (3):224-253.

Carothers, C. D., and R. M. Fujimoto. 2000. "Efficient Execution of Time Warp Programs on Heterogeneous, Now Platforms." *IEEE Transactions on Parallel and Distributed Systems* 11 (3):299-317.

Chandy, K., and J. Misra. 1981. "Asynchronous Distributed Simulation Via a Sequence of Parallel Computations." In *Communications of the Acm*.

Chandy, K. M., and J. Misra. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Transactions on Software Engineering* SE-5 (5):440-452.

Cleary, J., B. W. Unger, and X. Li. 1988. "A Distributed and-Parallel Backtracking Algorithm Using Virtual Time." In *SCS Conference on Distributed Simulation*, 177-182. SCS Simulation Series.

Cleary, J. G., G. A. Lomow, B. W. Unger, and Z. Xiao. 1985. "Jade's IPC Kernel for Distributed Simulation." In *Association of Simula Users Conference.*

Das, S., R. M. Fujimoto, K. Panesar, D. Allison, and M. Hybinette. 1994. "GTW: A Time Warp System for Shared Memory Multiprocessors." In *Proceedings of the 1994 Winter Simulation Conference*, edited by D. Sadowski, A. Seila, J. Tew, S. Manivannan, 1332-1339, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Das, S. R., and R. M. Fujimoto. 1997a. "Adaptive Memory Management and Optimism Control in Time Warp." *ACM Transactions on Modeling and Computer Simulation* 7 (2):239-271.

Das, S. R., and R. M. Fujimoto. 1997b. "An Empirical Evaluation of Performance-Memory Tradeoffs in Time Warp." *IEEE Transactions on Parallel and Distributed Systems* 8 (2):210-224.

Dennis, J., and D. P. Misunas. 1974. "A Preliminary Architecture for a Basic Data-Flow Processor." In *ACM SIGARCH Computer Architecture News*, 126-132. ACM.

Dennis, J. B. 1975. "Packet Communication Architecture." In *Sagamore Computer Conference on Parallel Processing*, IEEE Computer Society.

Dewar, A., and B. W. Unger. 1984. "Graphical Tracing and Debugging of Simulations." In *Conference on Simulation in Strongly Typed Languages*, 68-73. SCS Simulation Series.

DiGennaro, M., O. Walker, S. Doshi, B. Bressler, and R. Bagrodia. 2009. "Bcnis: Use of Live Virtual & Constructive (Lvc) Technology for Large Scale Operational Tests of Net-Centric Systems." In *Military Communications Conference (MILCOM).*

Doshi, S., U. Lee, and R. Bagrodia. 2006. "Wireless Network Testing and Evaluation Using Real-Time Emulation." *ITEA Journal*.

Ferenci, S., R. Fujimoto, M. H. Ammar, and K. Perumalla. 2002. "Updateable Simulation of Communication Networks." *16th Workshop on Parallel and Distributed Simulation*, May 2000.

Fujimoto, R. 2000. *Parallel and Distributed Simulation Systems*: Wiley Interscience

Fujimoto, R. M. 1988. "Performance Measurements of Distributed Simulation Strategies." In *Distributed Simulation*, 14-20. SCS.

Fujimoto, R. M. 1989a. "Time Warp on a Shared Memory Multiprocessor." *Transactions of the Society for Computer Simulation* 6 (3):211-239.

Fujimoto, R. M. 1989b. "The Virtual Time Machine." In *International Symposium on Parallel Algorithms and Architectures*, 199-208.

Fujimoto, R. M. 1990. "Performance of Time Warp under Synthetic Workloads." In *Proceedings of the Scs Multiconference on Distributed Simulation*, 23-28.

Fujimoto, R. M. 1993. "Parallel Discrete Event Simulation: Will the Field Survive?" *ORSA Journal on Computing* 5 (3):213-230.

Fujimoto, R. M. 2016. "Research Challenges in Parallel and Distributed Simulation." *ACM Transactions on Modeling and Computer Simulation* 24 (4).

Fujimoto, R. M., and P. Hoare. 1998. "HLA RTI Performance in High Speed Lan Environments." In *Proceedings of the Fall Simulation Interoperability Workshop*. Orlando, FL.

Fujimoto, R. M., and M. Hybinette. 1997. "Computing Global Virtual Time in Shared Memory Multiprocessors." *ACM Transactions on Modeling and Computer Simulation* 7 (4):425-446.

Fujimoto, R. M., K. S. Perumalla, A. Park, H. Wu, M. Ammar, and G. F. Riley. 2003. "Large-Scale Network Simulation -- How Big? How Fast?" *Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2003/10/01.

Fujimoto, R. M., J. J. Tsai, and G. C. Gopalakrishnan. 1992. "Design and Evaluation of the Rollback Chip: Special Purpose Hardware for Time Warp,." *IEEE Transactions on Computers,* 41 (1):68-82.

Gafni, A. 1985. "Space Management and Cancellation Mechanisms for Time Warp." Ph.D. Dissertation, Dept. of Computer Science, University of Southern California, Los Angeles, CA USA.

Gomes, F., J. Cleary, A. Covington, S. Franks, B. W. Unger, and Z. Xiao. 1995. "Simkit: A High Performance Logical Process Simulation Class Library in C++." In *Proceedings of the Winter Simulation Conference*, edited by W. Lilegdon, D. Goldsman, C. Alexopoulos, K. Kang, 706-713, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gupta, A., I. F. Akyildiz, and R. M. Fujimoto. 1991. "Performance Analysis of Time Warp with Multiple Homogeneous Processors." *IEEE Transactions on Software Engineering* 17 (10):1013-1027.

Hoare, C. A. R. 1978. "Communicating Sequential Processes." *Communications of the ACM* 21 (8):666–677.

Hybinette, M., and R. M. Fujimoto. 2001. "Cloning Parallel Simulations." *ACM Transactions on Modeling and Computer Simulation* 11 (2):378-407.

IEEE Std 1516.1-2010. 2010. Ieee Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) -- Interface Specification. New York, NY: Institute of Electrical and Electronics Engineers, Inc.

IEEE Std 1516.2-2000. 2000. *Ieee Standard for Modeling and Simulation (M&S) High Level Architecture (Hla) -- Object Model Template (Omt) Specification*. New York, NY: Institute of Electrical and Electronics Engineers, Inc.

IEEE Std 1516.3-2000. 2000. *Ieee Standard for Modeling and Simulation (M&S) High Level Architecture (Hla) -- Interface Specification*. New York, NY: Institute of Electrical and Electronics Engineers, Inc.

Jefferson, D. 1985. "Virtual Time." *ACM Transactions on Programming Languages and Systems* 7 (3):404-425.

Jefferson, D., B. Beckman, S. Hughes, E. Levy, T. Litwin, J. Spagnuolo, J. Vavrus, F. Wieland, and B. Zimmerman. 1985. "Implementation of Time Warp on the Caltech Hypercube." In *Society for Computer Simulation Conference on Distributed Simulation*, 24-26.

Jefferson, D., B. Beckman, F. Wieland, L. Blume, M. DiLoreto, P. Hontalas, P. Laroche, K. Sturdevant, J. Tupman, V. Warren, J. Wedel, H. Younger, and S. Bellenot. 1987. "Distributed Simulation and the Time Warp Operating System." In *11th Symposium on Operating Systems Principles.*

Jefferson, D., and H. Sowizral. 1982. "Fast Concurrent Simulation Using the Time Warp Method, Part I: Local Control." The Rand Corporation, Santa Monica, California USA.

Jefferson, D. R. 1990. "Virtual Time II: Storage Management in Distributed Simulation." In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 75-89.

Jha, V., and R. Bagrodia. 1994. "A Unified Framework for Conservative and Optimistic Distributed Simulation." In *Workshop on Parallel and Distributed Simulations*, edited by R. B. D.K. Arvind, and J. Y-B. Lin, 12-19.

Jones, D. W. 1986. "An Empirical Comparison of Priority-Queue and Event-Set Implementations." *Communications of the ACM* 29 (4):300-311.

Joyce, J., G. Lomow, K. Slind, and B. W. Unger. 1987. "Monitoring Distributed Systems." *ACM Transactions on Computer Systems* 5 (2):121-150.

Joyce, J., and B. W. Unger. 1985. "Graphical Monitoring of Distributed Systems." In *SCS Conference on AI, Graphics, and Simulation*, 85-92.

Kahn, G. 1974. "The Semantics of a Simple Language for Parallel Programming." *Information Processing* 74:471-475.

Leung, C. K., D. P. Misunas, A. Neczwid, and J. B. Dennis. 1976. "A Computer Simulation Facility for Packet Communication Architecture." *ACM SIGARCH Computer Architecture News* 4 (4):58-63.

Liao, W., and R. Bagrodia. 1994. "Maisie: A Language for the Design of Efficient Discrete-Event Simulations." *IEEE Transactions on Software Engineering* 20 (4):225-238.

Lomow, G., S. R. Das, and R. M. Fujimoto. 1991. "Mechanisms for User Invoked Retraction of Events in Time Warp." *ACM Transactions on Modeling and Computer Simulation* 1 (3):219-243.

Lomow, G., and B. W. Unger. 1985. "Distributed Software Prototyping and Simulation in Jade." *Canadian INFOR* 23 (1):69-89.

Lubachevsky, B. D. 1988. "Efficient Parallel Simulations of Dynamic Ising Spin Systems." *Journal of Computational Physics* 75 (1):103-122.

Malik, A. W., A. J. Park, and R. M. Fujimoto. 2010. "An Optimistic Parallel Simulation Protocol for Cloud Computing Environments." *SCS Modeling and Simulation Magazine, Society for Modeling and Simulation, Intl.* 1 (4).

Misra, J. 1986. "Distributed Discrete Event Simulation." *ACM Computing Surveys* 18 (1):39-65.

Neal, R., G. Lomow, M. Peterson, B. W. Unger, and I. H. Witten. 1984. "Inter-Process Communication in a Distributed Programming Environment." In *Canadian Information Processing Society Session '84*, 361-364.

Nicol, D., and S. Riffe. 1990. "A Conservative Approach to the Parallelization of the Sharks World Simulation." In *Proceedings of the Winter Simulation Conference*, edited by R. Sadowski, R. Nance, O. Balci, 186-190, Piscataway, New Jersey: IEEE.

Nicol, D. M. 1988. "Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks." *SIGPLAN Notices* 23 (9):124-137.

Nicol, D. M. 1993. "The Cost of Conservative Synchronization in Parallel Discrete Event Simulations." *Journal of the Association for Computing Machinery* 40 (2):304-333.

Nicol, D. M. 2017. "A Performance Model of Composite Synchronization." In *Proceedings of the 2017 SIGSIM PADS Conference.*

Nicol, D. M., and P. Heidelberger. 1993. "Parallel Algorithms for Simulating Continuous Time Markov Chains." In *Workshop on Parallel and Distributed Simulation*, 11-18.

Nicol, D. M., and P. Heidelberger. 1995. "A Comparative Study of Parallel Algorithms for Simulating Continuous Time Markov Chains." *ACM Transactions on Modeling and Computer Simulation* 5 (4):326-354.

Nicol, D. M., and J. Liu. 2002. "Composite Synchronization for Parallel Discrete Event Simulation." *IEEE Transactions on Parallel and Distributed Systems* 13 (5):433-446.

Nicol, D. M., and X. Liu. 1997. "The Dark Side of Risk." In *Proceedings of the 11th Workshop on Parallel and Distributed Simulation*, 188-195.

Nicol, D. M., and P. F. Reynolds. 1984. "Problem Oriented Protocol Design." In *Proceedings of the Winter Simulation Conference*, edited by U. Pooch, C. Pegden, S. Sheppard, 471-474, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Park, A., and R. M. Fujimoto. 2008. "Optimistic Parallel Simulation over Public Resource-Computing Infrastructures and Desktop Grids." *Workshop on Distributed Simulations and Real-Time Application*s.

Petrie, P. 2015. "Ssc Pacific Taking Cyber Test Analysis and Simulation to the Next Level." *CHIPS (Dept of the Navy IT Magazine)*.

Reed, D. A., and R. M. Fujimoto. 1987. *Multicomputer Networks: Message-Based Parallel Processing*. Cambridge, MA: MIT Press

Reinhardt, S. K., M. D. Hill, J. R. Larus, A. R. Lebeck, J. C. Lewis, and D. A. Wood. 1993. "The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers." In *Proceedings of the 1993 Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 48-60.

Riley, G., M. Ammar, R. M. Fujimoto, A. Park, K. Perumalla, and D. Xu. 2004. "A Federated Approach to Distributed Network Simulation." *ACM Transactions on Modeling and Computer Simulation* 14 (1):116-148.

Scalable Network Technologies. 2001. " Qualnet 1.0 User's Manual." Documentation Library; Scalable Network Technologies, Los Angeles, CA 90025;.

Scalable Network Technologies. 2008. " Exata 2.0 User's Manual." Documentation Library; Scalable Network Technologies, Los Angeles, CA 90025.

Steinman, J. 1993. "Breathing Time Warp." In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, 109-118.

Tomasulo, R. M. 1967. "An Efficient Algorithm for Exploiting Multiple Arithmetic Units." *IBM Journal of Research and Development,* 11 (1):25-33.

Unger, B. W., G. Birtwistle, J. Cleary, D. Hill, G. A. Lomow, R. Neal, M. Peterson, I. H. Witten, and B. L. M. Wyvill. 1984. " Jade: A Distributed Software Prototyping Environment." In *Conference on Simulation in Strongly Typed Languages*, 77-83. SCS Simulation Series.

Unger, B. W., A. Dewar, J. Cleary, and G. M. Birtwistle. 1986a. "A Distributed Software Prototyping and Simulation Environment: Jade." In *Conference on Intelligent Simulation Environments*, 63-71. SCS Simulation Series.

Unger, B. W., A. Dewar, J. Cleary, and G. M. Birtwistle. 1986b. "The Jade Approach to Distributed Software Development." In *Conference on AI Applied to Simulation*, 178-188. SCS Simulation Series.

Unger, B. W., Z. Xiao, J. Cleary, J. Tsai, and C. Williamson. 2000. "Parallel Shared-Memory Simulator Performance for Large Atm Networks." *ACM Transactions on Modeling and Computer Simulation* 10 (4):358-391.

Varshney, M., K. Pickett, and R. Bagrodia. 2011. "A Live-Virtual-Constructive (Lvc) Framework for Cyber Operations Test, Evaluation and Training." In *Military Communications Conference (MILCOM)*.

Wieland, F. 2001. "Practical Parallel Simulation Applied to Aviation Modeling." *15th Workshop on Parallel and Distributed Simulation*, 2001/05/15, at Lake Arrowhead, CA.

Williamson, M., J. Hoyle, and R. Bagrodia. 2012. "Software Virtual Networks (Svn) for Network Test, Training and Operations Lifecycle." Scalable Network Technologies Inc; White Paper,

Wilson, A. L., and R. M. Weatherly. 1994. "The Aggregate Level Simulation Protocol: An Evolving System." In *Proceedings of the 1994 Winter Simulation Conference*, edited by D. Sadowski, A. Seila, J. Tew, S. Manivannan, 781-787, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wu, H., R. M. Fujimoto, and G. F. Riley. 2001. "Experiences Parallelizing a Commercial Network Simulator." In *Proceedings of the Winter Simulation Conference*, edited by M. Rohrer, D. Medeiros, B. Peters, J. Smith, 1353-1360, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Xiao, Z., B. W. Unger, J. G. Cleary, G. Lomow, K. Slind, and L. Xining. 1987. "Jade's IPC for Distributed Simulation." In *Society for Computer Simulation Multi-Conference.*

Zeng, X., B. R., and M. Gerla. 1998. "Glomosim: A Library for Parallel Simulation of Large-Scale Wireless Networks." In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*.

Zhou, J., Z. Ji, and R. Bagrodia. 2006. "Twine: A Hybrid Emulation Testbed for Wireless Networks and Applications." In *IEEE INFOCOM.*

AUTHOR BIOGRAPHIES

RICHARD FUJIMOTO is a Regents' Professor in the School of Computational Science and Engineering at the Georgia Institute of Technology. He received a Ph.D. in Computer Science & Electrical Engineering from the University of California-Berkeley in 1983. He has been an active researcher in the parallel and distributed simulation field since 1985, and has authored or co-authored 3 books and over 250 technical papers on this subject including 7 award winning publications. He led the definition of the time management services for the High Level Architecture for modeling and simulation (IEEE standard 1516). He received the ACM Distinguished Contributions in Modeling and Simulation Award in 2013. His e-mail address is fujimoto@cc.gatech.edu.

RAJIVE BAGRODIA is the Founder & CEO of Scalable Network Technologies and a Professor Emeritus in the Computer Science Department at UCLA. He received his PhD in Computer Science from the University of Texas at Austin. Together with members of his research group at UCLA and

engineering team at Scalable, he has published over 150 papers in the areas of distributed computing, parallel simulation, wireless networks and cyber M&S; the groups have also created a number of public domain and commercial software that are in use worldwide including PARSEC, GloMoSim, QualNet™, and EXata™. He served as the Program Manager for the Future Combat System Communication Effects Server project at Scalable, which was recognized by a US Army Modeling & Simulation Award in 2008. His e-mail address is rbagrodia@scalable-networks.com.

RANDAL BRYANT is a University Professor in the Computer Science Department at Carnegie Mellon University. He received his B.S. in Applied Mathematics from the University of Michigan in 1973, and his PhD from MIT in 1981. He was an assistant professor at Caltech from 1981 to 1984 and has been on the faculty at Carnegie Mellon since 1984. Most of Dr. Bryant's research has focused on methods for formally verifying digital hardware and software. He is well known for the development of the ordered binary decision diagram (OBDD) data structure. He has received major awards from the IEEE, ACM, and industry organizations recognizing the impact of his work. Along with David R. O'Hallaron, he coauthored *Computer Systems: A Programmer's Perspective*, now in its third edition, a textbook in use at over 320 universities worldwide. His e-mail address is Randy.Bryant@cs.cmu.edu.

K. MANI CHANDY is the Simon Ramo Professor, Emeritus at the California Institute of Technology. He got his Bachelors in Electrical Engineering at the Indian Institute of Technology, Madras, in 1965; MS in Electrical Engineering at the Polytechnic Institute of New York in 1966; and a PhD in Operations Research at the Massachusetts Institute of Technology in 1969. He taught at the University of Texas at Austin, Computer Science Department, from 1969 to 1987, and at the California Institute of Technology from 1987 to 2014. He has written papers and books on performance modeling and concurrent systems. He received the IEEE Koji Kobayashi Award, the CMG A. A. Michelson Award, the ACM Dijkstra Award, and the IEEE Harry Goode Award He is an IEEE Fellow and a member of the National Academy of Engineering. His email address is kmchandy@gmail.com.

DAVID JEFFERSON is a Visiting Scientist at Lawrence Livermore National Laboratory. He holds a Ph.D. in Computer Science from Carnegie Mellon University, and has worked in the field of parallel discrete event simulation (PDES) for decades. He is the co-inventor (with Henry Sowizral) of optimistic methods for PDES and was the architect of the first optimistic simulator, the Time Warp Operating System. He has also worked in various other fields, including distributed computation, synchronization, cybersecurity, public election security, and artificial life. His email address is drjefferson@llnl.gov.

JAYADEV MISRA is Schlumberger Centennial Chair Emeritus and University Distinguished Teaching Professor Emeritus at the University of Texas at Austin. He works in the area of concurrent programming with emphasis on rigorous methods to improve the programming process. His work on the UNITY methodology, jointly with Mani Chandy, has been influential in both academia and industry, and has spawned a large number of tools and research projects. He and Mani Chandy (and, independently, Randy Bryant) pioneered the area of Distributed Discrete Event Simulation. He is currently working on a programming language, called "Orc", for concurrent orchestrations of interacting components. He is also spear-heading an effort, jointly with Tony Hoare, to automate large-scale program verification. Misra has been awarded the Harry H. Goode Memorial Award of IEEE for 2017 (jointly with K. Mani Chandy) and Doctor Honoris Causa by the Ecole Normale Superieure de Cachan, France, in 2010. He was identified as a highly cited researcher by ISI, in 2004. He is the author of two books, "Parallel Program Design: A Foundation", Addison-Wesley, 1988, co-authored with Mani Chandy, and "A Discipline of Multiprogramming", Springer-Verlag, 2001. His email is misra@utexas.edu.

DAVID NICOL is the Franklin W. Woeltge Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, and Director of the Information Trust Institute. He is the PI for two recently awarded national centers for infrastructure resilience: the DHS-funded Critical Infrastructure Reliance Institute, and the DoE funded Cyber Resilient Energy Delivery Consortium. His research interests include trust analysis of networks and software, analytic modeling, and parallelized discrete-event simulation, research which has lead to the founding of startup company Network Perception, and election as Fellow of the IEEE and Fellow of the ACM. He is the inaugural recipient of the ACM SIGSIM Outstanding Contributions award. He received the M.S. (1983) and Ph.D. (1985) degrees in computer science from the University of Virginia, and the B.A. degree in mathematics (1979) from Carleton College. His email address is dmnicol@illinois.edu.

BRIAN UNGER is Professor Emeritus, Computer Science, at the University of Calgary, Alberta, Canada. He is Board Vice Chair of KITD, a Cambodian NGO aimed at ICT4D. He was the founding President and CEO of iCORE, the "informatics Circle of Research Excellence" from 1999 though 2005. Under his leadership, iCORE funded 17 Chairs and Professors whose research teams in 2005 included over 500 faculty, post doctoral fellows, research staff and graduate students working in targeted areas within the communication networks, nano-informatics, and intelligent software areas. He was the founding President of Netera Alliance, now Cybera Inc., a research consortium, and was the founding Board Chair of C3.ca, now Compute Canada, a consortium of over 30 universities, corporations and government agencies. He was the founder of Jade Simulations, a private for-profit corporation, and served as its CEO, 1988-1993. Dr. Unger was named a Canada Pioneer of Computing in 2005, received the IWAY Public Leadership award for outstanding contributions to Canada's information society in 2004, and the 1993 ASTech award for "Innovation in Alberta Technology". His research has been focused on parallel and distributed simulation and on "grid computing" middleware. Over 150 published journal, conference papers and edited books. His e-mail address is unger@acm.org.