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

MULTI-LEVEL EDUCATIONAL EXPERIMENT IN DISTRIBUTED SIMULATION

Charles Turnitsa

TSYS School of Computer Science
Columbus State University
Columbus, GA 31907, USA

ABSTRACT

Giving collegiate students an opportunity to participate in a multi-faceted development project remains a goal of many computer science programs, as well as modeling and simulation educational programs. A key feature of many complex simulation studies remains that distributed systems are relied on for their implementation, yet presenting such a system to students as a project is often overwhelming. Devising a simpler architecture, which involves not only distributed hardware, but also development of simulation software, and interoperability software, is the goal of the approach described here. As a work in progress, final results are not yet available, but principles and some foundational decisions are presented, with enough details such that the effort can be reproduced by other institutions.

1 INTRODUCTION

Providing educational projects for students at the undergraduate and master's level of education is a process that requires balancing a number of constraints. The project should (1) provide an example of the principles and educational objectives the instructor wishes to impart, (2) allow the student to experience some of the application of those objectives in a developmental and experimental environment, and (3) allow for the completion of the project within the limits of time and resources that are necessarily imposed by the education process.

Often the satisfaction of these enumerated constraints reduces many simulation projects to the development of an individual simulation, the application of a simulation study to an application domain problem, or the investigation of theory – in either simulation or modeling. The process of developing a distributed simulation environment, were the student has access to both the distribution techniques, as well as the contributing simulators is one that can provide many educational benefits in preparation of technologists (scientists and engineers) for the applications field of working with distributed simulations, however it is a very difficult mixture because both elements (the distributed environment techniques and tools, and the individual simulators) require a significant amount of study and understanding to make such a study valuable and possible within the three constraint areas enumerated above.

As an approach to satisfying this problem, and towards providing a series of projects for students at different levels of their education, an experimental environment was envisioned and is reported on here, along with several of the initial activities within that environment. Over time, it will be possible to assess whether the enumerated constraints above will be satisfied, however, the initial design does take them into account, and it is presented here with the intention of making the approach available to others – as possible stimulus to other approaches, or as an initial assessment of lessons learned.

2 SIMULATION EDUCATION

Rather than consider the use of modeling and simulation as a tool for teaching other topics, this paper is focused on the teaching of aspects of modeling and simulation. As that is such a broad topic, however, some further definition may be useful. Here is an effort, at least from the perspective of a graduate level approach to the problem, of attempting to classify some of the elements that make up the topic of modeling and simulation education. Three different faces of this classification effort are presented here, a taxonomical categorization of topics, a review of the current state of the body of knowledge, and an index of some elements of Bloom's educational classification that may be useful for the generation of learning objectives for these topics.

2.1 Topics of Modeling and Simulation Education

The list of topics presented here is not intended to be complete, nor could it be. However, these are topics that it is felt are common to many of the different applications areas of modeling and simulation, and that are part of a number of different educational programs reviewed at different universities, making up what is perhaps a core of a modeling and simulation education at the graduate level.

- Models – Modeling paradigms and techniques of representing a system as a model
- Simulation – The use of a model to produce information about a system, distinctions between different simulation paradigms (stochastic, deterministic, etc.)
- Simulator development – Developing simulator software, verification and validation, visualization and human factors
- Simulation study – Interpretation of results, application of simulation, input analysis, output analysis, simulation project lifecycle
- Distributed Simulation – Distributed techniques, parallel and distributed simulation development and operations, interoperability
- Multi-Model Simulation – Model resolution, model harmonization, model orchestration

A recent examination of definitions for these terms, and many other related to modeling and simulation, along with a history of their use and origination, has recently be presented (Oren, 2011a).

There are, of course, many contributing disciplines that make up the foundation for these topics (mathematics, computer science, systems engineering, etc.). There are, also, many additional disciplines where these topics can take the student (cognitive systems, statistical analysis, data visualization, etc.). Frequently, because of the applied nature of modeling and simulation as a discipline, course content revolves around a particular applications approach (for example, Agent Based Modeling, Discrete Event Simulation, or Continuous Simulation), which might include material covering a number of the topics from above (modeling techniques, simulator development, simulation study), as they would pertain to that applications approach.

There are many different universities, and many different online programs. Several were consulted in preparation of the list of proposed topics here, and it is worth presenting the findings from one. In an approach towards defining an online M.S degree in Modeling and Simulation, the proposed content of such a program was presented by a number of authors representing the program at Arizona State University (Sarjoughian 2004). While a number of institutions have online curricula descriptions that were consulted during the preparation of this work, the rational and pedagogy behind the Arizona State University are available in publication, and are listed here. A similar effort has also been published concerning the University of Nebraska at Lincoln (Vakilzadian 2003), although focusing in that effort specifically on modeling and simulation for engineering programs at the graduate level. A similar effort was prepared under a National Science Foundation grant, exploring a curriculum for undergraduates (Vakilzadian and Möller 2009).That program has educational units presented in two categories, Core Courses, and Area Courses. Here are the enumerated lists:

<u>Category 1: Core Courses</u>

- Modeling and Simulation Theory and Application
- Simulating Stochastic Systems
- Design of Engineering Experiments
- Software Analysis and Design
- Software Project, Process and Quality Management

<u>Category 2: Area Courses</u>

- System Simulation
- Simulation in Manufacturing
- Supply Chain Modeling and Analysis
- Parallel and Distributed Simulation
- Scheduling Network Analysis Models
- Factory Physics
- Applied Stochastic Operations Research Models
- Software Requirements and Specification
- Software Verification, Validation and Testing
- Hardware Description Languages
- Synthesis with Hardware Design Language
- Applied Project

As can be seen, some of these courses fit directly to one of the topics listed above, and some cover multiple topics within the same course. The exact separation, as well as identification of additional applications areas, are appropriately different from institution to institution, representing the local focus.

2.2 Body of Knowledge

In order to have some expectation, either from the perspective of other institutions that a student may matriculate to, or from the perspective of the community of practice (employers, standards organizations, professional bodies), of there being a common core understanding from students receiving a degree in modeling and simulation, there must be some recognized body of knowledge that provides the basis that the discipline is drawn from.

In recent years, such a body of knowledge has been being catalogued through the combined output of several efforts. This has allowed a variety of different institutions to have some basis by which they can identify what should properly be considered as part of the M&S core discipline, and what is not. In the current case, with such a body of knowledge to rely on, there is still a great deal of appropriate variation – partly because of individual focus, and partly because of the breadth of valid topics.

The body of knowledge, with the explicitly named areas, and with the addition of the recent effort to assess and modernize the definitions for those named areas (Oren 2011b), provides an enumeration of not only the many contributing disciplines, but also the many different sub topics (which are often disciplines of their own study). A large number of different applications areas, and the particular types of modeling and simulation topics that are relevant to them, are also identified in the body of knowledge.

2.3 Learning Objectives of the Experiment

Of the three domains of educational activities that were identified by Bloom and his colleagues – knowledge, attitude and skills – the one most frequently quoted is the knowledge domain, and it is often presented via Bloom's Taxonomy (Bloom 1956). The taxonomy identifies six hierarchical categories of mental skills, that are to be mastered in turn before progressing to the next category. The six categories (presented here in their original nominative format) are:

- Level 1 - Knowledge
- Level 2 - Comprehension
- Level 3 - Application
- Level 4 - Analysis
- Level 5 - Synthesis
- Level 6 – Evaluation

The taxonomy has been the subject of numerous studies and reappraisals over the past decades, and this paper does not seek to add anything new to it. It is enumerated here to identify the sorts of knowledge that the different parts of this experiment is seeking to impart to the students through their participation. It is hoped that a future work will associate elements from the body of knowledge, with the cognitive levels of the taxonomy, and submit the proposed results to a survey of instructors from various universities.

The experimental project can be described simply, and then the individual parts and their learning objectives identified separately. Overall, the project is to devise an environment for distributed simulation, such that the students are responsible for not only the hardware of the environment, but also the software controlling the exchange of information between the distributed simulators. Additionally, the simulators themselves should be developed by the students. Ideally, a variety of different interaction and interoperability requirements would be addressed, requiring the distribution environment as well as the simulators to be sophisticated enough to handle each.

This experiment is intended to be broken into three responsibility areas (experiment layers), each handled by different levels of students, at different stages of their computer science (or modeling and simulation) education. The initial layer of experimentation is to be completed by students at the community college level (equivalent to a four-year institution student in the first two years of completion of a computer science or information technology program). The work done at this layer is the assembling and maintaining of the hardware environment of the distributed simulator network. This involves not only having the various hardware platforms networked together, but also ensuring that at the technical level, the network communications are available to each of the platforms, and to software artifacts running on those platforms. The learning objectives for this layer of the experiment are:

- Knowledge of computer hardware, to be used for a scientific/engineering experiment
- Knowledge of networking infrastructure elements.
- Comprehension of what the network will be used for, and its operating requirements.
- Application of the network infrastructure to constructing the distributed simulator environment. (done in concert with the second layer students)

The second experiment layer is to be completed by students at the four year university level, in the upper years of their undergraduate degree. This work is suitable, again, for students in a computer science or information technology degree program. Equally, students in a computer engineering or modeling and simulation (at the bachelor's degree level) degree would be suitable for this layer. The work done at this layer is the authoring (or modifying) network exchange software that is required by the simulators in order to coordinate and execute the exchange of data over the distributed simulator network. This involves being aware of what the simulators are attempting to do, the operational requirements of their data exchange, and the ability to enable such an exchange over the network provided by the first layer of students. The learning objectives for this layer of the experiment are:

- Knowledge of computer networking protocols

- Knowledge of software development
- Comprehension of what it means to author or customize network protocols for individual use
- Application of a network protocol to developing a "middle ware" that can enable the exchange of data between simulators. (done in concert with the first layer students)
- Analysis of the data interoperability requirements of the simulators.
- Synthesis of the requirements for data interoperability with the capability of the network protocol developed or modified. (done in concert with the third layer students)

The third experiment layer is to be completed by students at the graduate school level, ideally in a master's degree program. This work is suitable, primarily, for modeling and simulation students, but could be attempted by computer science or computer engineering students who have had an exposure to some of the core areas of the modeling and simulation body of knowledge. The work done at this layer is the authoring of simulator software, based on an underlying model, and designed to operate in a distributed simulation environment, making use of the distributed simulator network maintained by the first layer of students, and the network protocol designed by the second layer of students to support the interoperability requirements of the simulators. The learning objectives for this layer of the experiment are:

- Knowledge of modeling.
- Knowledge of simulation.
- Knowledge of software development.
- Comprehension of what it means to develop a simulator software package.
- Application of software development, to enable the simulation of a modeled system through a developed simulator.
- Analysis of data exchange requirements, and representational capability between simulators.
- Synthesis of exchange requirements with the capability of the network protocol developed or modified. (done in concert with the second layer students)
- Evaluation of the successful (or not) application of the selected and developed techniques within a proposed simulation study.

One of the design decisions for this experimental work that led to the development of the distinctive student layers was a requirement to give the students a chance to participate in a much larger project than their individual efforts, and also to enable the students to work together in a project team, representing different stages of design, development, application, and evaluation. Coordinating the layers of students is likely (as in the initial stages of the case reported on here) to be difficult to synchronize initially, so adjustments to the learning objectives that must be accomplished jointly should be expected. In addition to the experiential learning the students will gain from participating in a large distributed project, there are also the benefits to the instructors and institutions involved, from working with each other, such as (1) exposing students in the early stages of their careers to further educational possibilities, (2) exposing younger computer science and information technology students to topics of modeling and simulation, and (3) enabling faculty from varying levels, and varying institutions, a chance to share techniques and experiences.

3 DISTRIBUTED COMPUTING ENVIRONMENT

At the time of authoring this paper (spring 2014), the distributed environment within the instance of the experiment reported on here has not been developed yet. The contributing institutions, faculty members, and students have been identified, and the project has gone through several design and consultation meetings with the involved students, but the development work is intended to be carried out during the

2014-15 academic year, and results will be reported on after. The intended design, and other efforts consulted, can be detailed here.

3.1 Distributed Hardware Environment

In consultation with the faculty members surveyed, and also the students who will be part of the experiment, the hardware selected to form the platforms in the distributed environment will be Raspberry Pi platform. This is a credit card sized computer, well suited for our needs. One of the reasons for selecting this platform is that a number of other student projects, in the past few years at Columbus State University were performed, and there is already some knowledge of and respect for the hardware among the student population, as an experimental environment. The specific version is the Raspberry Pi 512MB Model B, with a 4GB SD card for memory (preloaded with the NOOBs operating system load). The operating system selected is a variant of Debian linux, optimized for the Raspberry Pi, called Raspbian (Wheezy build). Networking will be accomplished via one of the USB ports (it will be up to the second layer of students to decide if they want to use wireless or Ethernet connectivity – both are available at to the hardware).

Alternate hardware options that were considered were Arduino based kit computers and also a collection of older PCs. The Raspberry Pi option was selected by the first layer students, and agreed to by the advising faculty. The project envisions sixteen such hardware platforms making up the environment. These will be rendered bootable, and maintained, by the first layer of students throughout the experiment.

3.2 Distributed Software Environment

The selection of the software to support the distributed environment was a much more in depth decision, and involved several different options being evaluated. The first option was to repeat some work done earlier, and to treat the sixteen Raspberry Pi computers as nodes in a Beowulf cluster, and to rely on MPICH as the basis for the interoperability software. If the hardware selected were older PCs, then relying on an Ubuntu installation of linux would have made this approach more feasible, it was felt. Reference to the Raspberry Pi Beowulf cluster effort reported on from Boise State University (RPiCLUSTER) (Kiepert 2013) was used as a basis for evaluation. However in light of conversations about the interoperability requirements of the simulators, it was felt that the MPICH approach was too complex, and the features it enabled were not needed. The second option was chosen, for the initial experiment. The second option being to rely on the TCP/IP capability made possible through the Raspbian operating system, and to do peer-to-peer communications within the private network that the sixteen platforms exist within.

Realizing that many parallel and distributed computing problems will have to be addressed and solved through the development of custom built TCP/IP based interchanges, the data exchange requirement of the simulators was evaluated. For the first iteration of the experiment, the entire environment would not time-advance until all exchanges were reported on as complete. This is not an extremely efficient distributed computing approach, and additional approaches will be evaluated in later iterations.

Other possible software environments that were considered were developing a Jini type environment, using Apache River, to enable remote procedure calls between the simulators. This was abandoned, as it was thought that many of the principles of distributed simulation would be lost, and also because of the requirements of enabling Apache River on the Raspbian operating system. Additionally, it would tie the simulators to be developed in Java, or with a Java interface. The final (and perhaps best, while most complex) environment that was considered was to rely on the High Level Architecture, and implement a runtime interface that would support HLA 1516 interactions between the simulators. This could be done using an RTI similar to (or based on) either the OpenHLA or the CERTI RTI. While this would give the students the best exposure to current state of the art knowledge for distributed simulation practices, it was felt as if it was too complex for the first iteration. It is held out, however, as a possible strategy for later iterations of the experiment.

4 EXPERIMENTAL SIMULATION SOFTWARE

The development of the simulators that would run in the distributed environment is the layer of the experiment that has progressed the furthest, at the time of writing this paper. Because the environment is intended to survive and serve multiple different simulators and simulation studies, the initial effort has been to look at two different distributed simulators that have different types of data exchange requirements. The constraints on the chosen simulators are that they (1) be student developed, (2) be relatively simple to understand, for purposes of presentation, and also ease of attempting different degrees of adjustment for experimentation, and (3) have some requirement for exchange between distributed versions of the same (or slightly modified) versions. The two initial efforts have been a distributed disease spread model, and a distributed cellular automata model, based on Conway's game of life.

4.1 Disease Spread Model

The disease spread model is based on a version of an SIR (susceptible-infectious-removed) compartmental model (Brauer 2008). The development was done in stages, beginning with a literature review of epidemiology models, and compartmental models in general, and then focusing on modern efforts. The software was developed in python, and is available on request. Inspiration for a revision of the software was based previously published efforts (Keeling and Rohani 2007), and comparison of the approach in different languages was possible, although the initial student developer (a master's degree student in modeling and simulation) preferred to stay with python.

The initial version of the software was a typical application of an SIR compartmental model, with a single homogenous population. The next iteration of development was to introduce multiple populations with slight variations of both the Beta value and the Gamma value, representing changes in the rate of infection and also the progression of the disease. An agent based model, using netlogo, was also developed, to investigate easily the changes made to Beta and Gamma, however this line of development was dropped (a return to an agent based model, rather than having a homogenous population, may occur in the future – however it will be done with python rather than with netlogo). An interest in having a widely varying set of disease parameters encouraged the students to assume that rather than simulating a zombie-like disease, rather than influenza or some other well documented disease. This approach, while still teaching the principles of simulation development and mathematical modeling of biological functions, may be more compelling to the students, especially those at more junior levels of their education, and is with precedent in the literature (Munz 2009).

Various strategies have been discussed and attempted to increase the interaction between the various locations of the disease model, encouraging interaction, and creating the need for data interoperability in the distributed environment. Introducing countermeasures was the first attempt, following the attempt reported on at Texas A&M (Linhart 2012). This allowed the population to be given a rating that could be adjusted for each population center, representing different levels of the population being armed and ready to fight back. In order to make the interactions between populations (each simulator in the distributed environment being responsible for a different population) more interesting, and to exhibit more principles of distributed simulation, rather than use levels of armament, instead the approach was to introduce a factor representing the knowledge of countermeasures, and that knowledge (a factor that would serve as a counter to rate of infection), could be "communicated" between populations. Representing some risk to this, a chance for a negative value to be communicated was introduced, and a series of simulated strategies for expressing the knowledge, accepting the knowledge, and so on is being planned, and is based on a separate modeling effort concerning the modeling of communications, trust, behavior and ethics (Turnitsa 2013).

The distributed simulation approach in this last iteration of the model is for the population center, represented in one simulator, to express a communication about a possible cure to another population center (in a different simulator). That communication is then received by the simulator, along with factors describing the source, trust, and so forth – and the receiving simulator then processes the communication, deciding what to do with it (trust it, and adopt it, or reject it) based on programmed behavior factors.

4.2 Cellular Automata Model

The second approach at developing a software simulator is based on a cellular automata technique. In particular, the initial version (under development as of the time of this writing) of the software is being written in python, and is an implementation of Conway's game of life. Because of department interest, and other project expertise from students at Columbus State University, an implementation of the cellular automata generation technique presented by Stephen Wolfram (Wolfram 2002), was also discussed, and may serve as a third iteration of a software simulator. Conway's game of life (Gardner 1970), of course, is not a simulation of a referent system, however it is based on a model, and implementing it in a distributed manner serves the educational principles of the experiment. Conway's invention was originally termed a "simulation game" by Martin Gardner, when he reported on it in 1970, and the various rules of the game are intended to replicate biological functions.

Implementations of the game of life in python are not new, and a review of the various distinct versions found online and in publications is the subject of an upcoming student presentation. In the version planned and modeled at Columbus State University, the point is the distributed nature of a large board (or gridded surface) for the cellular automaton. The first version is to divide up the board into sub boards, and have each handled by a different simulator, and when the moore's neighborhood around a particular cell needs to be checked, in order to observe the rules for cell birth and death, then the interaction may reach into regions of the board that are dealt with by other simulators. This is the point of the interaction between simulators, and what has to be handled by the students working together at the second and third layer of the experiment. Several strategies have already been discussed, the easiest is to initiate a request for information about a particular cell, and then receive either an "on" or "off" answer. Other strategies include requesting for a string representing the whole adjacent edge from another simulator. In either case, the implementation should be simple enough. The point will be to measure the reaction time, and also to devise a strategy for when the whole board (of all the distributed simulators) may advance forward in time, once all knowledge about every cell and its surrounding moore's neighborhood is known.

In addition to other distributed techniques, future implementations are intended to discover other types of life games, or life simulators, that are possible in cellular automata. Possible variants include (1) the introduction of new rules in one of the distributed simulators, and then strategies for implementing that new rule through contact and pollination across to other regions of the board, (2) colored variations of life, where the different cells may be in different states, allowing for much more complicated rules, (3) a 3d spatial version of the basic game, with rules expanded out to allow for a three dimensional interaction. In all cases, not only must strategies for development (layer three of the experiment) and strategies for interaction (layers two and three working together) must be worked out, but also design and finally interaction with the human audience, through visualization (discussed as a fourth layer for the ongoing experimental group – introducing another group of students, either at the graduate or undergraduate level, but with their own programming responsibilities). Additionally, other implementation techniques have been discussed – including java, and possible cellular DEVS (Wainer 2002), using the CD++ package, available from Carleton university (Cell-Devs 2014).

5 SUMMARY

The effort presented here was devised to allow students from different levels of their academic careers to participate together in a distributed modeling and simulation experiment. As many of these students, especially at the lower levels, will be coming to modeling and simulation for the first time, part of the project is to expose them to some of the core ideas of modeling and simulation – in alignment with what various academic programs consider to be those core ideas, and also in alignment with what is presented in the body of knowledge for modeling and simulation. This exposure is to be as much a part of any measure of success for the experiment, as the actual project development is.

Educational criteria for success has been identified in the form of learning objectives, for the different layers of students involved in the experiment. These learning objectives are in alignment with not only the program at Columbus State University, but also with what the ACM has identified for computer science, as well as what other universities have identified as being core areas for Modeling and Simulation. Additionally, the terms of these learning objectives are in alignment with the terms of the body of knowledge.

In addition to the distributed network for the simulators, as well as a software based distributed simulation exchange technique, there is also the development of the simulators to take part in the experiment. Each of these simulators is subjected to a design process, a development process, coordination with the other elements of the experiment, and finally will be used as the basis for a simulation study, with rigor applied to input analysis, output analysis, as well as verification and validation against the identified models for the simulators that come out of the design process. The initial two areas, currently at different levels of completion, are (1) a distributed disease spread model, relying on a communications model for interpreting the distributed simulation interchanges, and (2) a cellular automata model instancing Conway's game of life, distributed over the environment, and relying several exchange strategies that will be subjected to study and comparison as part of the stimulation study.

As much of this experiment (which will, hopefully, be a multiyear, multi-institution effort) is under process at the time of preparation of this paper, a future report will have to be made on the results of progress. It is hoped that additional work will be done in several areas. First, new simulators and new simulation efforts will be identified by succeeding students. Second, implementation of industry standard distributed simulation techniques (distributed information system, high level architecture, etc.) will be attempted. Third, it is hoped that the students that gain experience in working with simulation software in this experiment, will then be encouraged to participate in the Modeling and Simulation Smackdown, or other student competition events that are designed to challenge M&S students.

REFERENCES

Brauer, F. 2008. Modeling Influenza: Pandemics and Seasonal Epidemics. In Brauer, F., Driessche, P., and Wu, J.*, editors, Mathematical Epidemiology*. Berlin: Springer.

Bloom, B. 1956. *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain.* New York: David McKay.

Cell-Devs 2014. "Introduction to CD++," available online at http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page last checked June 30, 2014.

Gardner, M. 1970. "Mathematical Games – the fantastic combinations of John Conway's new solitaire game 'life'," *Scientific American.* 223, pp. 120-123.

Keeling, M. and P. Rohani. 2007. *Modeling Infectious Diseases in Humans and Animals*. New Jersey: Princeton University Press.

Kiepert, J. 2013. "RPiCluster: Creating a Raspberry Pi-based Beowulf Cluster," available online at http://coen.boisestate.edu/ece/raspberry-pi/ last checked May 15, 2014.

Linhart, J. 2012. "Mathematical Modeling of a Zombie Outbreak," available online at http://www.math.tamu.edu/~jmlinhart/zombies2012.pdf last checked May 15, 2014.

Munz, P., I. Hudea, J. Imad, R. Smith?. 2009. "When zombies attack! Mathematical modeling of an outbreak of zombie infection," in Tchuenche, J. and C. Chiyaka, editors, *Infectious Disease Modelling Research Progress*. Berlin: Springer.

Ören, T. 2011a. "The Many Facets of Simulation through a Collection of about 100 Definitions," *SCS Modeling and Simulation Magazine*. April 2011.

Ören, T. 2011b. "A Critical Review of Definitions and About 400 Types of Modeling and Simulation," *SCS Modeling and Simulation Magazine*. July 2011.

Sarjoughian, H., J. Cochran, J. Collofello, J. Goss, and B. Zeigler. 2004. "Graduate Education in Modeling & Simulation: Rationale and Organization of an Online Masters Program," In *Proceedings, 2004 Summer Computer Simulation*. San Jose, CA.

Turnitsa, C. 2013. "Communication Model Elements for Societal Behavior Representation using Agent Based Models," in *Proceedings, 2013 Fall Simulation Interoperability Workshop*. Orlando, FL.

Vakilzadian, H. 2003. "A Graduate Program in Modeling and Simulation," In *Proceedings of International Conference on Simulation and Multimedia in Engineering Education*, pp. 49--53, January 2003.

Vakilzadian, H. and D. Möller. 2009. "Simulation in undergraduate education initiative in electrical engineering," In *Proceedings, 2009 Spring Simulation Multiconference*. San Diego, CA.

Wainer, G. 2002. "CD++: a toolkit to define discrete-even models," in *Software, Practice and Experience*. Wiley, 32:3.

Wolfram, S. 2002. *A New Kind of Science*. Champaign, IL: Wolfram Media.

AUTHOR BIOGRAPHY

CHARLES TURNITSA is Assistant Professor for Modeling and Simulation, in the TSYS School of Computer Science, at Columbus State University, in Columbus GA. He has previously been part of the research staff at the Virginia Modeling Analysis and Simulation Center at Old Dominion University in Norfolk, and is currently consulting with Joint Staff J7, through General Dynamics Information Technology, and serving as the project lead on a data modeling project, intended to support the new JLVC2020 simulation architecture.