Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

THE EVOLVING RELATIONSHIP BETWEEN SIMULATION AND EMULATION: FASTER THAN REAL-TIME CONTROLS TESTING

Bernard Brooks
Adam Davidson
Ian M^{c}Gregor

Emulate3D Ltd
Reading Enterprise Centre,
RG6 6BU, UK

ABSTRACT

Modern facilities, such as automated material handling systems, baggage handling, or manufacturing process systems, are complex systems controlled by various control units on different automation levels. The design and development of these facilities require the application of CAD, simulation, development and testing tools. Traditionally different tools have been used at each stage, and that separation has tended to impede the smooth delivery of the project. In this paper, we show that the same modeling software can be deployed at all stages of the design – from layout and simulation through to controls testing. Additionally we show that by incorporating a PLC emulator, tightly integrated with the virtual clock, and using internal communication channels, that real control logic can also be used at every stage. The engineer has complete control over the level of accuracy for modeling, simulating and emulating, while still running the model faster or slower than real-time.

1 INTRODUCTION

1.1 Simulation and Emulation

A *simulation task* (as distinct from an emulation task) is an empirical study of a modeled world by means of repeated experimentation with varying parameters, the aim of which is to extract key metrics, such as throughput, utilization, or path length. The task may also involve the development and comparison of key algorithms, such as routing and storage strategies (Seidel, Donath, and Haufe 2012; McGregor 2002).

Simulation tools provide an environment designed to approximate the real world with the aim of making sufficient computational savings that the model can be run many times faster than real-time. The simulator deliberately trades functional fidelity for scale and speed, in order to make the *simulation task* tractable in a reasonable time.

An *emulation task*, on the other hand, is the task of testing existing (or developing new) process-logic or control-logic without the need for real equipment. The aim is to complete the logic testing in advance of the facility being built or modified, and thereby easing the eventual commissioning of the process/control software (virtual commissioning and factory acceptance testing), and saving time and money.

Emulation tools must ensure functional fidelity by replicating the real world sufficiently faithfully that the connected test equipment cannot distinguish it from the real world (McGregor 2002). The emulated model can then be used to test the functionality or logic of the external equipment.

1.2 The Evolving Relationship Between Simulation and Emulation

Although the definitions of the terms *simulation* and *emulation* remain current, over the last 10 years, their use as applied to *software tools* has evolved. Between simulation and emulation there's a spectrum of modeling accuracy from 'gross approximation' at one end through to 'detailed accuracy' at the other.

Particularly in the context of automated material handling systems, baggage handling, and manufacturing process systems, simulation tools have traditionally occupied the 'approximation' Discrete Event Simulation (DES) end of the spectrum, while emulation tools have tended to occupy the more accurate '3D physics engine' Discrete Time Simulation (DTS) end. Indeed there continues to be a tendency to equate 'emulation tools' with 'physics engine' (and vice versa), and to equate 'simulation tools' with DES (and vice versa) (Seidel, Donath, and Haufe 2012). These categorizations were useful 10 years ago, when the marriage between emulation and physics was a key distinguishing factor between emulation tools and traditional simulation tools. Particularly when the simulation tools inherently simplified the world in a way incompatible with controls testing. For example, a top view of a sloping conveyor appears to compress its length, and a 2D world struggles to model a 'height sensitive' photo-eye in a baggage handling system (McGregor 2002). Such categorizations also served to divorce the new emulation tools from the inherent 'order driven' material flow management employed by the traditional simulation tools.

1.3 Extending the Simulation/Emulation Software Environment

Categorizations of simulation and emulation software, based largely on modeling accuracy and modeling techniques/algorithms is no longer seen as helpful. Fortunately it has increasingly given way to a more useful feature-based categorization.

A Discrete Event Simulator (DES) simulation tool needs to provide mechanisms for running the scenario multiple times, altering the parameters between each run, and collecting and analyzing statistics. A controls testing, or Material Flow Controller (MFC) / Material Control System (MCS) / Warehouse Management System (WMS) testing tool needs the appropriate external connectivity – perhaps OLE for Process Control (OPC) (Seidel, Donath, and Haufe 2012; McGregor 2002), but often more specialized. In neither case is the level of modeling accuracy provided by the software necessarily relevant in its categorization. Rather, that selection is entirely project dependent. Controls engineers may choose to use a highly simplified linear (one-dimensional) physics tool to test their control logic, if that level of world accuracy is sufficient for the logic being tested. Conversely, a simulation project may use a full blown 3D physics-based Discrete Time Simulator (DTS) if it runs at sufficient speed and repeatability.

Modern simulation/emulation software, such as Emulate3D, already reflects this reality. Such software already offers the full spectrum from linear, 2D to 3D DES through to a full 3D physics-based DTS within one tool (McGregor, 2012). And, by providing mechanisms for connecting to real systems (MFC, MCS, WMS, PLC, database, etc) all within the same tool, users can mix and match features to the task at hand. The assertion that "implementation, test and verification of each level [of MCS facility planning] are carried out separately, because no unified common test bench/environment [is] available for all levels" (Seidel, Donath, and Haufe 2012) is no longer true (Koflanovich and Hartman 2010).

Although the *simulation* and *emulation tasks* remain distinct, the *software tools* used to conduct those tasks need not make the same distinction.

In this paper we present a full PLC emulator embedded within the simulation/emulation software environment and capable of running existing PLC controls programs lock-step with the modeled world. We will show how a lock-step PLC emulator fits into both the traditional *simulation* and *emulation* tasks, and further blurs the role of the 'modeling software' in executing those tasks.

2 FASTER THAN REAL-TIME CONTROLS TESTING

2.1 State of the Art

It has frequently been reported that the testing of hardware-in-the-loop control systems requires the emulation model to run exactly (or as closely as possible) to real-time (Seidel, Donath, and Haufe 2012; Johnstone, Creighton, and Nahavandi 2007; McGregor 2002). Normally such models are connected to real hardware systems such as an MFC and/or PLC. Necessarily, for models connected in this way, the emulator must run exactly in real-time since the connected control systems will themselves be running in real-time. More specifically, the emulation cycle time must be no longer than the control cycle time, otherwise undefined/unreal results will be obtained (Hoher et al. 2011).

The essential characteristic of these assertions is that the emulation and the control system both have independent internal clocks, and those clocks need to be precisely synchronized in order that time-dependent algorithms remain valid (e.g. PLC timers, or shaft encoder pulses used to deduce speed). Where either clock can only run at real-time, as is typical of a PLC, then the entire emulation test environment must also run at real-time. Additionally, communication speed (latency and bandwidth) between the emulation and the PLC is often a defining limit (Johnstone, Creighton, and Nahavandi 2007), and attempting to run the model and PLC faster than real-time would increase the likelihood of exceeding that limit.

Both issues of time and communications need to be adequately addressed in order to run the system faster than real-time.

2.2 An Integrated PLC Emulator

Investigators in the area of simulated computer networks have already attempted to address these same issues. In developing a test bed for large computer networks, Zheng, Jin, and Nicol (2012), and Jin et al. (2012) used virtual machine technology (OpenVZ) to model each machine in the network. The software under test would run unmodified in their respective virtual machine. Their key contributions were to synchronize all the virtual machines with a single shared virtual clock, and to manage the timing and flow of communications across the virtual network using a central manager. In doing so, they managed to combine emulation and simulation in a single environment, and run it faster or slower than real-time.

We propose a simpler environment where the PLC is emulated within the simulation/emulation tool. This allows us to pull the external test equipment into the model, converting from hardware-in-the-loop to software-in-the-loop. Critically we also gain control both over the virtual clock and over the channels of communication between the model and the PLC emulator, but without the overhead and complexity of time-synchronized virtual machine technology.

The original requirement to include the PLC hardware in the test environment was only for it to host and execute the control logic. If this can be achieved within the simulation/emulation tool, we can maintain functional fidelity and still exercise and test the control logic.

The use of a single source of time, and having control over the virtual clock is the key. Given that the world model (including the DES or DTS), the emulated PLC, and the communication channels between the two all share the same concept of time, we can now run the model in virtual time – being faster or slower than real time, or pause, or single step between events.

2.3 Simulation or Emulation

Typically a real PLC will scan continuously – executing the control logic repeatedly in an endless loop. Different PLCs use different methodologies for running the PLC operating system – some run the control logic asynchronously with the operating system, and some run it synchronously. Either way, essentially the control logic is scanned repeatedly in a loop, without pause (Introductory PLC Programming; SIMATIC Programming with STEP 7 V5.5, ch. 4.2). With some exceptions (e.g. Beckhoff TwinCAT),

the PLC eagerly consumes PLC CPU time. There is no advantage to the PLC of entering the CPU idle state.

The scan cycle time then becomes a function of the speed of the PLC CPU and the typical length of time to execute one scan (which is itself dependent on the size of the control code). Since the code path through the control logic can vary depending on the inputs and the current state, the scan cycle time can vary too – but typically over a small range.

As with any emulation/simulation there is a tradeoff between high functional fidelity and speed. A similar debate to the simulation-versus-emulation modeling accuracy touched on in the Introduction can be leveled at the PLC emulator.

At one end of the spectrum, the PLC emulator must emulate the PLC operating system and the control logic, taking into account the interactions between the two, and ensuring that the execution timing of each CPU instruction is modeled accurately. This level of detail is usually only catered for by the PLC manufacturers own PLC simulator.

At the other end of the spectrum, the PLC emulator can take advantage of the fact that many scan cycles will evaluate the same inputs, execute the same logic, and produce the same outputs. By analyzing the program to determine what logical events cause a change in the outputs, the emulator can use a Discrete Event Simulator, scanning only when events (such as PLC timers and inputs changing) would result in a change to the outputs.

2.4 The Architecture of a PLC Emulator

The Emulate3D PLC Emulator can be configured to use either a DTS or DES. As a DTS it simulates the average PLC cycle time, and scans the control logic periodically (in model time) regardless of changes of state, inputs or timers.

As a DES, it analyzes the control logic, looking to see which internal controls the logic is evaluating. For example, a program that uses timers may only use the timer 'expired' signal in the logic, or it may use the content of the timer accumulation register. The rate at which these timer outputs change is quite different, and the analyzer must determine which is being used in the logic.

Using this information, the DES can then execute a scan only when a change occurs to an internal control (e.g. timer), or external input (e.g. photo-eye becoming blocked). Even in this mode, as events occur, it only executes the scan at the next (simulated) PLC cycle, so that changes to the inputs don't necessarily cause an immediate scan. The cycle time defines the time it will take the PLC to respond to changed input conditions and also determines the time taken to reply to external commands (Johnstone, Creighton, and Nahavandi 2007), and this detail should not be ignored by the emulator.

As with the choice of accuracy of physical world modeling, the level of PLC emulation detail is up to the modeler who can select the most appropriate mode for the task, and even mix and match within the same model.

3 APPLICATIONS OF AN EMULATED PLC

3.1 Controls Testing

To date, testing control systems has been constrained to real-time. By using an emulated PLC, the engineer can continue to test their control logic, but the real-time constraint is removed:

Faster than real-time: The most obvious advantage is that the model can be run faster than real-time. The engineer can run the model at speed in order to observe the system running over long periods of modeled time without having to wait.

Slower than real-time: If a problem has been observed, the engineer may want to see the problem develop in slow motion so as to better understand its mechanics.

Pause: Once a problem occurs, the engineer may want to pause the model, pausing the entire system – modeled world, communications and PLC in order to inspect the causes of the problem. The engineer can inspect the state in the model and the PLC at their leisure, correlating the two and looking for

discrepancies, without the PLC continuing to scan and change state. For troublesome scenarios, the engineer may even inject artificial events that can automatically pause the model when an error occurs (e.g. timer expiring in the PLC, or load falling off a conveyor in the modeled world), introducing a new debugging function 'run-to-error'.

Single step over events: The ultimate in debugging hard-to-see issues is the ability to advance time step by step, observing the problem as it develops.

3.2 Modeling and Testing Large Complex Facilities

To date, the engineer has been constrained to limit the size of their model to that which will reliably run in real-time. For a complex system this means it may be possible to test only parts of the control system. Glinsky et al. (2004) in their modeling of a hardware-in-the-loop system incrementally moved elements from the model into the real world as they became available. This same concept has been applied to emulation models. Control of elements within the model would be passed back and forth between the external control system and the simulation engine (Johnstone, Creighton, and Nahavandi, 2007) in order not to overwhelm the simulation.

With an emulated PLC, however, the engineer can model those larger, more complex systems, in their entirety. Models that may on occasion not run fast enough (as defined by Hoher et al. 2011), can now be tested without any loss of confidence in the test.

3.3 Multiple Controllers

An automated MHS facility may use a large number of controllers, and the modeler may need to connect many (or all) of those controllers in order to make the system work. Koflanovich and Hartman (2010) describe a live modernization of a mission-critical automated material-handling system. They describe how "twenty-one programmable controllers, a supervisory computer system, and four operator interface applications, all developed to operate the physical system, were connected to the model. All sub-systems were brought online, and the system was run as a whole."

As well as being expensive, attempting to operate and connect large numbers of controllers to a model eventually becomes unmanageable. Often the engineer can use the actual PLCs that are destined for the facility, but this may not always be possible, and even if it is, it prevents more than one engineer testing different aspects of the system at the same time, since they can't all connect the same set of controllers to different models. Alternatively engineers may use virtual operating system environments running PLC emulators in order to cope with lack of real equipment (Phillips and Montalvo 2010, Starner and Chessin 2010). Even so, modeling with multiple running simulators is awkward. Embedding an emulator in the modeling software can alleviate these problems.

Modeling simplicity: The code downloaded into the PLC emulator is incorporated into and saved with the model. This means that the modeled environment can be shared between engineers, including all the control code, without having to pass around multiple files, or requiring set-up and configuration of multiple pieces of hardware or simulations.

Modeling multiple PLCs: The modeling software (e.g. Emulate3D) can be made to present a virtual network to the outside world, with multiple PLC emulators each appearing as individual nodes on the network. The network and all the PLCs are in fact modeled entirely inside the software. The configured network and PLCs are indistinguishable from a real network with multiple PLCs. The engineer uses the normal PLC programming software to program and configure each PLC, exactly as they would in a real commissioning exercise. The world model and all the PLCs share the same virtual clock, and all run lock-step together.

3.4 Simulation of an Existing System

A simulation makes approximations in order to save CPU time, but it isn't necessarily *required* to do so. A simulation can include emulated elements where needed, and can still run faster than real-time so long

as repeat runs with the same initial parameters produce the same results. Or a simulation can include a mix of simulated (simplified) and emulated PLC control logic.

Often a simulation is deployed in order to solve problems or look for improvements in existing facilities. For example, simulation projects for retro-fitting new equipment into existing plants. Koflanovich and Hartman (2010) state "the risk of … incorrect documentation, and misunderstood functionality within the existing system is much higher than with a green-field installation." In these cases, it may be simpler and more robust just to use the existing control logic. As LeBaron and Thompson (1998) state "one of the main benefits of emulation is that it eliminates the need to reimplement code. Because the actual control system is used to develop, test, and refine algorithms and logic, it exists as developed in the real system. This eliminates reimplementation errors and provides greater confidence in the emulation results."

The use of an integrated PLC emulator, using a virtual clock and communication channels, allows the model to be run repeatedly and to function as a traditional simulator, while under the control of the actual control logic. By employing Discrete Event Simulation techniques in the PLC emulator, the original controls logic can be incorporated into the model within the DES framework of the software, maintaining all the speed advantages of simulation.

3.5 Simulation Credibility

Facility planning tends to follow a well trodden route (Seidel, Donath, and Haufe 2012): "The layout is designed using CAD software and is the basis for simulation tools which are employed to evaluate and confirm the systems KPI's. The layout is then improved through iterative adjustment of the simulation model. This leads to a precise requirements specification which is used to develop the MFC and PLC programs. With this workflow, a final joint test of the entire working system can only be done at a late stage in the project. The consequences of which are inconsistencies between the intended behavior, as detailed in the functional specification and the implemented behavior of the control systems."

Such inconsistencies may be a result of the approximations of the simulated system. Or they may be due to bugs or unforeseen consequences in the implementation of the MFC and PLC programs. Often the MFC and PLC engineers will have had to translate the logic used in the simulation (usually written in the simulators own programming language) into native controller language. Inevitably errors will occur in that translation.

But by using a single software tool and single model, capable of performing both the simulation and emulation tasks, the modeler is free to alter the traditional workflow.

Emulate First: The modeler may choose to make use of an initial emulation in order to gather the necessary statistics for a piece of equipment, that can then be plugged back into the simulation. If the facility is planning on using off-the-shelf components that already has existing well-tested control logic, it may be simpler just to use that logic directly in the initial simulation rather than back-port it (re-implement it) in simulation code.

Incremental Emulation: Glinsky et al. (2004) incrementally moved elements from the model into the real world as they became available. The modeler can do something similar: Starting with the simulation, they can incrementally replace parts with emulated systems as they become available. The simulation can be re-run as each program is developed to ensure continued adherence to the specification. Bleifuß and Spieckermann (2012) describe a successful case study where this incremental approach to simulation/emulation was employed.

High Fidelity Simulation: Sometimes the approximations being made in the simulation reduce the credibility of the results. It may be necessary for a particular project to increase the fidelity of the model in order to preserve credibility. Potentially, CPU and machine performance allowing, the entire simulation could be run with accurate physics and real control logic.

4 SUMMARY

Whereas the tasks that modelers set out to achieve is largely unchanged, simulation/emulation software has evolved significantly over the last 10 years. Competent modeling software should be comfortable in both arenas. Modeling software always offers an approximation on reality, but it's the modeler who should choose, based on the task at hand, the level of accuracy that the modeled world needs to employ. The software shouldn't dictate the modeling accuracy, nor be defined by it. Instead the software should offer a set of 'sliders' to allow the modeler to tune the level of detail that they require.

The real distinction between simulation from emulation *software tools* is the set of functionality that they offer: simulation requiring features to vary parameters, run repeat experiments, and to analyze the results; and emulation requiring the necessary features to test existing control logic, either externally or internally.

The addition of an emulated PLC to the set of features on offer further extends both types of tool. An integrated emulated PLC, sharing a virtual model clock and managing communications in model time, affords the modeler exciting new possibilities in simulation and emulation:

- The modeler can now conduct controls testing faster than real-time, slower than real-time, pause time, or single step event by event. Together, these new features significantly improves the testing experience.
- The performance of the model for large complex facilities need no longer be a concern for the controls engineer. Even if complexity causes the model to slow down below real-time, fidelity is maintained, and the test results remain valid.
- Without using an integrated PLC emulator, a system requiring large numbers of controllers quickly becomes expensive and unmanageable. Instead, the engineer can model numerous PLCs inside one software environment – accessing, programming and monitoring each with their usual programming software.
- By modeling separate parts of a facility (such as one conveyor assembly, or one baggage tag reader) the modeler can start with an emulation, gathering statistics on individual parts, switch to a simulation of the entire system using the data collected, and then as the requirements specification becomes clear, gradually switch parts back to emulation in order to test.

There are numerous combinations of simulation/emulation configurations, allowing the modeler to mix and match both the accuracy of physical world and/or the controls logic. By hosting all the functionality within one tool, the modeler has complete freedom either to follow the traditional planning workflow (layout, simulation, implementation, emulation and test) or to create their own workflow. The engineer no longer has the workflow dictated to them by the separation of tasks into different tools.

REFERENCES

Applied Materials, Inc. 2014. "AutoMod: Simulating Reality". Accessed April 15, 2014. http://www.appliedmaterials.com/global-services/automation-software/automod.

Bleifuß, R., and Spieckermann, S. 2012. "A Case Study on Simulation and Emulation of a New Case Picking System for a US Based Wholesaler." In Proceedings of the 2012 Winter Simulation Conference, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Berlin, Germany. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Emulate3D Ltd. 2014. "Emulate3D PLC Controls Testing and Virtual Commissioning". Accessed April 15, 2014. http://www.demo3d.com.

Glinsky, E., and Wainer, G. 2004. "Modeling and Simulation of Hardware/Software Systems with CD++." In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 198-205, Washington, DC. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hoher, C., Schindler, P., Göttlich, S., Schleper, V., and Röck, S. 2011. "System Dynamic Models and Real-time Simulation of Complex Material Flow Systems." In *Enabling Manufacturing Competitiveness and Economic Sustainability*, *Proceedings of the 4th International Conference on Changeable, Agile, Reconfigurable and Virtual Production*, edited by H. ElMaraghly, 316-321. Springer Berlin Heidelberg.

Introductory PLC Programming. http://en.wikibooks.org/wiki/Introductory_PLC_Programming. Accessed April 15, 2014.

Jin, D., Zheng, Y., Zhu, H., Nicol, D., and Winterrowd L. 2012. "Virtual Time Integration of Emulation and Parallel Simulation." In *Proceeding of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS)*, 120-130. Zhangjiajie, China.

Johnstone, M., Creighton, D., and Nahavandi, S. 2007. "Enabling Industrial Scale Simulation / Emulation Models." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1028-1034, Washington, DC. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Koflanovich, N., and Hartman, P. 2010. "Live Modernizations of Automated Material Handling Systems: Bridging the Gap Between Design and Startup Using Emulation." In *Proceedings of the 2010 Winter Simulation Conference,* edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, Baltimore, MD. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

LeBaron, T., and Thompson, K. 1998. "Emulation of a Material Delivery System." In *Proceedings of the 1998 Winter Simulation Conference*, edited by D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, 1055-1060, Washington, DC. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

McGregor, I. 2002. "The Relationship Between Simulation and Emulation." In *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 1683-1688, San Diego, CA. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

McGregor, I. 2012. "Introduction to Emulate3D – Emulation, Simulation, and Demonstration." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Berlin, Germany. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

OPC Foundation, 2014. "About OPC - What is OPC?" Accessed April 15, 2014. https://opcfoundation.org/about/what-is-opc.

Phillips, R., and Montalvo, B. 2010. "Using Emulation to Debug Control Logic Code." In *Proceedings of the 2010 Winter Simulation Conference,* edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, Baltimore, MD. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Seidel, S., Donath, U., and Haufe, J. 2012. "Towards an Integrated Simulation and Virtual Commissioning Environment for Controls of Material Handling Systems." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, Berlin, Germany. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Siemens. "SIMATIC Programming with STEP 7 V5.5." Chapter 4.2. Accessed 15 April, 2014. https://support.automation.siemens.com/WW/adsearch/pdfviewer.aspx?ehbid=45531107&nodeid=45531202.

Starner, C., and Chessin, M. 2010. "Using Emulation to Enhance Simulation." In *Proceedings of the 2010 Winter Simulation Conference,* edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, Baltimore, MD. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zheng, Y., Jin, D., and Nicol, D. 2012. "Validation of Application Behavior on a Virtual Time Integrated Network Emulation Testbed." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Berlin, Germany. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

BERNARD BROOKS is a Senior Software Engineer at Emulate3D since 2007, and at AutoSimulations prior to that. He holds a Ph.D. and a B.Sc. Hons in Computer Science from the University of Reading, England. His email address is bernard.brooks@demo3d.com.

ADAM DAVIDSON is a director and co-founder of Emulate3D Ltd since 2005. He has a B.Sc. Hons in Computer Science from the University of Reading in England, where he continued as researcher in Simulation and Scheduling. His research became the APF Real-Time Dispatcher and Reporter, which was acquired by AutoSimulations Inc. in 1997 (now Applied Materials Inc.) where he became the software development manager. His email address is adam.davidson@demo3d.com.

IAN W. MCGREGOR is a director and co-founder of Emulate3D Ltd since 2005. Prior to that he worked as Simulation Business Development Manager for AutoSimulations since joining them in 1996, and was posted in Singapore, Japan, and Utah. He has an M.Sc. in Computer Integrated Manufacturing from Cranfield Institute of Technology in England, a Diplome d'Ingenieur from the Universite de Technologie de Compiegne in France, and a B.Sc. in Production Engineering from Kingston Polytechnic in England. He is a Chartered Member of the Institute of Mechanical Engineers and served as Registration Chair for WSC 2001. His email address is ian.mcgregor@demo3d.com.